Metoda elementów skończonych dla informatyków (AD 2004)

Krzysztof Banaś

Spis treści

Rozdział 1

Wstęp

1.1 Krótka historia metody elementów skończonych

Metoda elementów skończonych (MES) jest typowym przykładem metody należącej do nauki o obliczeniach, która będąc początkowo metodą obliczeniową w konkretnej dziedzinie zastosowań, zyskała później powszechną akceptację i stała się narzędziem o uniwersalnym zastosowaniu.

Początki MES sięgają lat czterdziestych i pięćdziesiątych XX w., kiedy stosowana była do określania stanu naprężeń i odkształceń w konstrukcjach, najpierw samolotowych, a później także budowlanych [124, 8]. Termin „metoda elementów skończonych"[1] został po raz pierwszy użyty w artykule [55] w roku 1960. W 1967 roku wydany został pierwszy, po dziś dzień najpopularniejszy dzięki kolejnym wydaniom, podręcznik MES [184]. Już w tym okresie skrystalizowały się podstawowe techniki obliczeniowe MES, takie jak aproksymacja geometrii obszarów obliczeniowych, całkowanie numeryczne czy specjalne metody agregacji macierzy sztywności i rozwiązywania układów równań liniowych.

W latach sześćdziesiątych XX w. nastąpił szybki rozwój matematycznej teorii MES. Wykorzystano wcześniejsze prace dotyczące metod aproksymacji Ritza i Galerkina, a także aproksymacji kawałkami liniowej czy wielomianowej. Te wcześniejsze prace matematyczne (por. np. [58]) rozwijane były poza kontekstem MES, jako metody o praktycznej komputerowej realizacji. Dopiero lata sześćdziesiąte zaczęły przynosić dowody zbieżności i oszacowania

[1] Angielski termin *the finite element method* jest w polskiej literaturze tłumaczony dwojako: w analizie numerycznej używa się formy „metoda elementu skończonego" (wiernie oddającej treść angielską), w naukach inżynierskich przyjęło się stosować nazwę „metoda elementów skończonych". Ta druga forma, nawiązująca do tytułu polskiego tłumaczenia klasycznego podręcznika Zienkiewicza [183], jest konsekwentnie stosowana w niniejszej pracy.

a priori błędu aproksymacji dla praktycznych przybliżeń skończenie elementowych. Dalszy rozwój podstaw matematycznych metody elementów skończonych doprowadził w latach siedemdziesiątych i osiemdziesiątych ubiegłego wieku do popularyzacji MES jako bardzo ogólnego narzędzia aproksymacji szerokiej klasy zagadnień brzegowych i początkowo-brzegowych w wielu dziedzinach fizyki i techniki [100, 101, 126].

Istotnym elementem rozwoju matematycznej teorii MES było opracowanie, począwszy od lat osiemdziesiątych XX w., technik szacowania błędu aproksymacji na podstawie uzyskanego rozwiązania przybliżonego, tak zwanego szacowania a posteriori [10]. Wraz z metodami szacowania błędu wprowadzono techniki adaptacji, prowadzące do optymalizacji procesu aproksymacji – redukcji błędu jak najmniejszym kosztem obliczeniowym [9]. Strategia adaptacji zawsze polega na rozwiązaniu zadania na zgrubnej siatce początkowej, a następnie na odpowiednio ukierunkowanej zmianie parametrów decydujących o błędzie. Rozwijane techniki obejmowały adaptacje typu: r – przesuwanie węzłów siatki MES, h – lokalne zmniejszanie rozmiaru elementów, p – lokalne zwiększanie stopnia aproksymacji, i w końcu, najbardziej złożoną, hp – kombinację dwóch poprzednich. Adaptacja typu hp, choć w teorii najbardziej efektywna (jedyna dająca eksponencjalną redukcję błędu aproksymacji w funkcji liczby stopni swobody [11]), ze względu na swą złożoność nie zyskała szerokiej akceptacji. Do jej rozwoju przyczyniają się ostatnio prace nad *automatyczną* adaptacją hp [62], która zdejmuje z użytkownika ciężar obliczania oszacowania błędu i wyboru strategii adaptacji. Technika ta jest złożona w programowaniu. Jednym z celów badań przedstawianych w niniejszej książce jest opracowanie architektury programów MES, dla której implementacja adaptacji typu hp byłaby ułatwiona.

Adaptacyjna metoda elementów skończonych, mimo sukcesów w wielu dziedzinach, takich jak np. mechanika płynów czy statyczne zagadnienia teorii sprężystości, w pewnych obszarach zastosowań nie przyniosła spodziewanych efektów i nie zyskała popularności. Przykładami takich obszarów są: dynamika konstrukcji, gdzie stosowanie adaptacji może zaburzać przebieg symulowanych procesów, czy symulacje przepływów z reakcjami chemicznymi, gdzie wymagane jest lokalne przestrzeganie praw zachowania.

Zagadnienie poszerzania obszaru zastosowań adaptacyjnej MES nie jest rozważane w niniejszej pracy. Omawiane jest jej praktyczne wykorzystanie, łącznie z równoległą implementacją komputerową, w dziedzinach, w których teoretyczne i praktyczne zyski z jej zastosowania przesądzają o opłacalności użycia. Do dziedzin takich należą dynamika gazów i ogólne problemy eliptyczne, z

których czerpane są przykłady ilustrujące omawiane zagadnienia teoretyczne. W książce rozważane są algorytmy i techniki implementacji wykorzystywane w tych właśnie dziedzinach. Jednak mimo pozostawienia wielu metod i rozwiązań poza obszarem analiz pracy, jednym z jej celów jest prezentacja algorytmów i sposobów realizacji obliczeń o możliwie uniwersalnym charakterze.

Im bardziej metoda elementów skończonych staje się wyrafinowana i im potężniejsze komputery dostępne są badaczom, tym trudniejsze problemy próbuje się rozwiązać za jej pomocą [126, 41]. Charakterystyczne dla ostatnich lat połączenie wzrastającej złożoności rozwiązywanych problemów, stosowania wyrafinowanej aproksymacji oraz komplikacji niesionych przez równoległe i rozproszone wykonanie programów, powoduje zwiększenie zainteresowania implementacyjnymi aspektami MES. Od lat dziewięćdziesiątych ubiegłego wieku pojawiają się prace poświęconych stronie informatycznej MES. Badania nad efektywną, tak z punktu widzenia tworzenia oprogramowania, jak i jego wykonania, realizacją MES stanowią dziś odrębny i istotny kierunek rozwoju MES oraz nauk obliczeniowych. Należy tu wymienić, poza licznymi publikacjami w czasopismach, także wydania specjalne czasopism [72, 132], publikacje zbiorowe [6, 108] czy odrębne sesje konferencyjne [116, 59]. Nacisk kładziony początkowo na wykonanie równoległe, jako najistotniejszy informatyczny aspekt MES, ustępuje w ostatnich latach bardziej ogólnej i złożonej wizji, w ramach której efektywność realizacji równoległej jest tylko jednym z wielu kryteriów informatycznych, spełnienia których wymaga się od programów MES.

1.2 Notacja

W pracy wykorzystanych jest kilka konwencji dotyczących oznaczeń:

- liczby skalarne pisane są czcionką normalną, wektory i macierze (także funkcje wektorowe i macierzowe) pogrubioną, wektory małą literą, macierze wielką,

- wektory i macierze o małej liczbie wyrazów (związanych z różnymi kierunkami w przestrzeni geometrycznej lub z wektorowym charakterem rozwiązywanych równań różniczkowych) pisane są czcionką pochyłą, wektory i macierze algebraiczne o dużej liczbie wyrazów, związane z rozwiązywanymi układami równań liniowych, pisane są czcionką normalną,

- pojedyncze składowe wektorów i elementy macierzy pisane są czcionką normalną (zawsze pochyłą), dla odróżnienia od wariantów i podbloków

związanych z wektorami i macierzami, które pisane są czcionką pogrubioną (pochyłą lub normalną, zależnie od czcionki podstawowego wektora czy macierzy),

- $()^T$ oznacza transpozycję wektora lub macierzy, np. $\boldsymbol{u}^T$,
- iloczyn skalarny oznaczany jest symbolem „$\cdot$”, $\boldsymbol{u} \cdot \boldsymbol{u} = \boldsymbol{u}^T \boldsymbol{u}$,
- pochodne cząstkowe po czasie i kierunkach przestrzennych zapisywane są z użyciem przecinka: $_{,t} = \frac{\partial}{\partial t}, \quad _{,i} = \frac{\partial}{\partial x_i}$,
- powtórzone indeksy oznaczają zastosowanie konwencji sumacyjnej Einsteina, np. $u_i u_i = \boldsymbol{u} \cdot \boldsymbol{u} = \sum_{i=1}^{3} u_i^2, \quad u_{i,i} = \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i}$.

Prezentowane algorytmy zapisane są przy użyciu intuicyjnie, w intencji, zrozumiałego pseudokodu. Operacje opisywane są słownie i/lub stosując ścisłą notację podstawień i wywołań. Różnice wcięć zawsze oznaczają różne poziomy zagłębień pętli i bloków warunkowych algorytmów. Znak „/” na końcu linii oznacza, że jest ona kontynuowana w linii następnej.

Wykaz podstawowych symboli stosowanych w pracy znajduje się w Dodatku A (pominięte są symbole użyte jednorazowo, objaśniane w miejscu zastosowania).

Rozdział 2

Algorytmy obliczeń metodą elementów skończonych

Podstawowe równania wiekszości dziedzin fizyki i techniki (np. równania Maxwella dla elektromagnetyzmu, Naviera-Stokesa dla mechaniki płynów, równania teorii sprężystości, równania Schrödingera dla mechaniki kwantowej) mają postać stacjonarnych lub niestacjonarnych równań różniczkowych cząstkowych. Metoda elementów skończonych jest stosowana do ich dyskretyzacji albo jako wyłączna technika, albo jako element szerszej metodologii.

Niniejszy rozdział przedstawia podstawowe algorytmy wykorzystywane przy aproksymacji powyższych równań, ze szczególnym uwzględnieniem elementów specyficznych dla dyskretyzacji MES i decydujących o wydajności obliczeń dla problemów wielkiej skali.

2.1 Zagadnienia modelowe

Do zilustrowania działania analizowanych i testowanych w dalszych częściach pracy algorytmów i metod numerycznych wykorzystane są dwa konkretne przykłady zagadnień różniczkowych. Jednym z nich są równania Eulera opisujące przepływy idealnych gazów ściśliwych, drugim, problem czystej dyfuzji – ogólne zagadnienie eliptyczne w postaci równania Laplace'a w prostym obszarze obliczeniowym. Pierwszy problem pojawiać się będzie przy omawianiu technik klasycznej dyskretyzacji MES, drugi ilustrować będzie zastosowanie nieciągłej dyskretyzacji Galerkina.

2.1.1 Równania Eulera dynamiki gazów ściśliwych

Równania Eulera dla przepływów nielepkich gazów ściśliwych cechują się silną nieliniowością i wysoką niestabilnością, związaną między innymi z występowaniem fal uderzeniowych, oraz zmiennym, z matematycznego punktu widzenia, charakterem. Równania stacjonarne mają charakter eliptyczny w przypadku liczb Macha[1] mniejszych od jeden (przepływy poddźwiękowe) oraz hiperboliczny dla liczb Macha większych od jeden (przepływy naddźwiękowe). Dla przepływów okołodźwiękowych występują obszary eliptyczne i hiperboliczne. Przy uwzględnieniu niestacjonarności równania stają się równaniami hiperbolicznymi.

Mimo tak złożonego charakteru, równania Eulera można przedstawić w prostej postaci, przez co dobrze nadają się do ilustracji omawianych zagadnień dyskretyzacji. Zapisane w postaci zachowawczej wyglądają następująco:

$$\boldsymbol{u}(\boldsymbol{x},t)_{,t} + \boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u})_{,i} = 0 \tag{2.1}$$

gdzie $\boldsymbol{u} = (\rho, \rho v_j, \rho e)^T$ oznacza wektor zmiennych zachowawczych (gęstość, przestrzenne składowe pędu, energia całkowita), a $\boldsymbol{f}_i^{\mathrm{E}} = (\rho v_i, \rho v_i v_j + p\delta_{ij}, (\rho e + p)v_i)^T$ jest wektorem strumieni eulerowskich (p – ciśnienie, v_i – i-ta składowa prędkości).

Metody dyskretyzacji równań Eulera wykorzystywane są bezpośrednio w zagadnieniach, dla których oferowana przez model idealnego gazu nielepkiego dokładność jest wystarczająca (np. opływy wokół profili lotniczych bez uwzględnienia warstwy przyściennej), a także są podstawą metod dyskretyzacji dla bardziej złożonych modeli, takich jak np. równania Naviera-Stokesa, opisujące przepływy gazów z uwzględnieniem lepkości.

2.1.2 Liniowe zagadnienie dyfuzji – równanie Laplace'a

Przykładowym zagadnieniem dyfuzji jest skalarne równanie Laplace'a:

$$\Delta u = \Delta u_{\mathrm{ex}} \tag{2.2}$$

z warunkiem brzegowym Dirichleta wziętym z rozwiązania dokładnego:

$$u_{\mathrm{ex}} = \exp\left(-x^2 - y^2 - z^2\right) \tag{2.3}$$

[1] Liczba Macha jest definiowana jako lokalny, w danym punkcie przestrzeni, stosunek prędkości gazu $\boldsymbol{v}$ do prędkość dźwięku c, $M = \boldsymbol{v}/c$. Przy charakteryzowaniu zadań opływu stacjonarnych profili lotniczych (wykorzystywanych jako zadania przykładowe w niniejszej pracy) stosuje się liczbę Macha w punktach z dala od profili, w niezaburzonej strudze gazu.

Obszarem obliczeniowym jest jednostkowy sześcian $[0,1] \times [0,1] \times [0,1]$. Znajomość rozwiązania dokładnego jest wykorzystywana do oceny jakości aproksymacji.

Mimo swojej prostoty powyższe zagadnienie ma istotne znaczenie zarówno praktyczne, jako model wielu zjawisk fizycznych, jak i bardziej teoretyczne, jako podstawa wielu twierdzeń dotyczących metod aproksymacji, czy jako punkt wyjścia analiz dotyczących bardziej złożonych, choć zbliżonych problemów (np. problemy liniowej teorii sprężystości czy nieliniowe równania potencjału).

2.2 Dyskretyzacja czasowa

Nieliniowe, zależne od czasu równania różniczkowe są najbardziej złożoną formą równań dyskretyzowanych przy użyciu metodologii, w skład których wchodzi metoda elementów skończonych. W przypadku problemów sprzężonych zagadnienie może składać się z szeregu układów takich równań. Niezależnie od całościowej strategii rozwiązywania, konieczna jest dyskretyzacja czasowa i przestrzenna pojedynczego, w ogólnym przypadku wektorowego, równania.

W rozważanych w pracy strategiach, jako metoda dyskretyzacji przestrzennej zawsze stosowana jest metoda elementów skończonych. Może ona być łączona z różnorodnymi metodami dyskretyzacji czasowej [43, 32].

Z punktu widzenia MES i obliczeń wielkiej skali jedynym istotnym kryterium oceny metod dyskretyzacji czasowej jest liczba i charakter generowanych problemów przestrzennych. Czynnikiem, który wpływa na liczbę problemów jest wybór schematu całkowania (jawny lub niejawny, wyższego lub niższego rzędu). Wyższy rząd metody oznacza większą dokładność, ale zazwyczaj zwiększone nakłady obliczeniowe w pojedynczym kroku czasowym. Długość kroku czasowego zależy od cech symulowanego zjawiska i stabilności schematu całkowania. Celem specjalnych technik adaptacji długości kroku czasowego jest przeprowadzenie całości symulacji przy jak najmniejszej liczbie rozwiązywanych problemów [168].

2.2.1 Metody jedno- i wielokrokowe

W literaturze dotyczącej metod dyskretyzacji czasowej wiele miejsca zajmują metody wielokrokowe (*multi-step*), które dla aproksymacji w kolejnej chwili wykorzystują rozwiązania z kilku chwil poprzednich. Aby stosować metody

wielokrokowe, konieczne jest użycie tej samej dyskretyzacji przestrzennej w różnych krokach czasowych.

Z tej przyczyny adaptacyjna MES nie nadaje się do łączenia z metodami wielokrokowymi. Stosowana metodologia polega na użyciu metod jednokrokowych (*single-step*), wykorzystujących tylko rozwiązanie z chwili poprzedniej. Zastosowanie adaptacji połączone jest z dokonaniem interpolacji lub projekcji rozwiązania na siatkę zaadaptowaną. To nowe rozwiązanie staje się punktem wyjścia dla obliczeń w następnym kroku czasowym. W każdym kroku rozwiązuje się pojedynczy problem lub, w przypadku metod wielostopniowych (*multi-stage*), kilka problemów przestrzennych.

Zasadniczym podziałem w ramach metod jednokrokowych jest podział na metody jawne, gdzie szukane rozwiązanie w kolejnej chwili pojawia się w sformułowaniu tylko w operatorze dyskretyzacji czasowej, i metody niejawne, gdzie szukane rozwiązanie występuje także w innych wyrazach – liniowych bądź nieliniowych.

2.2.2 Jawne metody dyskretyzacji czasowej

Metody jawne w połączeniu z dyskretyzacją skończenie elementową generują układy równań, które albo można przybliżać układami z macierzami diagonalnymi, a więc nie wymagającymi rozwiązywania, albo można łatwo rozwiązać metodami iteracyjnymi, dzięki dominacji wyrazów na przekątnej głównej. W przypadku techniki skupiania wyrazów na przekątnej głównej (*mass lumping*), macierzy układu można nie tworzyć w ogóle, co prowadzi do oszczędności także pamięci operacyjnej.

Wadą metod jawnych jest względna stabilność – zależność długości maksymalnego dopuszczalnego kroku czasowego od rozmiaru komórek występujących w dyskretyzacji przestrzennej. Dla adaptacyjnej MES stosowanej w obliczeniach wielkiej skali, dla której zmniejszanie rozmiaru elementów jest podstawową techniką adaptacji, metody jawne prowadzą do istotnego zwiększenia nakładów obliczeniowych, poprzez zwiększenie liczby kroków czasowych.

2.2.3 Niejawne metody dyskretyzacji czasowej

Metody niejawne prowadzą zawsze do rozwiązania układu równań. W przypadku układu równań nieliniowych istotny jest stopień nieliniowości, od którego zależeć będzie wydajność iteracyjnego algorytmu rozwiązania problemu nieliniowego. W przypadku układu równań liniowych najważniejszym kryterium jest to, czy daje się on rozwiązać metodami iteracyjnymi oraz jaka jest

szybkość ich zbieżności. Zwiększone, w porównaniu z metodami jawnymi, nakłady obliczeniowe na rozwiązanie równań liniowych równoważone są większą stabilnością metod niejawnych, a co za tym idzie, dłuższym dopuszczalnym krokiem czasowym. W przypadku adaptacyjnej MES ma to istotne znaczenie.

2.2.4 Porównanie wybranych technik dyskretyzacji czasowej równań Eulera

Jako przykład zależności pojawiających się przy dyskretyzacji czasowej równań niestacjonarnych, przedstawione jest numeryczne porównanie kilku metod zastosowanych do całkowania czasowego równań Eulera dynamiki gazów ściśliwych (patrz p. 2.1.1).

Sformułowania dyskretne dla wybranych metod całkowania czasowego

Ze względu na hiperboliczny charakter równań (2.1) zastosowana bezpośrednio aproksymacja przestrzenna Galerkina prowadzi do niestabilnych rozwiązań z silnymi oscylacjami. Dlatego też aproksymacja przestrzenna musi zawierać formę stabilizacji rozwiązań. W przedstawionym poniżej sformułowaniu zastosowano stabilizację w postaci wyrazów drugiego rzędu ze specjalnie dobranymi macierzami sztucznej lepkości $\boldsymbol{K}_{ij}(\boldsymbol{u}, \nabla \boldsymbol{u})$ [27, 16]. Wyrazy takie pojawiają się w różnych wariantach metody SUPG [160, 87, 2, 86], szeroko stosowanej w mechanice płynów. Sformułowanie słabe, będące podstawą aproksymacji MES, uzyskane dzięki standardowym technikom mnożenia równań (2.1) przez funkcje testujące oraz całkowania po obszarze obliczeniowym Ω (z wykorzystaniem wzorów na uogólnione całkowanie przez części), ma dla powyższej stabilizacji postać:

Znajdź taką funkcję $\boldsymbol{u} \in \boldsymbol{V}_h$, aby dla każdej funkcji testującej $\boldsymbol{w} \in \boldsymbol{V}_h$ spełnione było:

$$\int_\Omega \boldsymbol{w}^T \boldsymbol{u}_{,t} - \int_\Omega \boldsymbol{w}_{,i}^T \boldsymbol{f}_i^{\mathrm{E}} d\Omega + \int_\Gamma \boldsymbol{w}^T \boldsymbol{f}_i^{\mathrm{E}} n_i d\Gamma + \\ + \int_\Omega \boldsymbol{w}_{,i}^T \boldsymbol{K}_{ij} \boldsymbol{u}_{,j} d\Omega = 0 \tag{2.4}$$

gdzie $\boldsymbol{n}$ jest wektorem jednostkowym, normalnym zewnętrznie względem brzegu obszaru obliczeniowego Γ, a $\boldsymbol{V}_h$ oznacza przestrzeń funkcji aproksymujących (dla przykładów z dynamiki gazów ściśliwych zawsze będzie ona

przyjmowana jako przestrzeń funkcji kawałkami liniowych dla zadanej triangulacji). W sformułowaniu (2.4) pominięto szczegóły uwzględnienia warunków brzegowych, nieistotne dla doboru schematu aproksymacji czasowej.

W celu ilustracji konsekwencji zastosowania różnych metod dyskretyzacji czasowej, przedstawione zostanie porównanie trzech metod całkowania czasowego: metody jawnej Eulera, metody niejawnej zlinearyzowanej oraz nieliniowej niejawnej metody Eulera.

Wszystkie te metody prowadzą do sekwencji zagadnień jednokrokowych dających się przedstawić w ujednoliconej postaci:

Dla danego w chwili t_n rozwiązania $\boldsymbol{u}^n \in \boldsymbol{V}_h$ znajdź $\boldsymbol{u}^{n+1} \in \boldsymbol{V}_h$ w chwili t_{n+1} ($t_{n+1} - t_n = \Delta t$) takie, że dla każdej funkcji testującej $\boldsymbol{w} \in \boldsymbol{V}_h$ spełnione jest:

$$\begin{aligned}
&\frac{1}{\Delta t}\int_\Omega \boldsymbol{w}^T\boldsymbol{u}^{n+1}d\Omega - \alpha\int_\Omega \boldsymbol{w}_{,i}^T\boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u}^{n+1})d\Omega + \\
&+\alpha\int_\Gamma \boldsymbol{w}^T\boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u}^{n+1})n_i d\Gamma + \beta\int_\Omega \boldsymbol{w}_{,i}^T\boldsymbol{K}_{ij}(\boldsymbol{u}^{n+1})\boldsymbol{u}_{,j}^{n+1}d\Omega = \\
&= \frac{1}{\Delta t}\int_\Omega \boldsymbol{w}^T\boldsymbol{u}^n d\Omega + (1-\alpha)\int_\Omega \boldsymbol{w}_{,i}^T\boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u}^n)d\Omega + \\
&-(1-\alpha)\int_\Gamma \boldsymbol{w}^T\boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u}^{n+1})n_i d\Gamma - (1-\beta)\int_\Omega \boldsymbol{w}_{,i}^T\boldsymbol{K}_{ij}(\boldsymbol{u}^n)\boldsymbol{u}_{,j}^n d\Omega
\end{aligned} \tag{2.5}$$

Metody różnią się wartościami parametrów α i β. Dla jawnej metody Eulera $\alpha = 0$, $\beta = 0$, dla metody niejawnej zlinearyzowanej $\alpha = 0$, $\beta = 1$, dla niejawnej metody Eulera $\alpha = 1$, $\beta = 1$.

Trzy powyższe sformułowania zostały uzyskane jako wersje bardzo prostej dyskretyzacji czasowej metodą linii [18]. Okazuje się jednak, że są one także szczególnymi przypadkami dla innych technik całkowania czasowego. Metoda niejawna liniowa ma podobną postać jak metoda Taylora-Galerkina [65, 173, 156, 63], będąca wersją skończenie elementową metody Laxa-Wendroffa [109]. Jedyna różnica, poza odmienną techniką uzyskania równań, polega na modyfikacji macierzy regularyzujących. Z kolei niejawna nieliniowa metoda Eulera może zostąć otrzymana w rezultacie zastosowania nieciągłej aproksymacji czasowej Galerkina [93, 95, 158], z kawałkami stałymi funkcjami bazowymi. Sformułowanie (2.5) ze specyficznymi wartościami parametrów α i β może stanowić przypadek szczególny dla szerokiej klasy metod dyskretyzacji czasowej.

W rozważanym przypadku dyskretyzacji równań Eulera porównywane metody poddane są dalszej transformacjii. Dla metody jawnej stosuje się skupienie wyrazów na przekątnej głównej macierzy układu równań liniowych. Wpro-

wadza to dodatkowy błąd aproksymacji, jednak pozwala uniknąć rozwiązywania układu równań (współczynniki na przekątnej głównej, jako liniowe, są obliczane dla danej siatki tylko raz). W tym wypadku cała procedura rozwiązania sprowadza się do numerycznego całkowania i uaktualniania wartości stopni swobody w węzłach siatki MES. Numeryczne całkowanie dotyczy tylko wektora prawej strony, obejmuje jednak złożone obliczanie strumieni eulerowskich i wyrazów stabilizujących. Schemat jawny dopuszcza kroki czasowe, które spełniają warunek Couranta-Friedrichsa-Levy'ego ograniczający liczbę *CFL* [90]:

$$CFL = \frac{\Delta t(c + |\boldsymbol{v}|)}{h} \tag{2.6}$$

gdzie: c – prędkość dźwięku, $\boldsymbol{v}$ – prędkość gazu, h – liniowy rozmiar elementu. W przypadku liniowych problemów jednowymiarowych liczba *CFL* dla schematu jawnego Eulera musi być mniejsza od 1. Dla zagadnień nieliniowych w złożonych obszarach wielowymiarowych to ograniczenie jest jeszcze bardziej restrykcyjne (w praktyce wartości rzędu 0.2–0.5).

Dla pierwszej wersji metody niejawnej ($\alpha = 0$, $\beta = 1$) po lewej stronie układu równań pozostają tylko wyrazy regularyzujące. Dzięki temu możliwe jest przeprowadzenie prostej linearyzacji, w której współczynniki macierzowe $\boldsymbol{K}_{ij}$ obliczane są dla znanej funkcji $\boldsymbol{u}^n$. Umieszczenie wyrazów drugiego rzędu po stronie niejawnej powoduje konieczność rozwiązania układu równań. Z drugiej strony, ten właśnie zabieg poprawia stabilność schematu całkowania. Dla zagadnień liniowych wielowymiarowych warunek stabilności ma postać $CFL < 1$ [61, 146]. Przy krokach czasowych tego rzędu układ równań liniowych jest dobrze uwarunkowany, dzięki występowaniu ilorazu $\frac{1}{\Delta t}$ w wyrazach na przekątnej głównej.

Niejawny nieliniowy schemat Eulera jest jedynym, dla którego nie istnieje ograniczenie długości kroku czasowego [25, 26]. Wymaga on jednak rozwiązania na każdym kroku układu równań nieliniowych. Szczegółom rozwiązywania układów równań nieliniowych poświęcony jest p. 2.3. Każda z przedstawionych tam procedur składa się z rozwiązania szeregu równań liniowych, co oznacza zwielokrotnienie nakładów obliczeniowych w porównaniu ze schematami: jawnym i niejawnym liniowym.

Ocena złożoności obliczeniowej algorytmów dyskretyzacji czasowej

Pamięciowa złożoność obliczeniowa każdej z metod całkowania czasowego zależy od liczby wektorów niewiadomych koniecznych do jednoczesnego przechowywania w pamięci operacyjnej. Metody wyższego rzędu najczęściej wymagają

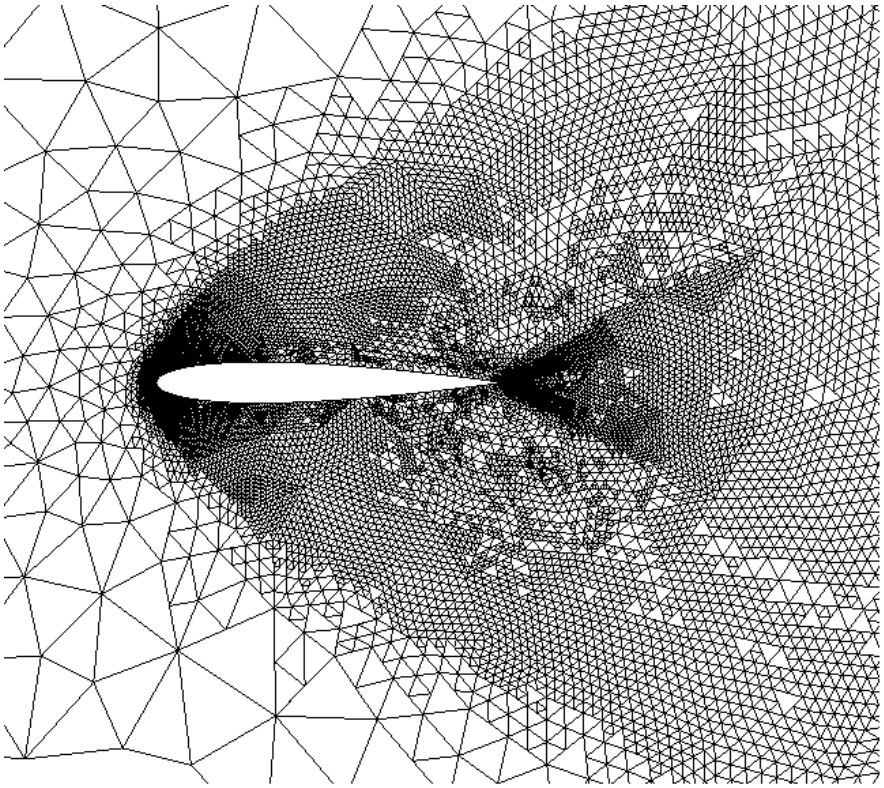
większej liczby przechowywanych wektorów niż metody niższego rzędu.

Złożoność czasowa wynika z sumy kosztów poszczególnych etapów obliczeń. Metody niejawne potrzebują zbliżonej do metod jawnych ilości czasu na całkowanie numeryczne i agregację (składanie) macierzy układu równań liniowych. Czas rozwiązania układu zależy od użytej metody. W opisywanych poniżej eksperymentach zastosowano metodę GMRES z poprawą uwarunkowania macierzy poprzez iteracje Gaussa-Seidla (metody te zostaną omówione szerzej w p. 2.7). Dla przedstawianego przypadku adaptacyjnych symulacji przepływów wokół dwuwymiarowych profili czas pojedynczej iteracji jest przeciętnie kilka razy krótszy od czasu całkowania numerycznego. Dla metody niejawnej zlinearyzowanej rozwiązanie układu równań liniowych wymaga wykonania kilkunastu iteracji. W połączeniu z możliwością użycia dłuższych kroków czasowych daje to zbliżony do metod jawnych koszt całkowania czasowego, w sensie wymaganego nakładu obliczeniowego (czasu CPU) na jednostkę czasu symulacji [13].

Metody nieliniowe mają odmienną charakterystykę od metod liniowych. W przypadku kiedy fizyka procesu wymusza zastosowanie krótkich kroków czasowych, metody nieliniowe, wymagające znacznie większych nakładów obliczeniowych w pojedynczym kroku czasowym, są zdecydowanie wolniejsze od liniowych. Kiedy jednak symulacja dopuszcza stosowanie kroków, dla których liczba *CFL* może sięgać kilkunastu lub więcej, ich wydajność może być lepsza niż metod liniowych.

Ten ostatni przypadek ma miejsce np. w technice kontynuacji w pseudoczasie (*pseudo-time continuation*) [98]. Stosuje się ją często w zagadnieniach silnie nieliniowych, takich jak równania Eulera, dla których zawodzą klasyczne techniki rozwiązywania zagadnień nieliniowych. W metodzie tej do rozwiązywanego zagadnienia stacjonarnego dodaje się sztuczny człon zależny od czasu (wzięty np. ze sformułowania niestacjonarnego) i stosuje całkowanie czasowe, jako sposób iteracji do stanu ustalonego. Otrzymane na skutek dyskretyzacji czasowej problemy jednokrokowe charakteryzują się słabszą nieliniowością od zagadnienia pierwotnego (faza zmian chwilowych, *transient phase*, na ścieżce od stanu początkowego do ustalonego przestaje być częścią pojedynczego problemu nieliniowego, a zostaje zamieniona na szereg nieliniowych kroków pseudoczasowych). Ułatwia to zastosowanie metod iteracyjnych do rozwiązania powstałych równań liniowych.

W technice całkowania pseudoczasowego można jeden globalny krok czasowy Δt zamienić na lokalne kroki Δt_{loc}, obliczane odrębnie dla pojedynczych węzłów i elementów siatki, korzystając z lokalnych ograniczeń liczby *CFL* (2.6)

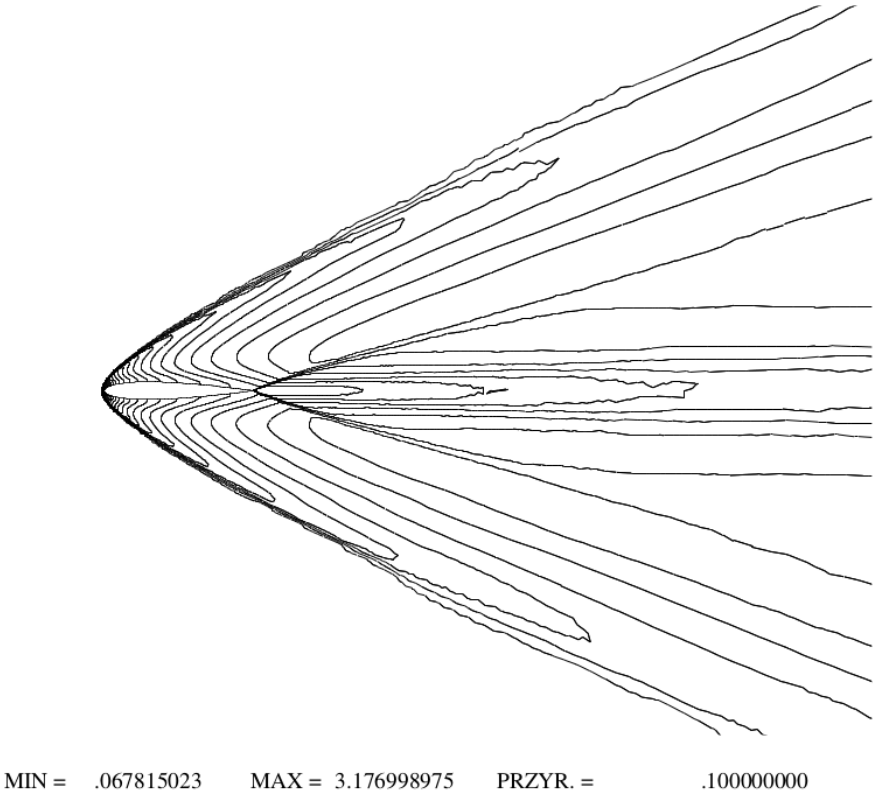


Rys. 2.1. Symulacja naddźwiękowego przepływu wokół profilu NACA0012: szczegół trzykrotnie zaadaptowanej siatki MES o 14 312 węzłach

[90]. Wprowadza się wtedy globalny parametr CFL, a lokalne długości kroku pseudoczasowego oblicza się z wzoru:

$$\Delta t_{\mathrm{loc}} = \mathrm{CFL} \cdot \frac{h}{c + |\boldsymbol{v}|} \tag{2.7}$$

Można także zrezygnować z dokładnego rozwiązywania układów równań, tak nieliniowych, jak i liniowych, a jednocześnie wprowadzić strategię doboru długości kroków czasowych (wartości parametru CFL) maksymalizującą szybkość zbieżności [82].

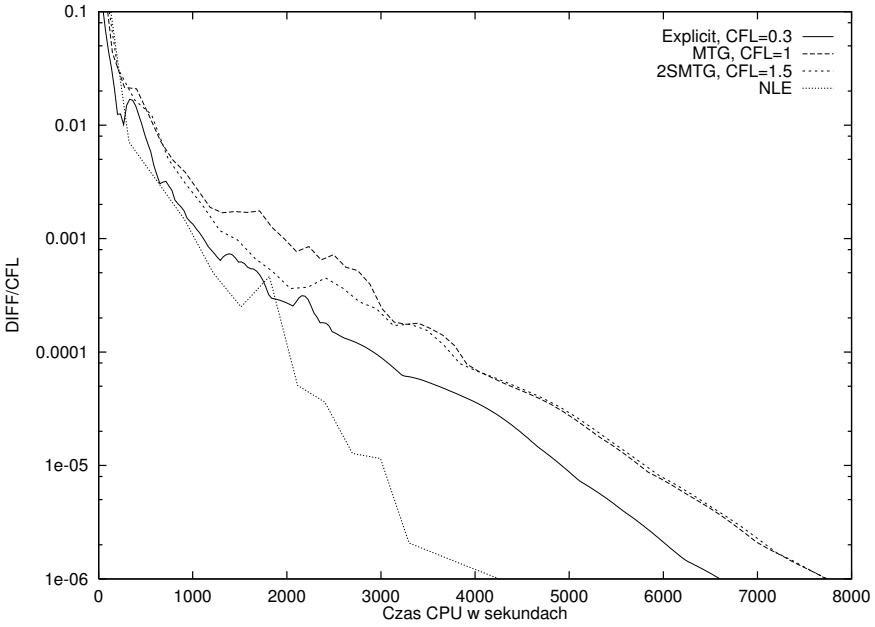


Rys. 2.2. Symulacja naddźwiękowego przepływu wokół profilu NACA0012: izolinie liczby Macha dla trzykrotnie zaadaptowanej siatki

Wyniki eksperymentów numerycznych

Porównanie opisanych wyżej metod dyskretyzacji czasowej zaprezentowane jest dla przykładu stacjonarnego przepływu naddźwiękowego wokół profilu lotniczego, oznaczanego symbolicznie jako NACA0012 [64]. Zastosowano metodę całkowania pseudoczasowego i zamianę globalnego kroku czasowego na lokalne, z wykorzystaniem globalnego parametru CFL i wzoru (2.7). Geometrię opływanego profilu wraz z fragmentem siatki wokół niego przedstawia rys. 2.1. Ostateczne rozwiązanie stacjonarne, do którego zbiegają się wszystkie opisywane metody, zilustrowano za pomocą izolinii liczby Macha na rys. 2.2. Widoczne są fale uderzeniowe rozchodzące się wokół profilu.

Rysunek 2.3 ilustruje zbieżność rozwiązań do stanu stacjonarnego dla poró-



Rys. 2.3. Zbieżność rozwiązań do stanu stacjonarnego (wyrażona za pomocą miary (2.8)) w trakcie symulacji naddźwiękowego przepływu wokół profilu NACA0012 z zastosowaniem różnych metod dyskretyzacji czasowej: Explicit – jawnej, MTG i 2SMTG – niejawnych liniowych, NLE – niejawnej nieliniowej (szczegółowy opis w tekście)

wnywanych metod w zależności od czasu obliczeń. Obliczenia wykonane zostały na procesorze MIPS R10000 (częstotliwość zegara 180 MHz, wydajność – wg miary SPEC [165]: SPECfp_base95 = 14.4, wg miary Linpack [111]: $\approx$ 100 Mflop/s) komputera Origin 200 firmy Silicon Graphics, a mierzony czas to czas wykorzystania procesora (*CPU*). Miarą przybliżania się do stanu stacjonarnego jest wielkość ilorazu:

$$\frac{\mathrm{DIFF}}{\mathrm{CFL}} = \max_{N=1,N_{\mathrm{NOD}}} \frac{|\rho_N^{n+1} - \rho_N^n|}{\mathrm{CFL}} \tag{2.8}$$

Do jego obliczenia wykorzystuje się maksymalną dla wszystkich węzłów siatki MES (N_{NOD} – liczba węzłów w siatce) różnicę gęstości na początku i końcu danego kroku czasowego ($\max_{N=1,N_{\mathrm{NOD}}} |\rho_N^{n+1} - \rho_N^n|$) podzieloną przez wartość parametru CFL [18].

Poszczególne linie na rys. 2.3 odpowiadają następującym metodom: Expli-

cit – metoda jawna, MTG – metoda niejawna zlinearyzowana (*Modified Taylor-Galerkin*), 2SMTG – dwustopniowa metoda niejawna zlinearyzowana, NLE – metoda nieliniowa niejawna (*NonLinear Euler*) . Metoda 2SMTG [18], nie opisywana szczegółowo w niniejszej pracy, sprowadza się do dwukrotnego rozwiązania układu równań liniowych na każdym kroku czasowym, z tą samą macierzą układu, lecz różnymi wektorami prawych stron. Dzięki takiej technice wzrasta stabilność metody (wydłuża się dopuszczalny krok czasowy) przy stosunkowo niewielkich nakładach obliczeniowych.

Spośród omawianych metod, nieliniowa niejawna metoda Eulera i dwustopniowa metoda zlinearyzowana wymagają przechowywania trzech wektorów niewiadomych w trakcie realizacji obliczeń. Pozostałe mają niższą złożoność pamięciową i wymagają przechowywania tylko dwóch wektorów. Dla metod niejawnych koszt przechowywania wektorów niewiadomych jest jednak z reguły znacznie niższy, w granicach rzędu wielkości, od pamięciowego kosztu rozwiązania układu równań liniowych.

Wyniki podobne do przedstawionych na rys. 2.3, ukazujące przewagę metody nieliniowej, uzyskane zostały także dla pozostałych typów przepływów – poddźwiękowego i okołodźwiękowego. Obliczenia dla tego ostatniego przypadku zostaną szerzej omówione w p. 2.3.4, prezentującym przykład rozwiązywania równań nieliniowych.

2.2.5 Podsumowanie

W rozważanym modelu implementacji programów MES dyskretyzacja czasowa nie wchodzi w skład rdzenia obliczeniowego[2]. Jednak architektura tego ostatniego, zwłaszcza jeśli jest projektowana jako ogólnego przeznaczenia, powinna uwzględniać możliwie szeroki zakres sposobów wykorzystania dyskretyzacji przestrzennej MES w rozwiązywaniu zagadnień zależnych od czasu. Przedstawiony w dalszej części pracy model implementacji może zostać zastosowany w realizacji praktycznie dowolnej jednokrokowej metody dyskretyzacji czasowej, także wielostopniowej, jak np. metody Rungego-Kutty. Szczególny nacisk położony jest na implementację technik związanych z dyskretyzacją niejawną, w ramach której generowane są problemy wymagające rozwiązania układów równań nieliniowych i liniowych.

[2]Szczegółowa definicja rdzenia obliczeniowego MES podana jest w rozdz. 4 na s. 166.

2.3 Rozwiązywanie układów równań nieliniowych

Układy równań nieliniowych pojawiają się w efekcie zastosowania dyskretyzacji przestrzennej MES do nieliniowych problemów stacjonarnych lub, w przypadku połączenia z odpowiednimi schematami całkowania czasowego, do nieliniowych problemów niestacjonarnych.

2.3.1 Metoda Newtona i jej warianty

Dominującą techniką rozwiązywania układów równań nieliniowych otrzymanych w wyniku dykretyzacji MES jest metoda Newtona i jej rozliczne wersje (zmodyfikowana metoda Newtona, quasi-metoda Newtona, niedokładna metoda Newtona, przybliżona metoda Newtona). Nieliniowy problem powstały w efekcie dyskretyzacji MES można zapisać w zwartej formie:

$$\mathbf{F}(\mathbf{u}) = 0 \tag{2.9}$$

gdzie $\mathbf{F}$ jest uzyskanym operatorem nieliniowym, a $\mathbf{u}$ jest globalnym wektorem niewiadomych. Metoda Newtona dla równania (2.9) ma postać:

Dla $k = 0, 1, 2, ...$ aż do uzyskania zbieżności, podstawiaj:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \boldsymbol{\Delta}\mathbf{u}_k \tag{2.10}$$

gdzie $\mathbf{u}_0$ jest znanym wektorem początkowym, a $\boldsymbol{\Delta}\mathbf{u}_k$ jest rozwiązaniem równania:

$$\mathbf{J}(\mathbf{u}_k) \cdot \boldsymbol{\Delta}\mathbf{u}_k = -\mathbf{F}(\mathbf{u}_k) \tag{2.11}$$

W klasycznej (ścisłej) metodzie Newtona macierz $\mathbf{J}$, będąca macierzą układu równań liniowych (2.11), jest macierzą jakobianową funkcji $\mathbf{F}$, $\mathbf{J} = \partial\mathbf{F}/\partial\mathbf{u}$. W praktyce obliczenie $\partial\mathbf{F}/\partial\mathbf{u}$ dla złożonych sformułowań skończenie elementowych bywa trudne i kosztowne obliczeniowo lub wręcz niemożliwe. W takim wypadku można w równaniu (2.11) użyć przybliżenia macierzy jakobianowej. Wprawdzie spowalnia to zbieżność procedury iteracyjnej, ale często zapewnia wystarczającą dokładność i efektywność dla całego procesu symulacji. W wariantach metody Newtona $\mathbf{J}$ jest przybliżeniem $\partial\mathbf{F}/\partial\mathbf{u}$, a strategia rozwiązania jest często wzbogacana o techniki przyspieszające zbieżność [70, 71].

2.3.2 Techniki bezmacierzowe

Jedną z najkorzystniejszych, z obliczeniowego punktu widzenia, modyfikacji metody Newtona jest użycie w obliczeniach, zamiast przybliżenia macierzy ja-

kobianowej, przybliżeń jej iloczynów z odpowiednimi wektorami. Taka strategia, będąca podstawą tzw. metod bezmacierzowych (*matrix free*) [94, 102, 14], ma zastosowanie wówczas, gdy do rozwiązania problemów liniowych (2.11) użyte są metody nie korzystające z macierzy układu, a tylko z jej iloczynów ze specjalnymi, tworzonymi w trakcie realizacji algorytmu, wektorami (metody takie mogą powstać np. jako warianty metod podprzestrzeni Kryłowa, omawianych w p. 2.7.5). Metody bezmacierzowe pozwalają uzyskać dobre przybliżenia efektów działania macierzy jakobianowej, a ponadto zapewniają oszczędność pamięci w porównaniu z metodami klasycznymi.

Do przybliżenia iloczynu $\partial\mathbf{F}/\partial\mathbf{u}$ z dowolnym wektorem $\mathbf{v}$ wykorzystuje się wzór na aproksymacje pochodnej kierunkowej:

$$\frac{\partial\mathbf{F}}{\partial\mathbf{u}}(\mathbf{u}_k)\cdot\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{u}_k+\epsilon\mathbf{v})-\mathbf{F}(\mathbf{u}_k)}{\epsilon} \tag{2.12}$$

W powyższym wzorze ϵ jest stałą dobieraną w taki sposób, aby przybliżenie (2.12) było jak najdokładniejsze, a jednocześnie, aby nie wprowadzić istotnych błędów zaokrągleń. W praktyce najczęściej stosuje się wartość ϵ równą pierwiastkowi kwadratowemu dokładności słowa maszynowego odpowiadającego typowi zmiennych, dla których dokonuje się obliczeń (np. 10^{-7} dla liczb podwójnej precyzji).

2.3.3 Ocena złożoności obliczeniowej algorytmów rozwiązywania równań nieliniowych

Złożoność obliczeniowa algorytmów rozwiązywania równań nieliniowych, podobnie jak algorytmów dyskretyzacji czasowej, w istotnym stopniu zależy od liczby i charakteru generowanych problemów liniowych. Złożoność pamięciowa zależy ponownie od liczby przechowywanych wektorów niewiadomych. W przypadku zastosowania metody Newtona do zagadnień stacjonarnych wymagane jest najczęściej przechowywanie dwóch wektorów, odpowiadających poprzedniej i bieżącej iteracji. W przypadku zagadnień niestacjonarnych i metod kontynuacji w pseudoczasie dochodzi jeszcze trzeci wektor – wektor rozwiązania z poprzedniej chwili. Dodatkowo istotne jest, czy ogólna strategia w połączeniu z algorytmem rozwiązywania równań liniowych wymaga tworzenia macierzy układu czy nie. W przypadku technik bezmacierzowych złożoność pamięciowa jest liniową funkcją rozmiaru zadania. W pozostałych przypadkach zależy od złożoności algorytmu rozwiązywania równań liniowych. Tematowi temu poświecony jest p. 2.7.9.

Złożoność czasowa zależy od szybkości zbieżności algorytmu iteracyjnego. Kwadratowa zbieżność metody Newtona prowadzi do najmniejszej liczby problemów liniowych, jednak ich konstrukcja jest kosztowna. Inne metody pozwalają na wykorzystanie tej samej macierzy (lub szeregu macierzy otrzymanych w wyniku mało kosztownych modyfikacji) do ciągu problemów, ale zbieżność tych metod jest wolniejsza [43].

W przypadku zastosowania metod całkowania pseudoczasowego można zamienić niestacjonarne zagadnienie nieliniowe na szereg problemów jawnych, w szczególnych przypadkach nie wymagających rozwiązywania układu równań. Prowadzi to do redukcji złożoności pamięciowej, często jednak kosztem złożoności czasowej – mniejsza dokładność obliczeń z macierzą skupioną może prowadzić do wydłużenia procesu zbiegania się rozwiązania do stanu ustalonego. Niestacjonarne zagadnienie nieliniowe można także zamienić na problemy niejawne i dodatkowo sterować parametrami całkowania czasowego (przede wszystkim długością kroku czasowego) tak, aby kontrolować zbieżność metod iteracyjnych stosowanych do rozwiązywania równań nieliniowych i liniowych.

2.3.4 Porównanie algorytmów rozwiązywania równań nieliniowych dla dyskretyzacji stacjonarnych równań Eulera

W celu ilustracji zastosowania wybranych technik rozwiązywania równań nieliniowych ponownie wykorzystane zostaną równania Eulera dla gazów ściśliwych. Załóżmy, że do aproksymacji równań stacjonarnych użyta została metoda kontynuacji w pseudoczasie, a więc rozwiązywany jest układ niestacjonarny (2.1), z tym że czas pełni rolę parametru formalnego (kroki czasowe Δt_{loc} są obliczane lokalnie dla każdego elementu według wzoru (2.7)). Załóżmy, że zastosowano nieliniową niejawną metodę Eulera, odpowiadającą równaniom (2.5) z parametrami $\alpha = 1$, $\beta = 1$. Porównanie dotyczy dwóch strategii rozwiązywania równań nieliniowych: w jednej wykorzystuje się przybliżenie macierzy jakobianowej, w drugiej stosuje technikę bezmacierzową.

Algorytm z przybliżeniem macierzy jakobianowej

W metodzie tej wykorzystuje się jednorodność strumieni Eulera, jako funkcji wektora zmiennych zachowawczych:

$$\boldsymbol{f}_i^{\mathrm{E}}(\boldsymbol{u}^{n+1}) = \frac{\partial \boldsymbol{f}_i^{\mathrm{E}}}{\partial \boldsymbol{u}}(\boldsymbol{u}^{n+1}) \cdot \boldsymbol{u}^{n+1} \tag{2.13}$$

Podstawienie zależności (2.13) do wzorów (2.5) (z parametrami $\alpha = 1$, $\beta = 1$) prowadzi do sformułowania:

$$\begin{aligned}\int_\Omega \frac{1}{\Delta t_{\text{loc}}} \boldsymbol{w}^T \boldsymbol{u}^{n+1} d\Omega - \int_\Omega \boldsymbol{w}_{,i}^T \frac{\partial \boldsymbol{f}_i^{\text{E}}}{\partial \boldsymbol{u}}(\boldsymbol{u}^{n+1}) \cdot \boldsymbol{u}^{n+1} d\Omega + \\ + \int_\Omega \boldsymbol{w}_{,i}^T \boldsymbol{K}_{ij}(\boldsymbol{u}^{n+1}) \boldsymbol{u}_{,j}^{n+1} d\Omega + \\ + \int_\Gamma \boldsymbol{w}^T \frac{\partial \boldsymbol{f}_i^{\text{E}}}{\partial \boldsymbol{u}}(\boldsymbol{u}^{n+1}) \cdot \boldsymbol{u}^{n+1} n_i d\Gamma = \int_\Omega \frac{1}{\Delta t_{\text{loc}}} \boldsymbol{w}^T \boldsymbol{u}^n d\Omega\end{aligned} \tag{2.14}$$

Proces iteracyjny dla nieliniowego problemu (2.14) wykorzystuje metodę Picarda (iteracji prostej). W rozważanym przypadku polega ona na rozwiązywaniu równania (2.14) dla kolejnych przybliżeń $\boldsymbol{u}_{k+1}^{n+1}$ funkcji $\boldsymbol{u}^{n+1}$, przy wykorzystaniu współczynników macierzowych po lewej stronie (2.14), obliczonych dla przybliżenia $\boldsymbol{u}_k^{n+1}$ uzyskanego w poprzedniej iteracji (przy czym $\boldsymbol{u}_0^{n+1} = \boldsymbol{u}^n$).

Po zastosowaniu dyskretyzacji przestrzennej MES powyższy algorytm iteracji prostej można interpretować jako wersję przybliżonej metody Newtona, z odpowiadającą, szczególną aproksymacją macierzy jakobianowej, oznaczaną dalej przez $\tilde{\mathbf{J}}$. Wadą tej metody jest niska dokładność przybliżenia $\tilde{\mathbf{J}}$, fakt mający istotne znaczenie w przypadku opisywanej dyskretyzacji równań Eulera, w której występują silnie nieliniowe macierzowe funkcje regularyzujące (macierze sztucznej lepkości $\boldsymbol{K}_{ij}$).

Algorytm techniki bezmacierzowej

Drugą porównywaną procedurą jest kombinacja, z zastosowaniem techniki bezmacierzowej, przybliżonej metody Newtona i algorytmu rozwiązywania układów równań liniowych. Przyjęcie ciągłej, kawałkami liniowej dyskretyzacji przestrzennej oraz zastosowanie standardowych procedur MES sprowadza sformułowanie (2.4) do układu algebraicznych równań nieliniowych, dającego się zapisać w postaci równania (2.9) z podstawieniem funkcji $\mathbf{u}^{n+1}$ za niewiadomą funkcję $\mathbf{u}$. Pozostałe kroki algorytmu postępują zgodnie ze schematem opisanym w pp. 2.3.1 i 2.3.2. Jako metodę rozwiązywania układów równań linowych stosuje się metodę GMRES z poprawą uwarunkowania macierzy układu. Techniki poprawy uwarunkowania (nieistniejącej) macierzy układu dla algorytmów bezmacierzowych opisane są w p. 2.7.6.

Tablica 2.1. Szybkość zbieżności metody Newtona (wyrażana miarą (2.15)) w wybranym kroku symulacji okołodźwiękowego przepływu wokół profilu NACA0012 z zastosowaniem niejawnej nieliniowej metody Eulera dla różnych wartości parametru CFL i różnych sposobów aproksymacji macierzy jakobianowej: NLE – opartego na linearyzacji, NKS – wykorzystującego technikę bezmacierzową (szczegółowy opis metod w tekście)

	CFL = 1	CFL = 4	CFL = 256
NLE	0.132	0.418	—
NKS	0.001	0.007	0.173

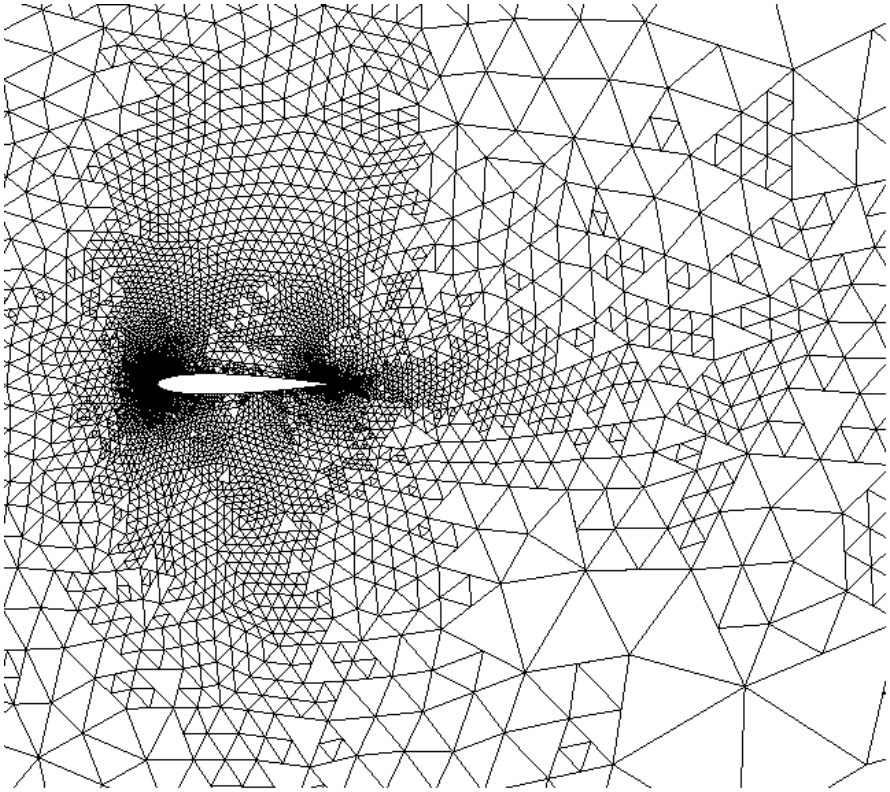
Wyniki eksperymentów numerycznych

Tablica 2.1 pokazuje porównanie dwóch przedstawionych powyżej strategii rozwiązywania równań nieliniowych dla wybranego pojedynczego kroku pseudoczasowego przy symulacji opływu gazu ściśliwego wokół profilu NACA0012. W rozważanym przypadku przepływ jest okołodźwiękowy (liczba Macha 0.85), a kąt natarcia wynosi 1^{o}. Wyniki odpowiadają siatce, której szczegół przedstawiony jest na rys. 2.4. Rysunek 2.5 prezentuje rozwiązanie w postaci izolinii liczby Macha, z widocznymi falami uderzeniowymi rozgraniczającymi obszary naddźwiękowe wokół skrzydeł. W obu przypadkach do rozwiązania układu równań liniowych zastosowano algorytm GMRES z poprawą uwarunkowania macierzy.

Oznaczenia w tablicy są następujące: NLE – metoda iteracji prostej, NKS – metoda bezmacierzowa [48] (skrót pochodzi od nazwisk: Newton–Kryłow––Schwarz, związanych z zastosowanymi metodami; dwie ostatnie są omówione w p. 2.7). Im większa wartość parametru CFL, tym dłuższe lokalne kroki czasowe i bardziej nieliniowy problem (2.9). Liczby podane w tablicy, określające szybkość zbieżności obu metod, są ilorazami normy maksymalnej wektora przyrostów dla niewiadomej funkcji $\mathbf{u}^{n+1}$ po dwóch pierwszych iteracjach metody Newtona:

$$\frac{\|\boldsymbol{\Delta}\mathbf{u}_0^{n+1}\|_{\max}}{\|\boldsymbol{\Delta}\mathbf{u}_0^{n+1}\|_{\max}} \tag{2.15}$$

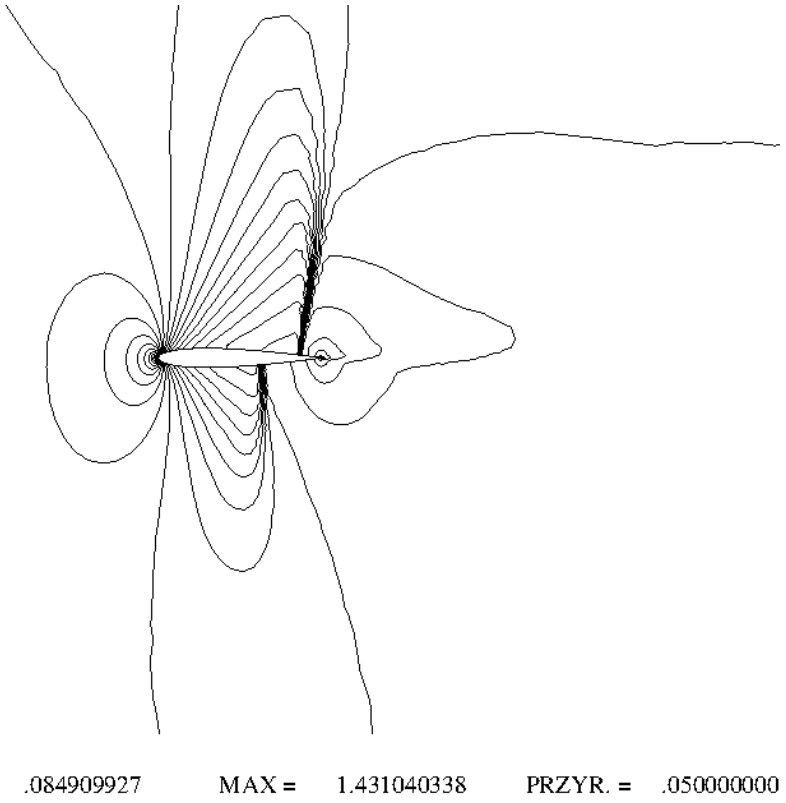
Wyniki pokazują zdecydowaną przewagę metody bezmacierzowej nad metodą iteracji prostej, która przestaje być zbieżna już dla wartości parametru CFL rzędu kilkunastu.



Rys. 2.4. Symulacja okołodźwiękowego przepływu wokół profilu NACA0012: szczegół dwukrotnie zaadaptowanej siatki MES o 5143 węzłach

2.3.5 Podsumowanie

Podobnie jak schematy całkowania czasowego, tak i metody rozwiązywania układów równań nieliniowych nie są traktowane w niniejszej pracy jako część rdzenia obliczeniowego MES. Jednak podobnie jak w przypadku technik całkowania czasowego, rdzeń obliczeniowy MES musi być przystosowany do współpracy z różnymi technikami rozwiązywania układów równań nieliniowych. Dotyczy to konieczności posiadania efektywnego solwera iteracyjnego dla układów równań liniowych, uwzględniającego możliwość stosowania technik bezmacierzowych. Te ostatnie mają istotne znaczenie dla obliczeń wielkiej skali, w przypadku których wymagania pamięciowe związane z przechowywaniem macierzy jakobianowej mogą wielokrotnie przewyższać wymagania pamięciowe pozosta-



Rys. 2.5. Symulacja okołodźwiękowego przepływu wokół profilu NACA0012: izolinie liczby Macha dla dwukrotnie zaadaptowanej siatki

łych części kodu (w tym m.in. struktur danych siatki MES i aproksymacji).

2.4 Sformułowanie słabe metody elementów skończonych

W dwóch poprzednich punktach pokazano, jak różne metody dyskretyzacji czasowej i rozwiązywania równań nieliniowych mogą prowadzić do konieczności rozwiązania pojedynczego liniowego problemu MES. Od tego momentu rozpoczyna się omawianie technik dyskretyzacji przestrzennej metodą elementów skończonych, technik które będą bezpośrednio związane z rdzeniem obliczeniowym programów MES. Niniejszy punkt dotyczy sposobów uzyskiwania

sformułowania słabego dla bardzo ogólnej formy równań konwekcji–dyfuzji––reakcji. Wyprowadzenie przeprowadzone jest dla równań niestacjonarnych. Jego głównym celem jest identyfikacja poszczególnych wyrazów, które pojawią się w ogólnym sformułowaniu pojedynczego problemu MES, będącego podstawą opracowanej implementacji.

2.4.1 Ogólne zagadnienie konwekcji–dyfuzji–reakcji

Równania konwekcji–dyfuzji–reakcji dla wektorowej funkcji $\boldsymbol{u} = [u_1, u_2, ..., u_{N_u}]$ w obszarze obliczeniowym $\Omega \subset I\!R^s$, $s = 2$ lub 3, można zapisać jako[3]:

$$\boldsymbol{M}\frac{\partial \boldsymbol{u}}{\partial t} - \left(\boldsymbol{A}^{ij}\boldsymbol{u}_{,j}\right)_{,i} + \left(\boldsymbol{B}^{i}\boldsymbol{u}\right)_{,i} + \boldsymbol{C}\boldsymbol{u} = \boldsymbol{s} - \boldsymbol{q}^{i}_{,i} \tag{2.16}$$

Macierze $\boldsymbol{M}, \boldsymbol{A}^{ij}, \boldsymbol{B}^{i}, \boldsymbol{C}$ i wektory $\boldsymbol{s}$, $\boldsymbol{q}^{i}$, $i, j = 1, .., s$ są współczynnikami, które mogą być funkcjami nieliniowymi oraz nieciągłymi[4]. Wektory i macierze (kwadratowe) mają wymiar N_u.

Na brzegu obszaru Γ dopuszcza się trzy typy warunków brzegowych:

- warunki Dirichleta (zasadnicze) na brzegu Γ_{D}:

$$\boldsymbol{u} = \boldsymbol{f}^{\mathrm{D}}(\boldsymbol{x}, t) \tag{2.17}$$

- warunki Neumanna (naturalne) na brzegu Γ_{N}:

$$\boldsymbol{A}^{ij}\boldsymbol{u}_{,j}n^{i} = \boldsymbol{f}^{\mathrm{N}}(\boldsymbol{x}, t) \tag{2.18}$$

- warunki Robina (naturalne) na brzegu Γ_{R}:

$$\boldsymbol{A}^{ij}\boldsymbol{u}_{,j}n^{i} = \left(\boldsymbol{u} - \boldsymbol{f}^{\mathrm{R}}(\boldsymbol{x}, t)\right)\boldsymbol{K}^{\mathrm{R}}(\boldsymbol{x}, t) \tag{2.19}$$

[3]Często używana jest także inna notacja dla wektorowego zagadnienia konwekcji–dyfuzji––reakcji wykorzystująca operatory różniczkowe:

$$\sum_{l=1}^{N_u} m_{kl}\frac{\partial u_l}{\partial t} - \boldsymbol{\nabla}\cdot\left(\sum_{l=1}^{N_u}\boldsymbol{A}_{kl}\boldsymbol{\nabla}u_l\right) + \boldsymbol{\nabla}\cdot\left(\sum_{l=1}^{N_u}\boldsymbol{b}_{kl}u_l\right) + \sum_{l=1}^{N_u} c_{kl}u_l = s_k - \boldsymbol{\nabla}\boldsymbol{q}_k$$

z $k = 1, 2, ...N_u$. Obie notacje są równoważne, a wybór (2.16) podyktowany jest jej przydatnością do analiz implementacji MES.

[4]Nie dla wszystkich typów nieliniowości i nieciągłości daje się uzyskać efektywne sformułowania metody elementów skończonych. Zagadnienie to nie jest szczegółowo analizowane w pracy, choć pojawia się np. w kontekście konieczności regularyzacji sformułowania MES dla aproksymacji równań Eulera.

gdzie $\boldsymbol{K}^{\mathrm{R}}$ oznacza zadaną funkcję macierzową, a $\boldsymbol{f}^{\mathrm{D}}$, $\boldsymbol{f}^{\mathrm{N}}$ i $\boldsymbol{f}^{\mathrm{R}}$ są zadanymi funkcjami wektorowymi.

Przy symulacji zjawisk, w których występuje konwekcja, składowe funkcji $\boldsymbol{u}$ odpowiadają fizycznym wielkościom unoszonym przez pole prędkości charakteryzowane przez macierze $\boldsymbol{B}$. Istotny staje się wówczas podział brzegu Γ na część, gdzie następuje wpływ, oznaczaną przez Γ^{-}, oraz na część, gdzie nastepuje wypływ Γ^{+}. Dla uproszczenia przyjęto, że klasyfikacja brzegu dotyczy wszystkich składowych $\boldsymbol{u}$ oraz że na brzegu Γ_{D} nie ma wypływu, a na brzegu Γ_{N} nie ma wpływu. W praktyce (np. przy symulacji przepływów gazu ściśliwego) należy rozważyć bardziej złożone przypadki, które jednak mieszczą się w ogólnym modelu implementacji przedstawionym w następnych rozdziałach.

Warunek początkowy:

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}^{0} \tag{2.20}$$

jest naturalnym uzupełnieniem sformułowania dla zagadnień niestacjonarnych.

2.4.2 Etapy dyskretyzacji przestrzennej MES

Dyskretyzację przestrzenną metodą elementów skończonych przeprowadza się najczęściej w dwóch etapach. W pierwszym uzyskuje się sformułowanie słabe na poziomie nieskończenie wymiarowym, wykorzystując zerowanie się iloczynu skalarnego residuum rozważanego równania z funkcjami testującymi[5]. Istotną rolę, zwłaszcza dla numerycznej analizy zbieżności i błędu, odgrywają przestrzenie funkcyjne, w których znajdują się funkcje niewiadome i testujące [73, 143, 1].

W drugim etapie dokonuje się redukcji zagadnienia do problemu skończenie wymiarowego poprzez ograniczenie rozważanych funkcji niewiadomych i testujących do elementów przestrzeni $\boldsymbol{V}_h$, będącej skończenie wymiarową przestrzenią funkcji określonych dla zadanego podziału obszaru Ω na elementy skończone. Ta ostatnia przestrzeń jest najczęściej, także dla wszystkich rozważań w niniejszej pracy, przestrzenią funkcji będących wielomianami dla pojedynczego elementu. Dla tak zdefiniowanej przestrzeni sformułowanie słabe na poziomie skończenie wymiarowym nazywane będzie sformułowaniem skończenie elementowym.

Ze względu na położenie nacisku na zagadnienia implementacji, przedstawione poniżej wyprowadzenia sformułowań MES operują od razu pojęciem

[5]Sformułowanie słabe jest rozumiane jako pojęcie ogólniejsze od sformułowania wariacyjnego. Obejmuje także przypadki, gdy nie istnieje minimalizowany lub maksymalizowany funkcjonał związany ze sformułowaniem.

przestrzeni funkcyjnej skończenie wymiarowej. Warunki brzegowe naturalne wchodzą w skład sformułowań, natomiast szczegóły implementacji warunków zasadniczych (np. funkcja kary) pozostawione są do implementacji specyficznych problemów (w notacji pominięte są możliwe modyfikacje przestrzeni $\boldsymbol{V}_h$ wprowadzane w celu uwzględnienia warunków brzegowych Dirichleta).

2.4.3 Podział obszaru obliczeniowego na elementy skończone

W celu zdefiniowania dyskretyzacji MES dla określonego problemu, obszar obliczeniowy Ω dzielony jest na elementy skończone Ω_e. Każdy element przedstawia się jako obraz elementu odniesienia $\hat{\Omega}_e$ poprzez odpowiednią wzajemnie jednoznaczną, odwracalną transformację $\boldsymbol{T}_e$:

$$\boldsymbol{T}_e : \hat{\Omega}_e \to \Omega_e \tag{2.21}$$

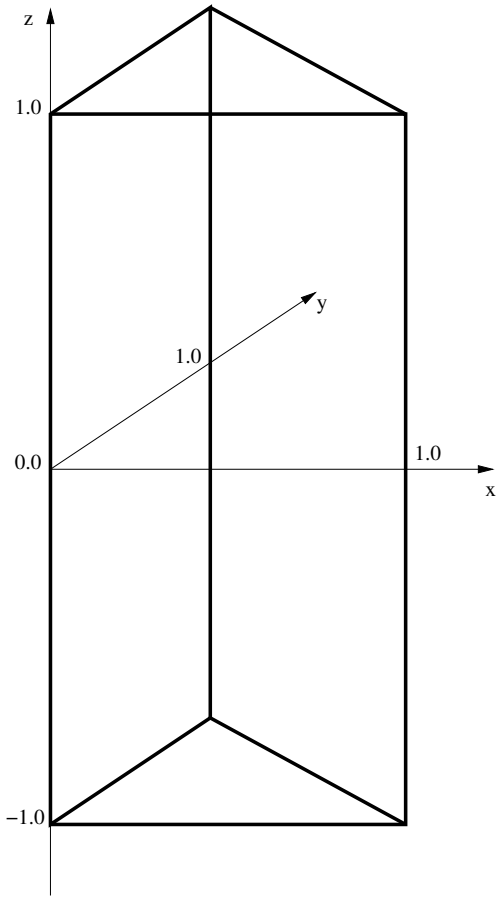
Każdemu punktowi elementu odniesienia, parametryzowanego przez współrzędne $\boldsymbol{\xi}$, przyporządkowuje się punkt elementu rzeczywistego:

$$\boldsymbol{T}_e : \boldsymbol{\xi} \to \boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}) \tag{2.22}$$

Najczęściej elementy siatki są obrazami kilku podstawowych elementów odniesienia: trójkąta, prostokąta, czworościanu, sześcianu, pryzmy (graniastosłupa o podstawie trójkątnej, rys. 2.6) [91, 42]. Elementy jednej siatki mogą być obrazami różnych elementów odniesienia, przy spełnieniu założeń nienakładania się elementów na siebie i sumowania się elementów do całości obszaru obliczeniowego. Transformacje $\boldsymbol{T}_e$, będąc w dużym stopniu arbitralne, muszą jednak pozwalać na efektywne całkowanie funkcji za pomocą kwadratur zdefiniowanych dla elementów odniesienia i różniczkowanie funkcji określonych dla elementów odniesienia (nakłada to pewne ograniczenia na kształt obszaru obliczeniowego dla sformułowania MES, który często jest tylko przybliżeniem rzeczywistego obszaru, w którym zachodzi modelowane zjawisko).

W nieadaptacyjnej MES, a także w pewnych odmianach adaptacyjnej MES, wykorzystywane są wyłącznie siatki regularne, zwane także siatkami zgodnymi (*conforming*). Dla takich siatek wierzchołki elementów mogą znajdować się tylko na końcach krawędzi innych elementów, a każdy bok elementu jest jednocześnie bokiem elementu sąsiadującego. Przykładem takiej siatka jest siatka elementów trójkątnych przedstawiona na rys. 3.5.

Innym typem siatek są siatki nieregularne, zwane także siatkami niezgodnymi (*non-conforming*). Najczęściej powstają one w wyniku zastosowania, dla początkowej siatki regularnej albo uprzednio zaadaptowanej siatki regularnej



Rys. 2.6. Pryzmatyczny element odniesienia

lub nieregularnej, adaptacji typu h, dopuszczającej dzielenie elementów bez modyfikacji ich sąsiadów. Wariantem nieregularności przydatnym do obliczeń dopuszczających zagęszczanie i rozrzedzanie siatki jest jednonieregularność. Polega ona na tym, że nie pozwala się na dalsze podziały elementu, dopóki wszystkie jego boki i krawędzie nie są jednocześnie bokami i krawędziami elementów sąsiadujących. Przykładami siatek jednonieregularnych są siatki pokazane na rys. 2.1 i 2.4.

2.4.4 Wyprowadzenie sformułowania słabego dla przypadków aproksymacji ciągłej i nieciągłej

Poniżej przedstawione są wyprowadzenia sformułowań skończenie elementowych dla dwóch różnych typów aproksymacji: klasycznej ciągłej dyskretyzacji MES i nieciągłej dyskretyzacji Galerkina w wersji zaproponowanej w [125].

Nieciągła aproksymacja Galerkina ma kilka cech istotnie odróżniających ją od podejścia klasycznego (por. zbiór artykułów w [56]). Najważniejszą zaletą opisywanej wersji aproksymacji nieciągłej jest jej zachowawczy charakter (dla zachowawczych procesów ciągłych), lokalnie na poziomie pojedynczego elementu. Sformułowania klasyczne potrafią odtworzyć zachowawczość lokalną tylko za pomocą odpowiednich procedur postprocessingu [92]. Zachowawczość dyskretyzacji jest szczególnie istotna w modelowaniu zagadnień przepływów, zwłaszcza przy uwzględnieniu reakcji chemicznych, kiedy oscylacje rozwiązania prowadzą do zachowań niefizycznych. Cechami negatywnymi prezentowanej wersji dyskretyzacji nieciągłej są: brak stabilności dla stopni aproksymacji niższych od dwóch i nieoptymalna zbieżność dla zagadnień eliptycznych w normie L_2 [152].

W celu uzyskania sformułowania słabego równanie (2.16) jest mnożone przez funkcję testującą $\boldsymbol{w}$, a następnie całkowane indywidualnie po obszarze każdego elementu. Po zastosowaniu wzorów na całkowanie przez części dla obszarów wielowymiarowych, otrzymuje się:

$$\begin{aligned}\int_{\Omega_e}\Big(\boldsymbol{M}\boldsymbol{w}\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{A}^{ij}\boldsymbol{w}_{,i}\boldsymbol{u}_{,j}-\boldsymbol{B}^{i}\boldsymbol{w}_{,i}\boldsymbol{u}+\boldsymbol{C}\boldsymbol{w}\boldsymbol{u}\Big)d\Omega+\\ +\int_{\Gamma_e}\Big(-\boldsymbol{A}^{ij}n_e^i\boldsymbol{w}\boldsymbol{u}_{,j}+\boldsymbol{B}^{i}n_e^i\boldsymbol{w}\boldsymbol{u}\Big)d\Gamma=\\ =\int_{\Omega_e}\Big(\boldsymbol{s}\boldsymbol{w}+\boldsymbol{q}^i\boldsymbol{w}_{,i}\Big)\,d\Omega-\int_{\Gamma_e}\boldsymbol{q}^i n_e^i\boldsymbol{w}d\Gamma\end{aligned} \tag{2.23}$$

gdzie $\boldsymbol{n}_e=[n_e^1,n_e^2,n_e^3]$ jest wektorem jednostkowym, normalnym zewnętrznie względem brzegu elementu Γ_e.

Dla klasycznej ciągłej dyskretyzacji wysumowanie całek elementowych w całym obszarze powoduje zniesienie się wszystkich wyrażeń na brzegu między elementami. Na brzegach obszaru, dla których zadane zostały warunki Neumanna i Robina, funkcje z warunków brzegowych podstawia się w miejsce odpowiednich wyrażeń. Warunki Dirichleta uwzględnia się poprzez modyfikacje sformułowania lub modyfikacje przestrzeni $\boldsymbol{V}_h$[6] [184, 91, 42]. Po kilku

[6]Modyfikacje te nie są analizowane w niniejszej pracy (choć model implementacji uwzględ-

prostych transformacjach uzyskuje się ostateczne sformułowanie MES dla klasycznej dyskretyzacji:

$$\begin{aligned}\int_\Omega \left(\boldsymbol{M}\boldsymbol{w}\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}^{ij}\boldsymbol{w}_{,i}\boldsymbol{u}_{,j} - \boldsymbol{B}^i\boldsymbol{w}_{,i}\boldsymbol{u} + \boldsymbol{C}\boldsymbol{w}\boldsymbol{u} \right) d\Omega + \\ + \int_{\Gamma^+ \cup \Gamma_{\mathrm{R}}^-} \boldsymbol{B}^i n^i \boldsymbol{w}\boldsymbol{u} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}^{\mathrm{R}}\boldsymbol{w}\boldsymbol{u} d\Gamma = \\ = \int_\Omega \boldsymbol{s}\boldsymbol{w} d\Omega + \int_\Omega \boldsymbol{q}^i \boldsymbol{w}_{,i} d\Omega - \int_\Gamma \boldsymbol{q}^i n^i \boldsymbol{w} d\Gamma + \\ + \int_{\Gamma_{\mathrm{N}}} \boldsymbol{w}\boldsymbol{f}^{\mathrm{N}} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}^{\mathrm{R}}\boldsymbol{w}\boldsymbol{f}^{\mathrm{R}} d\Gamma\end{aligned} \tag{2.24}$$

Inaczej jest w przypadku rozwiązań, dla których nie wymaga się ciągłości na brzegu międzyelementowym. Dla każdej pary sąsiadujących elementów Ω_e, Ω_f bok Γ_{ef} jest wspólną częścią ich brzegów, $\Gamma_{ef} = \Gamma_e \cap \Gamma_f$. Całość tak zdefiniowanego brzegu międzyelementowego oznacza się poprzez Γ_{int}, $\Gamma_{\mathrm{int}} = \bigcup \Gamma_{ef}$. Dla każdego brzegu Γ_{ef} wybiera się (na podstawie arbitralnej konwencji) jeden wektor normalny jednostkowy $\boldsymbol{n}$. Na brzegu zewnętrznym definiuje się $\boldsymbol{n} = \boldsymbol{n}_e$. Wprowadza się operatory skoku [ ] i uśrednienia <> dla funkcji określonych na Γ_{ef}:

$$[\boldsymbol{v}] = \boldsymbol{v}^{\mathrm{L}} - \boldsymbol{v}^{\mathrm{R}} = \boldsymbol{v}|_{\Gamma_e \cap \Gamma_{ef}} - \boldsymbol{v}|_{\Gamma_f \cap \Gamma_{ef}} \tag{2.25}$$

$$<\boldsymbol{v}> = 0.5 * (\boldsymbol{v}^{\mathrm{L}} + \boldsymbol{v}^{\mathrm{R}}) \tag{2.26}$$

Wysumowanie całek elementowych i zastosowanie powyższych operatorów daje w wyniku:

$$\begin{aligned}\int_\Omega \left(\boldsymbol{M}\boldsymbol{w}\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}^{ij}\boldsymbol{w}_{,i}\boldsymbol{u}_{,j} - \boldsymbol{B}^i\boldsymbol{w}_{,i}\boldsymbol{u} + \boldsymbol{C}\boldsymbol{w}\boldsymbol{u} \right) d\Omega + \\ + \int_\Gamma \left(-\boldsymbol{A}^{ij} n^i \boldsymbol{w}\boldsymbol{u}_{,j} + \boldsymbol{B}^i n^i \boldsymbol{w}\boldsymbol{u} \right) d\Gamma + \\ + \int_{\Gamma_{\mathrm{int}}} \left(-[\boldsymbol{A}^{ij} n^i \boldsymbol{w}\boldsymbol{u}_{,j}] + [\boldsymbol{B}^i n^i \boldsymbol{w}\boldsymbol{u}] \right) d\Gamma = \\ = \int_\Omega \boldsymbol{s}\boldsymbol{w} d\Omega + \int_\Omega \boldsymbol{q}^i \boldsymbol{w}_{,i} d\Omega - \int_\Gamma \boldsymbol{q}^i n^i \boldsymbol{w} d\Gamma - \int_{\Gamma_{\mathrm{int}}} [\boldsymbol{q}^i n^i \boldsymbol{w}] d\Gamma\end{aligned} \tag{2.27}$$

Dla wyrazów dyfuzji na brzegu Γ_{int} stosuje się podstawienie:

$$[ab] = <a> [b] + [a] <b> \tag{2.28}$$

nia możliwość ich stosowania).

oraz dodaje człon, zerujący się dla ciągłych funkcji $\boldsymbol{u}$:

$$< \boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} > [\boldsymbol{u}] + < \boldsymbol{w} > [\boldsymbol{A}^{ij} n^i \boldsymbol{u}_{,j}] \tag{2.29}$$

Warunki brzegowe Neumanna i Robina stosuje się w analogiczny sposób jak w przypadku ciągłym, natomiast warunki Dirichleta wymusza się w sposób słaby, poprzez dodanie do obu stron równania (2.27) wyrazów:

$$\int_{\Gamma_{\mathrm{D}}} \boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} \boldsymbol{u} d\Gamma \quad \text{do lewej i} \quad \int_{\Gamma_{\mathrm{D}}} \boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} \boldsymbol{f}^{\mathrm{D}} d\Gamma \quad \text{do prawej.}$$

Ze względu na nieciągłość funkcji $\boldsymbol{u}$ i $\boldsymbol{w}$ istnieje możliwość modyfikacji wyrazów konwekcyjnych (pierwszego rzędu) nie zmieniającej rozwiązań ciągłych. W niniejszym wyprowadzeniu przyjmuje się najprostszy, gwarantujący stabilność schemat dyskretyzacji „pod prąd" (*upwind*). Oznacza ona, że na brzegu międzyelementowym używa się w wyrazach konwekcji wartości $\bar{\boldsymbol{u}}$ obliczanych po stronie wpływu. Strumienie $\boldsymbol{q}^i$, które w ogólnym przypadku mogą być zarówno nieliniowe, jak i nieciągłe, są również dyskretyzowane pod prąd. Zastosowanie wymienionych wyżej kroków prowadzi do sformułowania:

$$\begin{aligned}
&\int_{\Omega} \left(\boldsymbol{M}\boldsymbol{w} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{A}^{ij} \boldsymbol{w}_{,i} \boldsymbol{u}_{,j} - \boldsymbol{B}^i \boldsymbol{w}_{,i} \boldsymbol{u} + \boldsymbol{C}\boldsymbol{w}\boldsymbol{u} \right) d\Omega + \\
&+ \int_{\Gamma_{\mathrm{int}}} \left(< \boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} > [\boldsymbol{u}] - [\boldsymbol{w}] < \boldsymbol{A}^{ij} n^i \boldsymbol{u}_{,j} > + [\boldsymbol{B}^i n^i \boldsymbol{w}] \bar{\boldsymbol{u}} \right) d\Gamma + \\
&+ \int_{\Gamma_{\mathrm{D}}} \left(\boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} \boldsymbol{u} - \boldsymbol{A}^{ij} n^i \boldsymbol{w} \boldsymbol{u}_{,j} \right) d\Gamma + \\
&+ \int_{\Gamma^+ \cup \Gamma_{\mathrm{R}}^-} \boldsymbol{B}^i n^i \boldsymbol{w}\boldsymbol{u} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}^{\mathrm{R}} \boldsymbol{w}\boldsymbol{u} d\Gamma = \\
&= \int_{\Omega} \boldsymbol{s}\boldsymbol{w} d\Omega + \int_{\Omega} \boldsymbol{q}^i \boldsymbol{w}_{,i} d\Omega - \int_{\Gamma} \boldsymbol{q}^i n^i \boldsymbol{w} d\Gamma - \int_{\Gamma_{\mathrm{int}}} \bar{\boldsymbol{q}}^i n^i [\boldsymbol{w}] d\Gamma + \\
&+ \int_{\Gamma_{\mathrm{D}}} \boldsymbol{A}^{ij} n^i \boldsymbol{w}_{,j} \boldsymbol{f}^{\mathrm{D}} d\Gamma - \int_{\Gamma_{\mathrm{D}}^-} \boldsymbol{B}^i n^i \boldsymbol{w} \boldsymbol{f}^{\mathrm{D}} d\Gamma + \\
&+ \int_{\Gamma_{\mathrm{N}}} \boldsymbol{w} \boldsymbol{f}^{\mathrm{N}} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}^{\mathrm{R}} \boldsymbol{w} \boldsymbol{f}^{\mathrm{R}} d\Gamma
\end{aligned}$$

2.4.5 Ostateczna postać sformułowania skończenie elementowego dla rozważanych aproksymacji

Ostatecznie do celów implementacji oba wyprowadzone w punkcie poprzednim sformułowania, ciągłe i nieciągłe, zapisane są w klasycznej postaci wykorzystującej formy: dwuliniową i liniową. Pojedynczy problem zapisany jest tak, jakby

do każdego ze sformułowań zastosowano metodę dyskretyzacji czasowej (2.5). Dodatkowo uogólnia się wzory dopuszczając różne współczynniki po lewej i prawej stronie równań. Otrzymane równania obejmują, jako szczególne przypadki, wiele istniejących sformułowań słabych MES dla różnorodnych problemów, w tym wszystkie dotychczas omawiane sformułowania dla dyskretyzacji czasowej i problemów nieliniowych. Sformułowanie ciągłe (oznaczane indeksem MES) otrzymuje formę:

Znajdź taką funkcję $\boldsymbol{u} \in \boldsymbol{V}_h$, aby dla każdej funkcji testującej $\boldsymbol{w} \in \boldsymbol{V}_h$ spełnione było:

$$a_{\mathrm{MES}}(\boldsymbol{u}^{n+1}, \boldsymbol{w}) = l_{\mathrm{MES}}(\boldsymbol{w}) \tag{2.30}$$

gdzie:

$$\begin{aligned} a_{\mathrm{MES}}(\boldsymbol{u}^{n+1}, \boldsymbol{w}) = &\int_\Omega \frac{1}{\Delta t_{\mathrm{loc}}} \boldsymbol{M}_{\mathrm{L}} \boldsymbol{w} \boldsymbol{u}^{n+1} d\Omega + \\ &+ \int_\Omega \left(\boldsymbol{A}_{\mathrm{L}}^{ij} \boldsymbol{w}_{,i} \boldsymbol{u}_{,j}^{n+1} - \boldsymbol{B}_{\mathrm{L}}^{i} \boldsymbol{w}_{,i} \boldsymbol{u}^{n+1} + \boldsymbol{C}_{\mathrm{L}} \boldsymbol{w} \boldsymbol{u}^{n+1} \right) d\Omega + \\ &+ \int_{\Gamma^+ \cup \Gamma_{\mathrm{R}}^-} \boldsymbol{B}_{\mathrm{L}}^{i} n^i \boldsymbol{w} \boldsymbol{u}^{n+1} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}_{\mathrm{L}}^{\mathrm{R}} \boldsymbol{w} \boldsymbol{u}^{n+1} d\Gamma \end{aligned}$$

$$\begin{aligned} l_{\mathrm{MES}}(\boldsymbol{w}) = &\int_\Omega \frac{1}{\Delta t_{\mathrm{loc}}} \boldsymbol{M}_{\mathrm{P}} \boldsymbol{w} \boldsymbol{u}^{n} d\Omega + \\ &+ \int_\Omega \left(-\boldsymbol{A}_{\mathrm{P}}^{ij} \boldsymbol{w}_{,i} \boldsymbol{u}_{,j}^{n} + \boldsymbol{B}_{\mathrm{P}}^{i} \boldsymbol{w}_{,i} \boldsymbol{u}^{n} - \boldsymbol{C}_{\mathrm{P}} \boldsymbol{w} \boldsymbol{u}^{n} \right) d\Omega + \\ &- \int_{\Gamma^+ \cup \Gamma_{\mathrm{R}}^-} \boldsymbol{B}_{\mathrm{P}}^{i} n^i \boldsymbol{w} \boldsymbol{u}^{n} d\Gamma + \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}_{\mathrm{P}}^{\mathrm{R}} \boldsymbol{w} \boldsymbol{u}^{n} d\Gamma + \\ &+ \int_\Omega \boldsymbol{s} \boldsymbol{w} d\Omega + \int_\Omega \boldsymbol{q}^i \boldsymbol{w}_{,i} d\Omega - \int_\Gamma \boldsymbol{q}^i n^i \boldsymbol{w} d\Gamma + \\ &+ \int_{\Gamma_{\mathrm{N}}} \boldsymbol{w} \boldsymbol{f}^{\mathrm{N}} d\Gamma - \int_{\Gamma_{\mathrm{R}}} \boldsymbol{K}^{\mathrm{R}} \boldsymbol{w} \boldsymbol{f}^{\mathrm{R}} d\Gamma \end{aligned}$$

Sformułowanie nieciągłe, oznaczane przez NG, ma postać:

Znajdź taką funkcję $\boldsymbol{u} \in \boldsymbol{V}_h$, aby dla każdej funkcji testującej $\boldsymbol{w} \in \boldsymbol{V}_h$ spełnione było:

$$a_{\mathrm{NG}}(\boldsymbol{u}^{n+1}, \boldsymbol{w}) = l_{\mathrm{NG}}(\boldsymbol{w}) \tag{2.31}$$

gdzie:

$$a_{\mathrm{NG}}(\boldsymbol{u}^{n+1}, \boldsymbol{w}) = \int_\Omega \frac{1}{\Delta t_{\mathrm{loc}}} \boldsymbol{M}_{\mathrm{L}} \boldsymbol{w} \boldsymbol{u}^{n+1} d\Omega +$$

$$
\begin{aligned}
&+\int_{\Omega}\left(\boldsymbol{A}_{\mathrm{L}}^{ij}\boldsymbol{w}_{,i}\boldsymbol{u}_{,j}^{n+1}-\boldsymbol{B}_{\mathrm{L}}^{i}\boldsymbol{w}_{,i}\boldsymbol{u}^{n+1}+\boldsymbol{C}_{\mathrm{L}}\boldsymbol{w}\boldsymbol{u}^{n+1}\right)d\Omega+\\
&+\int_{\Gamma_{\mathrm{int}}}\left(<\boldsymbol{A}_{\mathrm{L}}^{ij}n^{i}\boldsymbol{w}_{,j}>[\boldsymbol{u}^{n+1}]-[\boldsymbol{w}]<\boldsymbol{A}_{\mathrm{L}}^{ij}n^{i}\boldsymbol{u}_{,j}^{n+1}>\right)d\Gamma+\\
&+\int_{\Gamma_{\mathrm{D}}}\left(\boldsymbol{A}_{\mathrm{L}}^{ij}n^{i}\boldsymbol{w}_{,j}\boldsymbol{u}^{n+1}-\boldsymbol{A}_{\mathrm{L}}^{ij}n^{i}\boldsymbol{w}\boldsymbol{u}_{,j}^{n+1}\right)d\Gamma+\\
&+\int_{\Gamma_{\mathrm{int}}}[\boldsymbol{B}_{\mathrm{L}}^{i}n^{i}\boldsymbol{w}]\bar{\boldsymbol{u}}^{n+1}d\Gamma+\int_{\Gamma^{+}\cup\Gamma_{\mathrm{R}}^{-}}\boldsymbol{B}_{\mathrm{L}}^{i}n^{i}\boldsymbol{w}\boldsymbol{u}^{n+1}d\Gamma+\\
&-\int_{\Gamma_{\mathrm{R}}}\boldsymbol{K}_{\mathrm{L}}^{\mathrm{R}}\boldsymbol{w}\boldsymbol{u}^{n+1}d\Gamma
\end{aligned}
$$

$$
\begin{aligned}
l_{\mathrm{NG}}(\boldsymbol{w})=&\int_{\Omega}\frac{1}{\Delta t_{\mathrm{loc}}}\boldsymbol{M}_{\mathrm{P}}\boldsymbol{w}\boldsymbol{u}^{n}d\Omega+\\
&+\int_{\Omega}\left(-\boldsymbol{A}_{\mathrm{P}}^{ij}\boldsymbol{w}_{,i}\boldsymbol{u}_{,j}^{n}+\boldsymbol{B}_{\mathrm{P}}^{i}\boldsymbol{w}_{,i}\boldsymbol{u}^{n}-\boldsymbol{C}_{\mathrm{P}})\boldsymbol{w}\boldsymbol{u}^{n}\right)d\Omega+\\
&+\int_{\Omega}\boldsymbol{s}\boldsymbol{w}d\Omega+\int_{\Omega}\boldsymbol{q}^{i}\boldsymbol{w}_{,i}d\Omega-\int_{\Gamma}\boldsymbol{q}^{i}n^{i}\boldsymbol{w}d\Gamma-\int_{\Gamma_{\mathrm{int}}}\bar{\boldsymbol{q}}^{i}n^{i}[\boldsymbol{w}]d\Gamma+\\
&+\int_{\Gamma_{\mathrm{int}}}\left(<-\boldsymbol{A}_{\mathrm{P}}^{ij}n^{i}\boldsymbol{w}_{,j}>[\boldsymbol{u}^{n}]+[\boldsymbol{w}]<\boldsymbol{A}_{\mathrm{P}}^{ij}n^{i}\boldsymbol{u}_{,j}^{n}>\right)d\Gamma+\\
&+\int_{\Gamma_{\mathrm{D}}}\left(-\boldsymbol{A}_{\mathrm{P}}^{ij}n^{i}\boldsymbol{w}_{,j}\boldsymbol{u}^{n}+\boldsymbol{w}\boldsymbol{A}_{\mathrm{P}}^{ij}n^{i}\boldsymbol{u}_{,j}^{n}\right)d\Gamma+\\
&-\int_{\Gamma_{\mathrm{int}}}[\boldsymbol{B}_{\mathrm{P}}^{i}n^{i}\boldsymbol{w}]\bar{\boldsymbol{u}}^{n}d\Gamma+\int_{\Gamma^{+}\cup\Gamma_{\mathrm{R}}^{-}}-\boldsymbol{B}_{\mathrm{P}}^{i}n^{i}\boldsymbol{w}\boldsymbol{u}^{n}d\Gamma+\\
&+\int_{\Gamma_{\mathrm{D}}}\boldsymbol{A}^{ij}n^{i}\boldsymbol{w}_{,j}\boldsymbol{f}^{\mathrm{D}}d\Gamma-\int_{\Gamma_{\mathrm{D}}^{-}}\boldsymbol{B}^{i}n^{i}\boldsymbol{w}\boldsymbol{f}^{\mathrm{D}}d\Gamma+\int_{\Gamma_{\mathrm{N}}}\boldsymbol{w}\boldsymbol{f}^{\mathrm{N}}d\Gamma+\\
&+\int_{\Gamma_{\mathrm{R}}}\boldsymbol{K}_{\mathrm{P}}^{\mathrm{R}}\boldsymbol{w}\boldsymbol{u}^{n}d\Gamma-\int_{\Gamma_{\mathrm{R}}}\boldsymbol{K}^{\mathrm{R}}\boldsymbol{w}\boldsymbol{f}^{\mathrm{R}}d\Gamma
\end{aligned}
$$

Oba sformułowania są podstawą dla omawianej w niniejszej pracy implementacji rdzenia obliczeniowego MES.

2.5 Dyskretyzacja przestrzenna i struktura układu równań liniowych

Ostatecznym krokiem dyskretyzacji przestrzennej jest określenie skończenie wymiarowej reprezentacji funkcji niewiadomych i testujących. Prowadzi to do przekształcenia równań całkowych tworzących sformułowania słabe w układy algebraicznych równań liniowych. Dotychczasowe rozważania prowadzone były

dla funkcji wektorowych będących elementami przestrzeni skończenie elementowej $\boldsymbol{V}_h$. W dalszej części tego rozdziału zakłada się, że każda ze składowych funkcji aproksymującej i testującej należy do tej samej przestrzeni funkcji skalarnych V_h. Analizy i implementacje dla przestrzeni skalarnych są podstawą dla wielu bardziej rozbudowanych modeli, dotyczących np. problemów wektorowych lub sprzężonych. W pewnych przypadkach uogólnienia mają charakter niebanalny, jak np. w przypadku równań Maxwella dla elektromagnetyzmu [147, 148], wykraczając poza ramy rozważań niniejszej pracy. Kolejne punkty niniejszego podrozdziału przedstawiają definicję przestrzeni V_h i pojęcia związane z dyskretyzacją MES, które wykorzystywane są w rozważaniach dotyczących implementacji komputerowej.

2.5.1 Przestrzenie funkcji kształtu

W celu zdefiniowania przestrzeni skończenie elementowej V_h wykorzystuje się przestrzenie funkcji wielomianowych S_{p_e}, określone dla elementów odniesienia[7]. Indeks p_e odnosi się do stopnia aproksymacji i przyjmuje różną postać w przypadku różnych typów elementów odniesienia. Bazę dla przestrzeni S_{p_e} tworzą wielomiany $\hat{\phi}_l$, zwane funkcjami kształtu. Dla każdej z przestrzeni S_{p_e} (zwanych dalej przestrzeniami funkcji kształtu) można konstruować różne bazy, zależnie od konkretnych uwarunkowań.

Przykładami przestrzeni S_{p_e} typowymi dla różnych rodzajów elementów są [184, 91, 42]:

- przestrzeń zupełnych wielomianów stopnia p na płaszczyźnie – $S_{p_{\xi\eta}}$, dla elementu trójkątnego,
- przestrzeń wielomianów będących iloczynami tensorowymi wielomianów jednowymiarowych – $S_{p_\xi p_\eta}$, dla elementu prostokątnego,
- przestrzeń zupełnych wielomianów stopnia p w przestrzeni – $S_{p_{\xi\eta\zeta}}$, dla elementu czworościennego,
- przestrzeń wielomianów będących iloczynami tensorowymi wielomianów z przestrzeni $S_{p_{\xi\eta}}$ i wielomianów jednowymiarowych z przestrzeni S_{p_ζ} – $S_{p_{\xi\eta}p_\zeta}$, dla elementu pryzmatycznego,
- przestrzeń wielomianów będących iloczynami tensorowymi wielomianów jednowymiarowych – $S_{p_\xi p_\eta p_\zeta}$, dla elementu sześciennego.

[7] Geometrycznie identyczne elementy odniesienia z *różnymi* funkcjami kształtu traktowane są jako *różne*.

Tablica 2.2. Wymiary typowych przestrzeni funkcji kształtu dla podstawowych elementów odniesienia

Typ elementu	Oznaczenie przestrzeni	Wymiar przestrzeni
trójkątny	$S_{p_{\xi\eta}}$	$\frac{1}{2}(p+1)(p+2)$
prostokątny	$S_{p_\xi p_\eta}$	$(p+1)^2$
czworościenny	$S_{p_{\xi\eta\zeta}}$	$\frac{1}{6}(p+1)(p+2)(p+3)$
pryzmatyczny	$S_{p_{\xi\eta}p_\zeta}$	$\frac{1}{2}(p+1)^2(p+2)$
sześcienny	$S_{p_\xi p_\eta p_\zeta}$	$(p+1)^3$

Tablica 2.2 przedstawia dla każdej z powyższych przestrzeni jej wymiar (liczbę funkcji kształtu) w zależności od stopnia aproksymacji p, w przypadku aproksymacji izotropowej. Przestrzenie wielomianów będących iloczynami tensorowymi dopuszczają aproksymację anizotropową (różne stopnie wielomianów dla różnych kierunków przestrzennych), w przypadku której implementacja i analiza złożoności obliczeniowej stanowią prostą modyfikację przypadku izotropowego (inaczej jest z analizą błędu aproksymacji [144]).

2.5.2 Przestrzenie skończenie elementowe

Przestrzeń skończenie elementową V_h definiuje się jako przestrzeń takich funkcji u, które ograniczone do pojedynczego elementu Ω_e, są odwzorowaniami funkcji z przestrzeni S_{p_e} za pomocą transformacji $\boldsymbol{T}_e$ (2.21), przekształcającej element odniesienia $\hat{\Omega}_e$ w elementy siatki MES[8]:

$$V_h = \{u : u|_{\Omega_e} \circ \boldsymbol{T}_e \in S_{p_e}\} \tag{2.32}$$

$$u(\boldsymbol{x})|_{\Omega_e} = u(\boldsymbol{x}(\boldsymbol{\xi})) = \hat{u}(\boldsymbol{\xi}) \tag{2.33}$$

gdzie: $\boldsymbol{\xi} \in \hat{\Omega}_e$ i $\hat{u}(\boldsymbol{\xi}) \in S_{p_e}$. Bazę dla przestrzeni V_h tworzą funkcje ϕ_l, zwane dalej funkcjami bazowymi MES (lub w skrócie po prostu funkcjami bazowymi), które ograniczone do pojedynczego elementu są przeniesionymi

[8] Jeśli elementy siatki MES odpowiadają różnym elementom odniesienia, spełniony musi zostać warunek, aby ostatecznie skonstruowana przestrzeń V_h spełniała wymagania ciągłości stawiane konkretnemu typowi aproksymacji.

funkcjami kształtu[9]:

$$\phi_l|_{\Omega_e} = \hat{\phi}_l \circ \boldsymbol{T}_e^{-1} \tag{2.34}$$

$$\phi_l(\boldsymbol{x})|_{\Omega_e} = \phi_l(\boldsymbol{x}(\boldsymbol{\xi})) = \hat{\phi}_l(\boldsymbol{\xi}) \quad \text{dla} \quad \boldsymbol{x} \in \Omega_e \tag{2.35}$$

Składania funkcji kształtu w funkcje bazowe MES dokonuje się w taki sposób, aby zapewnić realizację wymagań dotyczących ciągłości aproksymacji i jednocześnie uzyskać funkcje bazowe różne od zera na jak najmniejszym podobszarze obszaru obliczeniowego. To ostatnie wymaganie decyduje o efektywności obliczeń MES (patrz p. 2.5.5).

2.5.3 Powiązanie funkcji bazowych MES z obiektami siatki i obszary zerowania się funkcji bazowych

Dla uściślenia dalszych analiz wprowadza się następującą definicję:

Definicja 2.1 Obiektem siatki *nazywany jest dowolny wierzchołek elementu, krawędź elementu, bok elementu lub sam element.*

Krawędzie i boki elementu mogą być zakrzywione. W przypadku dwuwymiarowym pojęcia bok i krawędź elementu utożsamiają się. W celu zdefiniowania funkcji bazowych MES element utożsamia się z jego wnętrzem.

W rozważaniach niniejszej pracy zakłada się, że każda funkcja bazowa MES związana jest z pojedynczym obiektem siatki. Przyjmuje się, że funkcja bazowa związana z danym obiektem siatki jest niezerowa na całym obszarze tego obiektu (dla wierzchołków oznacza to niezerowanie się funkcji bazowej w punkcie, dla elementów niezerowanie się wewnątrz elementu).

W klasycznych ciągłych liniowych sformułowaniach MES funkcje bazowe wiąże się z węzłami siatki MES, będącymi wierzchołkami elementów. Funkcja bazowa związana z danym wierzchołkiem jest różna od zera w tym wierzchołku i równa zeru we wszystkich pozostałych wierzchołkach[10].

Dla ciągłej aproksymacji wyższego stopnia wygodne jest definiowanie funkcji bazowych dla wszystkich obiektów siatki MES [60, 24]. Obok klasycznych funkcji bazowych związanych z wierzchołkami, wprowadza się funkcje bazowe

[9] W adaptacyjnej MES typu *hp* dopuszcza się występowanie siatek nieregularnych i zezwala na sąsiadowanie z sobą elementów o różnych stopniach aproksymacji. W tym przypadku definicje funkcji bazowych MES są bardziej złożone [60, 24].

[10] Powiązanie wierzchołka siatki MES z funkcją bazową prowadzi do klasycznego pojęcia węzła siatki MES. Ze względu na przyjęcie modelu aproksymacji, w którym funkcje bazowe mogą być związane z dowolnymi obiektami siatki, pojęcie węzła siatki MES nie jest używane w niniejszej pracy.

związane z pojedynczą krawędzią, które zerują się w wierzchołkach i wzdłuż innych krawędzi, funkcje związane z konkretnym bokiem, które zerują się wzdłuż krawędzi i na innych bokach, oraz funkcje związane z wnętrzem elementu, które zerują się na bokach. Spełniony jest wtedy postulat, aby każda funkcja bazowa była niezerowa na obszarze odpowiadającego jej obiektu siatki, a ponadto realizowane podstawowe dla MES założenie niezerowania się funkcji bazowych MES na możliwie najmniejszym podobszarze obszaru obliczeniowego. Tak więc funkcje bazowe związane z wnętrzem elementu są niezerowe tylko w jednym elemencie, związane z bokiem w dwóch, pozostałe zaś mogą być niezerowe w kilku, w przypadku siatek niestrukturalnych najwyżej w kilkunastu elementach.

W aproksymacji nieciągłej istnieje większa swoboda definiowania funkcji bazowych. Jednak dla ograniczenia obszaru ich niezerowania się najczęściej przyjmuje się, że funkcjami bazowymi są elementowe funkcje kształtu. W konsekwencji, każda z funkcji bazowych jest związana z pojedynczym elementem, we wnętrzu którego i na bokach którego jest niezerowa, zeruje się natomiast we wnętrzach wszystkich innych elementów.

2.5.4 Przekształcenie sformułowania słabego w układ równań liniowych

Przy rozważanej w niniejszej pracy dyskretyzacji każdą składową niewiadomej funkcji $\boldsymbol{u}^{n+1}$ i funkcji testującej $\boldsymbol{w}$ przedstawia się jako kombinację liniową N' funkcji bazowych przestrzeni V_h:

$$\boldsymbol{u}^{n+1}(\boldsymbol{x}) = \sum_{l=1}^{N'} \boldsymbol{u}_l^{n+1} \phi_l(\boldsymbol{x}) \tag{2.36}$$

$$\boldsymbol{w}(\boldsymbol{x}) = \sum_{m=1}^{N'} \boldsymbol{w}_m \phi_m(\boldsymbol{x}) \tag{2.37}$$

Podstawienie powyższych zależności do sformułowań (2.30) i (2.31) prowadzi, po standardowych dla metody elementów skończonych transformacjach [184, 91, 42], do przekształcenia równań całkowych w równania algebraiczne, dające się zapisać w postaci:

$$\sum_{m=1}^{N'} \left(\sum_{l=1}^{N'} \boldsymbol{A}_{ml} \boldsymbol{u}_l^{n+1} - \boldsymbol{b}_m \right) \boldsymbol{w}_m = 0 \qquad \forall \boldsymbol{w} \in \boldsymbol{V}_h \tag{2.38}$$

Różnica pomiędzy sformułowaniami przejawia się w różnej postaci macierzy $\boldsymbol{A}_{ml}$ i wektorów $\boldsymbol{b}_m$. Ze względu na dowolność funkcji $\boldsymbol{w}$ równanie (2.38) jest równoważne układowi równań liniowych:

$$\sum_{l=1}^{N'} \boldsymbol{A}_{ml} \boldsymbol{u}_l^{n+1} = \boldsymbol{b}_m \qquad m = 1, ..., N' \tag{2.39}$$

W powyższym wzorze $\boldsymbol{u}_l^{n+1}$ oznacza wektor stopni swobody o wymiarze N_u związany z pojedynczą funkcją bazową ϕ_l. Podobnie $\boldsymbol{b}_m$ jest wektorem, a $\boldsymbol{A}_{ml}$ macierzą kwadratową, każde o wymiarze N_u.

W celu uproszczenia rozważań w dalszej części pracy notacja układu równań przedstawiana jest w zmodyfikowanej postaci. Globalny wektor stopni swobody, złożony z wszystkich wektorów $\boldsymbol{u}_l^{n+1}$, jest oznaczany przez $\mathbf{u}$. Liczba wszystkich jego składowych, N, określa liczbę niewiadomych w układzie (liczbę stopni swobody rozwiązania dyskretnego) i stanowi najważniejszy parametr determinujący złożoność obliczeniową algorytmów rdzenia obliczeniowego MES. Funkcje bazowe są przenumerowane tak, że pojedynczej składowej u_l wektora $\mathbf{u}$ odpowiada pojedyncza funkcja bazowa ϕ_l[11]. Zmodyfikowany układ przyjmuje postać:

$$\sum_{l=1}^{N} A_{ml} u_l = b_m \qquad m = 1, ..., N \tag{2.40}$$

lub w notacji macierzowej:

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{2.41}$$

Pojedynczy wyraz macierzy $\mathbf{A}$ oblicza się z wzoru:

$$A_{ml} = a_{\mathrm{MES}}(\phi_m, \phi_l) \quad \text{lub} \quad A_{ml} = a_{\mathrm{NG}}(\phi_m, \phi_l) \tag{2.42}$$

a pojedynczy wyraz wektora prawej strony $\mathbf{b}$ dany jest jako:

$$b_m = l_{\mathrm{MES}}(\phi_m) \quad \text{lub} \quad b_m = l_{\mathrm{NG}}(\phi_m) \tag{2.43}$$

Każde równanie w układzie (2.41) i każda składowa wektora $\mathbf{b}$ odpowiadają jednej funkcji bazowej ϕ_m. Każda kolumna macierzy $\mathbf{A}$ odpowiada pojedynczej funkcji bazowej ϕ_l, a każdy wyraz A_{ml} w macierzy $\mathbf{A}$ odpowiada parze funkcji ϕ_m i ϕ_l.

[11] Dla analiz w niniejszej pracy przyjęte jest założenie, że różnym składowym wektorowych funkcji niewiadomych i testujących odpowiadają identyczne funkcje bazowe. Model implementacji dopuszcza jednak istnienie różnych funkcji bazowych i różnych przestrzeni funkcji aproksymujących dla różnych składowych, a co za tym idzie, obejmuje także takie wersje MES, jak metody mieszane czy hybrydowe.

2.5.5 Bloki elementarne i struktura układu równań liniowych MES

Zerowanie się funkcji bazowych MES w obszarze obliczeniowym wszędzie, poza bardzo małymi podobszarami, prowadzi, na mocy mechanizmu tworzenia układu równań, do charakterystycznego dla metody elementów skończonych zerowania się większości wyrazów macierzy $\mathbf{A}$. Dzięki temu niska jest złożość całkowania numerycznego i rozwiązywania układów równań liniowych. Dla ścisłego określenia struktury układu równań liniowych wprowadza się następującą definicję:

Definicja 2.2 Podobszarem całkowania *nazywany jest pojedynczy element lub bok elementu, po obszarze którego wykonuje się całkowanie dla co najmniej jednego wyrazu ze sformułowania słabego.*

Pojedynczy wyraz A_{ml}, obliczany z wzorów (2.42), jest niezerowy wtedy, gdy odpowiadające mu funkcje bazowe ϕ_m i ϕ_l są jednocześnie różne od zera przynajmniej w jednym podobszarze całkowania.

Jeśli rozważana aproksymacja jest aproksymacją wyższego stopnia (kiedy z pojedynczym obiektem siatki może być związane kilka funkcji bazowych) lub aproksymacją problemu wektorowego, funkcje bazowe w sposób naturalny łączą się w grupy. Takie grupowanie się funkcji bazowych indukuje podział globalnego wektora niewiadomych $\mathbf{u}$ (jako wektora współczynników kombinacji liniowej funkcji bazowych) na mniejsze podwektory. Wprowadza się następującą definicję:

Definicja 2.3 Podwektorem elementarnym $\mathbf{u}_k^{\mathrm{E}}$ *wektora niewiadomych* $\mathbf{u}$ *nazywany jest zbiór stopni swobody odpowiadający funkcjom bazowym związanym z pojedynczym obiektem siatki.*

Na mocy założeń dotyczących aproksymacji MES i powiązania funkcji bazowych z obiektami siatki, podwektory $\mathbf{u}_k^{\mathrm{E}}$ są rozłączne i sumują się do całego wektora niewiadomych $\mathbf{u}$. Podobne podziały wprowadza się dla wektora prawej strony $\mathbf{b}$ i macierzy układu $\mathbf{A}$, która uzyskuje postać blokową zgodnie z wzorem:

$$\sum_{k=1}^{N_{\mathrm{ble}}} \mathbf{A}_{jk}^{\mathrm{E}} \mathbf{u}_k^{\mathrm{E}} = \mathbf{b}_j^{\mathrm{E}} \qquad j = 1, ..., N_{\mathrm{ble}} \tag{2.44}$$

gdzie N_{ble} oznacza liczbę wszystkich podwektorów elementarnych $\mathbf{u}_k^{\mathrm{E}}$.

Definicja 2.4 Blokami elementarnymi $\mathbf{A}^{\mathrm{E}}_{jk}$ *macierzy układu* **A** *nazywane są bloki określone za pomocą wzoru (2.44)*[12].

Pojedynczy blok elementarny macierzy **A** odpowiada parze obiektów siatki. Wyjątkiem są bloki na przekątnej głównej, zwane blokami diagonalnymi, które odpowiadają pojedynczemu obiektowi. Blok jest niezerowy, jeśli odpowiadające mu obiekty siatki znajdują się blisko siebie. Bliskość jest definiowana w kontekście zerowania się funkcji bazowych związanych z obiektami siatki. Dwa obiekty są blisko siebie, jeśli funkcje bazowe związane z nimi są jednocześnie różne od zera w co najmniej jednym podobszarze całkowania[13].

Struktura macierzy **A** zilustrowana jest na rys. 2.7 dla przykładu skalarnej aproksymacji trzeciego stopnia w elementach czworokątnych. Wypisane są tylko niezerowe wyrazy macierzy: e_1 oznacza pewien czworokąt (ściśle jego wnętrze), w_1 jeden z jego wierzchołków, a b_1 jeden z jego boków; b_2 oznacza krawędź należącą do elementu sąsiedniego e_2, nie należącą do e_1. Z obiektami siatki związane są grupy funkcji bazowych, a tym z kolei odpowiadają zaznaczone na rys. 2.7 grupy wierszy i kolumn **A**. Poszczególne bloki elementarne opisane są poniżej dla kolejnych grup wierszy macierzy **A** związanych z kolejnymi obiektami siatki. Taka grupa wierszy, poziome pasmo macierzy **A**, będzie miało istotne znaczenie przy implementacji solwera równań liniowych.

Blokami elementarnymi na rys. 2.7 są:

- dla wierzchołka w_1 – blok diagonalny $\{A_{1,1}\}$ i bloki pozadiagonalne: $\left\{ \begin{array}{cc} A_{1,2} & A_{1,3} \end{array} \right\}$, $\left\{ \begin{array}{cccc} A_{1,4} & A_{1,5} & A_{1,6} & A_{1,7} \end{array} \right\}$ i $\left\{ \begin{array}{cc} A_{1,8} & A_{1,9} \end{array} \right\}$,

- dla krawędzi b_1 – blok diagonalny $\left\{ \begin{array}{cc} A_{2,2} & A_{2,3} \\ A_{3,2} & A_{3,3} \end{array} \right\}$ i bloki pozadiagonalne: $\left\{ \begin{array}{c} A_{2,1} \\ A_{3,1} \end{array} \right\}$, $\left\{ \begin{array}{cccc} A_{2,4} & A_{2,5} & A_{2,6} & A_{2,7} \\ A_{3,4} & A_{3,5} & A_{3,6} & A_{3,7} \end{array} \right\}$ i $\left\{ \begin{array}{cc} A_{2,8} & A_{2,9} \\ A_{3,8} & A_{3,9} \end{array} \right\}$,

[12] W przypadku liniowej aproksymacji funkcji wektorowych, takiej jak np. opisywana w pracy aproksymacja równań Eulera czy klasyczna aproksymacja równań liniowej teorii sprężystości, bloki elementarne $\mathbf{A}^{\mathrm{E}}_{jk}$ bezpośrednio odpowiadają macierzom występującym we wzorach (2.38) i (2.39).

[13] Przy aproksymacji związanej [60] na siatkach nieregularnych nie ze wszystkimi obiektami siatki można związać funkcje bazowe. Jednocześnie niektóre funkcje bazowe mogą być niezerowe w elementach nie zawierających obiektu, z którym związana jest dana funkcja. Mimo to ogólna struktura macierzy **A** pozostaje taka sama – niezerowe są te bloki, które odpowiadają obiektom siatki bliskim sobie, komplikuje się nieco tylko definicja bliskości.

$$
\begin{array}{cccccccccccc}
 & & w_1 & & b_1 & & & e_1 & & & b_2 & \\
 & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\
w_1 & ... & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} & A_{1,7} & A_{1,8} & A_{1,9} & ... \\
b_1 & ... & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & A_{2,5} & A_{2,6} & A_{2,7} & A_{2,8} & A_{2,9} & ... \\
b_1 & ... & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} & A_{3,5} & A_{3,6} & A_{3,7} & A_{3,8} & A_{3,9} & ... \\
e_1 & ... & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} & A_{4,5} & A_{4,6} & A_{4,7} & & & ... \\
e_1 & ... & A_{5,1} & A_{5,2} & A_{5,3} & A_{5,4} & A_{5,5} & A_{5,6} & A_{5,7} & & & ... \\
e_1 & ... & A_{6,1} & A_{6,2} & A_{6,3} & A_{6,4} & A_{6,5} & A_{6,6} & A_{6,7} & & & ... \\
e_1 & ... & A_{7,1} & A_{7,2} & A_{7,3} & A_{7,4} & A_{7,5} & A_{7,6} & A_{7,7} & & & ... \\
b_2 & ... & A_{8,1} & A_{8,2} & A_{8,3} & & & & & A_{8,8} & A_{8,9} & ... \\
b_2 & ... & A_{9,1} & A_{9,2} & A_{9,3} & & & & & A_{9,8} & A_{9,9} & ...
\end{array}
\tag{2.45}
$$

Rys. 2.7. Przykładowa struktura niezerowych wyrazów macierzy **A** układu równań liniowych MES

- dla elementu e_1 – blok diagonalny $\left\{ \begin{array}{cccc} A_{4,4} & A_{4,5} & A_{4,6} & A_{4,7} \\ A_{5,4} & A_{5,5} & A_{5,6} & A_{5,7} \\ A_{6,4} & A_{6,5} & A_{6,6} & A_{6,7} \\ A_{7,4} & A_{7,5} & A_{7,6} & A_{7,7} \end{array} \right\}$ i bloki pozadiagonalne: $\left\{ \begin{array}{c} A_{4,1} \\ A_{5,1} \\ A_{6,1} \\ A_{7,1} \end{array} \right\}$ i $\left\{ \begin{array}{cc} A_{4,2} & A_{4,3} \\ A_{5,2} & A_{5,3} \\ A_{6,2} & A_{6,3} \\ A_{7,2} & A_{7,3} \end{array} \right\}$,
- dla krawędzi b_2 – blok diagonalny $\left\{ \begin{array}{cc} A_{8,8} & A_{8,9} \\ A_{9,8} & A_{9,9} \end{array} \right\}$ i bloki pozadiagonalne: $\left\{ \begin{array}{c} A_{8,1} \\ A_{9,1} \end{array} \right\}$ i $\left\{ \begin{array}{cc} A_{8,2} & A_{8,3} \\ A_{9,2} & A_{9,3} \end{array} \right\}$.

Bloki diagonalne są zawsze kwadratowe, bloki pozadiagonalne mogą być prostokątne. Schemat tworzenia bloków elementarnych macierzy **A** można w sposób prosty powtórzyć dla większych bloków związanych z grupami obiektów siatki. Łatwo wyobrazić sobie bloki związane ze wszystkimi obiektami tworzącymi element e_1. Znów powstałby jeden kwadratowy blok diagonalny (jego fragmentem byłby kwadrat $A_{1,1} \div A_{1,7} \times A_{1,1} \div A_{1,7}$) oraz bloki pozadiagonalne związane z obiektami sąsiadującymi z e_1. Fragmentem jednego z takich bloków, odpowiadającemu parze elementów e_1 i e_2, byłby prostokąt

$A_{1,8} \div A_{1,9} \times A_{1,1} \div A_{1,7}$. Idąc dalej, można tworzyć bloki związane z małymi grupami sąsiadujących elementów („łatami" elementów) i jeszcze dalej, bloki związane z dużymi podobszarami obszaru obliczeniowego.

Istotną cechą bloków elementarnych jest to, że dla standardowych aproksymacji wszystkie ich wyrazy są niezerowe. Dla większych bloków nie jest to prawdą. Już bloki związane z pojedynczym elementem zawierają zerowe fragmenty macierzy **A**. Na rys. 2.7 widać, że prostokątny blok pozadiagonalny związany z e_1 i e_2 będzie miał zera we wszystkich wierszach odpowiadających wnętrzu e_1 (wyrazy na przecięciu wierszy od 4 do 7 i kolumn od 8 do 9). Przy tworzeniu jeszcze większych bloków zera pojawiają się także w blokach diagonalnych. Dla bloku diagonalnego związanego z parą elementów e_1 i e_2 wymienione wyżej zera z bloku pozadiagonalnego przenoszą się do bloku diagonalnego. W sytuacji gdy bloki odpowiadają dużym podobszarom, ich struktura jest zbliżona do struktury samej macierzy **A**.

Rzadka struktura macierzy **A** jest wykorzystywana przez wszystkie solwery układów równań liniowych używane w programach MES. Proste rozumowanie dla strukturalnej siatki sześciennej i ciągłej liniowej skalarnej aproksymacji pokazuje, że liczba niezerowych wyrazów w pojedynczym wierszu **A** jest stała i wynosi 27. Przy liczbie niewiadomych rzędu milionów daje to współczynnik rzadkości (procent zer w macierzy) ponad 99.99%. Dla dowolnej siatki i ograniczonego stopnia aproksymacji (zazwyczaj w praktyce stopień aproksymacji nie przekracza dwudziestu) liczba niezerowych wyrazów w pojedynczym wierszu **A** jest ograniczona. W obliczeniach wielkiej skali, gdy liczba stopni swobody wynosi kilka milionów, współczynnik rzadkości przekracza 99%, nawet w przypadku aproksymacji wyższego stopnia i siatek niestrukturalnych.

2.6 Całkowanie numeryczne

2.6.1 Całki sformułowania słabego

Istotnym czynnikiem koniecznym do uwzględnienia przy implementacji programów MES jest sposób obliczania poszczególnych wyrazów macierzy **A** i wektora prawej strony **b** . Na mocy wzorów (2.42) i (2.43) każdy taki wyraz jest sumą całek ze sformułowania słabego. Każda całka jest sumą całek lokalnych, pochodzących od poszczególnych podobszarów całkowania. W każdej całce lokalnej (całce elementowej lub całce po boku elementu) występują funkcje bazowe (bezpośrednio albo po zróżniczkowaniu), odpowiadające danemu wyrazowi macierzy **A** lub wektora **b**. Pojedyncza całka elementowa charakteryzowana jest przez następujące parametry:

- podobszar całkowania i związany z nim element całkowania (objętościowy, powierzchniowy lub liniowy),
- współczynnik będący składową odpowiedniej funkcji ($\boldsymbol{M}$, $\boldsymbol{A}^{ij}$, $\boldsymbol{B}^{i}$, $\boldsymbol{C}$, $\boldsymbol{s}$ lub $\boldsymbol{q}^{i}$) występującej w aproksymowanym zagadnieniu,
- ewentualnie, rozwiązanie i/lub jego pochodna w poprzedniej chwili lub iteracji,
- iloczyn funkcji bazowej (lub jej pochodnej) odpowiadającej funkcji niewiadomej z funkcją bazową (lub jej pochodną) odpowiadającą funkcji testującej.

W całkach odpowiadających wektorowi prawej strony w miejscu pary funkcji bazowych występuje pojedyncza funkcja bazowa odpowiadająca funkcji testującej.

Wzory analityczne związane z procedurami całkowania przedstawione zostaną poniżej na przykładzie całki elementowej, odpowiadającej wyrazowi konwekcyjnemu ze sformułowania MES dla obszaru trójwymiarowego, zawierającej wszystkie typy funkcji pojawiające się w sformułowaniach (2.30) i (2.31):

$$\mathcal{C} = \int_{\Omega_e} b^1(\boldsymbol{x}, \boldsymbol{u}^n, \nabla \boldsymbol{u}^n) \frac{\partial \phi_m}{\partial x_1} \phi_l d\Omega \tag{2.46}$$

Całkowanie pozostałych wyrazów ze sformułowań (2.30) i (2.31) odbywa się według procedur uzyskanych przez proste modyfikacje technik opisanych dla (2.46).

2.6.2 Kwadratury całkowania numerycznego

Podstawową techniką całkowania w programach MES jest całkowanie numeryczne oparte na kwadraturach zdefiniowanych dla elementów odniesienia. Pierwszym krokiem całkowania jest zmiana zmiennych i przeniesienie obszaru całkowania do elementu odniesienia:

$$\mathcal{C} = \int_{\hat{\Omega}_e} b^1(\boldsymbol{x}(\boldsymbol{\xi}), \boldsymbol{u}^n(\boldsymbol{\xi}), \nabla_{\boldsymbol{x}} \boldsymbol{u}^n(\boldsymbol{\xi})) \frac{\partial \hat{\phi}_m(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial x_1} \hat{\phi}_l(\boldsymbol{\xi}) \det \boldsymbol{J}_{\boldsymbol{T}_e} d\Omega \tag{2.47}$$

gdzie $\boldsymbol{J}_{\boldsymbol{T}_e}$ oznacza macierz jakobianową transformacji $\boldsymbol{T}_e$, a $\frac{\partial \boldsymbol{\xi}}{\partial x_i}$ jest kolumną macierzy jakobianowej $\boldsymbol{J}_{\boldsymbol{T}_e^{-1}}$, $\boldsymbol{J}_{\boldsymbol{T}_e^{-1}} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}$, która odpowiada transformacji odwrotnej $\boldsymbol{T}_e^{-1}$.

Znajdujący się w całce (2.46) współczynnik funkcyjny ma maksymalny stopień złożoności – jest zależny od położenia punktu oraz wartości niewiadomej funkcji i jej gradientu, dla poprzedniej chwili lub iteracji. Zazwyczaj złożoność zależności jest mniejsza, w wielu typowych zagadnieniach (np. liniowa teoria sprężystości) współczynniki są stałe w całym obszarze elementu.

Istnieje wiele kwadratur zaprojektowanych dla poszczególnych typów elementów odniesienia i różnych typów aproksymacji. Mają one różne własności numeryczne, jednak ich implementacja przebiega według tego samego schematu. Rozważana całka zamieniana jest na sumę wartości funkcji podcałkowej w wybranych punktach pomnożonych przez wagi związane z punktami. Jeśli $\boldsymbol{\xi}^I$ oznacza współrzędne I-tego punktu całkowania w elemencie odniesienia, a v^I wagę z nim związaną, to całkę $\mathcal{C}$ przybliża się wzorem:

$$\mathcal{C} \approx \sum_{I=1}^{N_\mathrm{I}} \hat{b}^1(\boldsymbol{\xi}^I) \frac{\partial \hat{\phi}_m}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}^I) \frac{\partial \boldsymbol{\xi}}{\partial x_1}(\boldsymbol{\xi}^I) \hat{\phi}_l(\boldsymbol{\xi}^I) \det \boldsymbol{J}_{\boldsymbol{T}_e}(\boldsymbol{\xi}^I) v^I \tag{2.48}$$

W powyższym wzorze pominięto zależności pośrednie funkcji b^1. Ostatecznym, niezależnym argumentem współczynników funkcji podcałkowych są współrzędne punktu całkowania, od których zależą wszystkie funkcje pośrednie.

Dobór typu kwadratury i liczby punktów całkowania N_I zależy od funkcji podcałkowej i obszaru całkowania. W analizach niniejszej pracy zakłada się zastosowanie, optymalnych dla całkowania funkcji wielomianowych i opracowanych dla typowych elementów odniesienia, kwadratur Gaussa (różnych rodzajów: Gaussa-Legendre'a, Gaussa-Radau'a lub Gaussa-Lobatto).

Rząd wykorzystywanej kwadratury, określający liczbę punktów całkowania, jest istotnym czynnikiem dla efektywności całkowania numerycznego. W klasycznej teorii zbieżności MES, dla eliptycznych problemów liniowych i elementów izoparametrycznych, przyjmuje się, że całkowanie powinno być dokładne dla wielomianu występującego jako funkcja podcałkowa, bez uzwględnienia transformacji do elementu rzeczywistego [52]. W jednym wymiarze przestrzennym oznacza to, że dla funkcji kształtu będących wielomianami stopnia p konieczne jest wycałkowanie dokładne wielomianów stopnia $2p$, co w przypadku zastosowania kwadratur Gaussa-Legendre'a wymaga użycia $p+1$ punktów całkowania. Jeśli funkcje kształtu w elemencie prostokątnym lub sześciennym są iloczynami kartezjańskimi jednowymiarowych wielomianów o stopniu p, to dla sformułowań (2.30) i (2.31) wystarczające jest zastosowanie np. kwadratury Gaussa-Legendre'a, będącej iloczynem kartezjańskim jednowymiarowych kwadratur rzędu $p+1$ ($p+1$ punktów całkowania w każdym kierunku). Dla innych typów elementów lub innych rodzajów kwadratur zależność liczby punktów cał-

kowania od stopnia aproksymacji jest bardziej złożona [69], [161], jednak rząd zależności pozostaje ten sam, a liczba punktów zbliżona.

2.6.3 Aproksymacja geometrii elementów

Postać procedur całkowania numerycznego i w konsekwencji ich złożoność obliczeniowa zależą od definicji transformacji $\boldsymbol{T}_e$, określających geometrię elementów. Podobnie jak rozwiązanie dyskretyzowanego problemu, również geometria pojedynczych elementów, a w konsekwencji całego obszaru obliczeniowego, może być aproksymowana przy użyciu funkcji kształtu. Współrzędne dowolnego punktu w elemencie rzeczywistym są wtedy kombinacjami liniowymi odpowiednich geometrycznych funkcji kształtu. Zależnie od sposobu aproksymacji i doboru funkcji kształtu współczynnikami kombinacji są albo współrzędne wybranych punktów w elemencie rzeczywistym (tak będzie np. dla interpolacji Lagrange'a), albo inne parametry, np. uzyskane poprzez rozwiązanie szeregu zadań lokalnej projekcji [24].

Jeżeli funkcje kształtu użyte do aproksymacji geometrii są identyczne z użytymi do aproksymacji rozwiązania, mamy do czynienia z elementem izoparametrycznym. W analizach przeprowadzanych w niniejszej pracy i w modelu implementacji zakłada się sytuację ogólniejszą. Dla każdego elementu istnieje N_{G} geometrycznych funkcji kształtu. Współrzędne dowolnego punktu w elemencie obliczane są jako ich kombinacja liniowa o znanych współczynnikach (dostarczanych np. przez generator siatki MES). Podobnie jak przy aproksymacji rozwiązania, funkcje kształtu związane są z elementem odniesienia, natomiast współczynniki kombinacji liniowej (geometryczne stopnie swobody) są odrębne dla każdego elementu rzeczywistego.

Przypadkiem często spotykanym w praktyce jest zastosowanie liniowej aproksymacji geometrii, co odpowiada afinicznemu odwzorowaniu elementu odniesia, będącemu połączeniem translacji, obrotu i przeskalowania osi. Zaletą odwzorowania afinicznego jest stałość macierzy jakobianowych $\boldsymbol{J}_{\boldsymbol{T}_e}$ i $\boldsymbol{J}_{\boldsymbol{T}_e^{-1}}$ w całym elemencie.

Zastosowanie aproksymacji skończenie elementowej pierwszego stopnia zwiększa złożoność obliczeniową transformacji (nie dotyczy to elementów trójkątnych i czworościennych), dając w zamian większą elastyczność w kształtowaniu siatki MES. Geometria elementu rzeczywistego jest wtedy odwzorowywana za pomocą kombinacji wieloliniowych funkcji kształtu, których współczynnikami są współrzędne wierzchołków elementu. Elementy odniesienia mogą zostać odwzorowane w elementy rzeczywiste o dowolnie położonych wierzchołkach. Ceną, jaką płaci się za tę elastyczność, jest zwiększony koszt

obliczeniowy związany ze zmiennością jakobianu transformacji wewnątrz elementu.

Ze względu na powyższe komplikacje w przypadku wieloliniowej aproksymacji geometrii praktyczną alternatywą staje się użycie elementu izoparametrycznego z aproksymacją geometrii wyższego stopnia. Nie ponosi się wtedy dodatkowych kosztów związanych z obliczaniem wartości geometrycznych funkcji kształtu, które są identyczne z funkcjami kształtu dla rozwiązania. Jedyny dodatkowy koszt związany jest z obliczeniem pochodnej $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}$, będącej kombinacją liniową funkcji kształtu.

2.6.4 Złożoność obliczeniowa całkowania numerycznego

Pamięciowa złożoność obliczeniowa całkowania numerycznego jest stała, niezależna od rozmiaru zadania. Koszt przechowywania struktury danych MES (wymagania związane z przechowywaniem danych o siatce i aproksymacji MES) jest zależny od implementacji, i jako taki jest analizowany w rozdz. 4.

Złożoność czasowa całkowania numerycznego określa czas tworzenia macierzy układu $\mathbf{A}$ i wektora prawej strony $\mathbf{b}$, w funkcji wybranych parametrów. Parametrami tymi mogą być: liczba stopni swobody N, składające się na nią – liczba elementów w siatce MES N_{E} i stopień aproksymacji p, a także charakterystyki określające geometrię elementów i rodzaj kwadratury całkowania. W dwóch następnych punktach przedstawiona jest analiza złożoności czasowej w przypadku obliczania całek składających się na wyrazy macierzy $\mathbf{A}$. Złożoność obliczeń związanych z tworzeniem wektora $\mathbf{b}$, ze względu na podobny sposób całkowania pojedynczych wyrazów, pozostaje do niej w proporcji bliskiej proporcji liczby wyrazów wektora $\mathbf{b}$ do liczby niezerowych wyrazów macierzy $\mathbf{A}$. Ta ostatnia proporcja zależy od struktury siatki elementów i typu aproksymacji. Niemniej koszt tworzenia wektora prawej strony $\mathbf{b}$ jest co najmniej o jeden rząd wielkości mniejszy (w przypadku obliczeń trójwymiarowych) od kosztu tworzenia macierzy $\mathbf{A}$.

Liczba operacji związanych z tworzeniem macierzy $\mathbf{A}$, wykonywanych w pojedynczym podobszarze całkowania, wynika bezpośrednio z wzorów całkowania numerycznego. Każda całka z odpowiedniego sformułowania słabego musi zostać obliczona dla każdej pary funkcji bazowych, nie zerujących się w danym podobszarze całkowania. Każdej funkcji bazowej ograniczonej do podobszaru całkowania odpowiada funkcja kształtu[14]. Każda całka obliczana jest jako suma wartości w poszczególnych punktach całkowania.

[14]Wyjątkiem jest wspominana już aproksymacja związana na siatkach nieregularnych. Jednak i w tym wypadku całkowanie numeryczne przeprowadzane jest w sposób opisany dla

Efektem całkowania po elemencie jest lokalna macierz elementowa i elementowy wektor prawej strony[15]. Wyrazy z macierzy elementowej i elementowego wektora prawej strony wpisywane są w odpowiednie miejsca globalnej macierzy układu i globalnego wektora prawej strony w procedurze agregacji. Złożoność obliczeniowa tej ostatniej zależy od implementacji solwera równań liniowych i interfejsu pomiędzy rdzeniem obliczeniowym MES i solwerem. Złożoność ta jest analizowana w rozdz. 4, dotyczącym implementacji algorytmów MES.

Analiza czasowej złożoności obliczeniowej całkowania numerycznego przeprowadzona zostanie w dwóch etapach. W pierwszym rozważone będą wszystkie elementy składowe występujące w sformułowaniach (2.30) i (2.31). Towarzyszyć temu będzie określenie rzędu złożoności dla przypadku trójwymiarowego elementu o aproksymacji geometrii za pomocą N_{G} geometrycznych funkcji kształtu i aproksymacji rozwiązania z użyciem N_{K} funkcji kształtu. W wypadku użycia elementu izoparametrycznego $N_{\mathrm{G}} = N_{\mathrm{K}}$, a w wypadku liniowej aproksymacji geometrii $N_{\mathrm{G}} = 4$. W drugim etapie analizy omówiony zostanie szczegółowo konkretny przykład aproksymacji nieciągłej z zastosowaniem elementów pryzmatycznych stopnia p.

2.6.5 Analiza czasowej złożoności obliczeniowej całkowania numerycznego – przypadek ogólny

Utworzenie lokalnej elementowej macierzy dla sformułowań (2.30) i (2.31) związane jest z obliczeniem, w każdym punkcie całkowania, elementów składowych występujących w odpowiednich całkach. Elementami tymi są:

- wartości funkcji kształtu $\hat{\phi}_m(\boldsymbol{\xi})$ i ich pochodnych względem współrzędnych elementu odniesienia $\frac{\partial \hat{\phi}_m(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$. Liczba koniecznych operacji jest zależna od stopnia aproksymacji. Dla pojedynczego jednowymiarowego wielomianu stopnia p liczba operacji do obliczenia wartości w punkcie, przy zastosowaniu klasycznego wzoru Hornera, wynosi $2p$. Dla funkcji będących iloczynami wielomianów jednowymiarowych rząd złożoności pozostaje ten sam. Dla obliczanych oddzielnie N_{K} funkcji kształtu daje to $O(pN_{\mathrm{K}})$ operacji. W przypadku przestrzeni zupełnych wielomianów stopnia p, której bazę stanowią wszystkie jednomiany o sumie potęg poszczególnych zmiennych mniejszej lub równej p, liczba składników w

klasycznej aproksymacji, a różnice pojawiają się w fazie agregacji globalnej macierzy układu **A** [60].

[15] Tradycyjne nazwy, wywodzące się z zastosowań w mechanice ciała stałego, to elementowa macierz sztywności i elementowy wektor obciążenia.

pojedynczym wielomianie jest rzędu p^3 w przestrzeni i p^2 na płaszczyźnie. Najwyższa potega występująca w składnikach wynosi p. W przypadku odrębnego obliczania wartości dla różnych funkcji kształtu, liczba operacji byłaby rzędu $p^4 N_{\mathrm{K}}$. W praktyce postępuje się inaczej. Najczęściej, w konkretnym punkcie całkowania oblicza się jednocześnie wartość wszystkich funkcji kształtu i ich pochodnych względem współrzędnych w elemencie odniesienia. Pozwala to zmniejszyć liczbę operacji do rzędu N_{K}, ze współczynnikiem przy najwyższej potędze rzędu kilku lub kilkunastu (w dalszych oszacowaniach przyjmowana będzie liczba operacji $8N_{\mathrm{K}}$, spotykana w praktyce przy funkcjach kształtu będących iloczynami tensorowymi);

- wartości geometrycznych funkcji kształtu i ich pochodnych. Liczba operacji jest podobna jak przy obliczaniu wartości funkcji kształtu dla aproksymacji rozwiązania, przy czym rolę N_{K} pełni liczba geometrycznych funkcji kształtu N_{G};

- współrzędne punktu w elemencie rzeczywistym $\boldsymbol{x}(\boldsymbol{\xi}^I)$ i ich pochodne względem współrzędnych elementu odniesienia, tworzące macierz jakobianową $\boldsymbol{J}_{\boldsymbol{T}_e}$. Jeśli wynik jest kombinacją liniową geometrycznych funkcji kształtu, obliczenia wymagają $24N_{\mathrm{G}}$ operacji. Wraz z kosztem obliczenia wartości geometrycznych funkcji kształtu daje to $(8 + 24)N_{\mathrm{G}}$ operacji. Jeśli aproksymacja geometrii nie jest wielomianowa, koszt obliczeń może być znacznie wyższy i zależny od wymaganych działań;

- odwrotność $\boldsymbol{J}^{-1}_{\boldsymbol{T}_e}$ macierzy jakobianowej $\boldsymbol{J}_{\boldsymbol{T}_e}$, $\boldsymbol{J}^{-1}_{\boldsymbol{T}_e} = \boldsymbol{J}_{\boldsymbol{T}^{-1}_e}$. Odwrócenie macierzy 2×2 lub 3×3 wymaga stałej liczby operacji, oznaczanej przez N_{J} (w ostatecznych obliczeniach dla przykładów trójwymiarowych przyjęta jest liczba 54). Przy okazji obliczany jest wyznacznik $\det \boldsymbol{J}_{\boldsymbol{T}_e}$;

- wartości pochodnych funkcji kształtu względem współrzędnych elementu rzeczywistego. Obliczenie trzech pochodnych każdej z funkcji kształtu wymaga $3 \cdot 6$ operacji obliczenia iloczynu z macierzą jakobianową $\boldsymbol{J}_{\boldsymbol{T}^{-1}_e}$. Łączny koszt obliczeń uwzględniający konieczne kroki pośrednie wynosi $(6 + 18)N_{\mathrm{G}} + (6 + 18)N_{\mathrm{K}} + N_{\mathrm{J}}$;

- wartość funkcji niewiadomej. Obliczenie kombinacji liniowej funkcji kształtu wymaga $2N_{\mathrm{K}}$ operacji;

- wartości pochodnych funkcji niewiadomej względem współrzędnych elementu rzeczywistego. Każda z trzech pochodnych jest kombinacją li-

niową pochodnych funkcji kształtu względem współrzędnych elementu rzeczywistego. Łączny koszt obliczeń wynosi $(6 + 18)N_{\mathrm{G}} + (6 + 18+ +6)N_{\mathrm{K}} + N_{\mathrm{J}}$;

- wartość odpowiedniego współczynnika ze sformułowania słabego. Średnia liczba operacji potrzebnych do obliczenia współczynnika w pojedynczym punkcie całkowania będzie oznaczana przez N_{W}. Może ona przybierać wartości od zera (współczynniki stałe w elemencie) do tysięcy, zależnie od problemu.

Klasycznym sposobem całkowania numerycznego jest wykonywanie pętli po punktach całkowania, jako najbardziej zewnętrznej pętli algorytmu. Dzięki temu w pojedynczym punkcie całkowania można obliczyć wartości wszystkich funkcji kształtu i ich pochodnych oraz innych potrzebnych do całkowania wielkości. Następnie w podwójnej pętli po funkcjach kształtu (odpowiadających niewiadomej funkcji i funkcji testującej) następuje wysumowanie całek odpowiadających poszczególnym wyrazom lokalnej macierzy elementowej. Istotnym elementem klasycznego algorytmu całkowania numerycznego jest jednoczesne obliczanie w pojedynczym punkcie całkowania wszystkich całek ze sformułowania słabego, przy wykorzystaniu jednorazowo obliczonych elementów składowych całek. Postępowanie inne, w którym poszczególne całki obliczane są oddzielnie, niesie z sobą zwielokrotnienie nakładów obliczeniowych, ze względu na konieczność wielokrotnych obliczeń tych samych elementów składowych dla różnych całek.

Dla pojedynczego wyrazu macierzy elementowej liczba wyrazów o różnych współczynnikach, odpowiadających różnym kombinacjom pochodnych i wartości funkcji kształtu, może wynosić maksymalnie 13 dla sformułowań (2.30) i (2.31) (9 dla macierzy $\boldsymbol{A}^{ij}$, 3 dla macierzy $\boldsymbol{B}^{i}$ i po jednym dla macierzy $\boldsymbol{C}$ i $\boldsymbol{M}$, $i, j = 1, 2, 3$). Jeśli oznaczymy ją przez N_{C}, to liczba operacji wymaganych do wysumowania pojedynczego wyrazu macierzy elementowej, w pojedynczym punkcie całkowania wynosić będzie $3N_{\mathrm{C}}$. W połączeniu z liczbą wyrazów w macierzy elementowej N_{K}^2 i liczbą punktów całkowania N_{I} daje to $3N_{\mathrm{C}}N_{\mathrm{I}}N_{\mathrm{K}}^2$ operacji.

Ostatecznie złożoność czasową algorytmu całkowania numerycznego dla utworzenia pojedynczej macierzy elementowej można oszacować z góry przez wielkość $\left(3N_{\mathrm{C}}N_{\mathrm{K}}^2 + N_{\mathrm{C}}N_{\mathrm{W}} + 34N_{\mathrm{G}} + 34N_{\mathrm{K}} + N_{\mathrm{J}}\right) N_{\mathrm{I}}$, gdzie uwzględnione zostały wszystkie nakłady w pojedynczym punkcie całkowania[16].

[16]W przypadku problemów wektorowych i jednakowych funkcji kształtu dla wszystkich składowych funkcji niewiadomej i testującej, modyfikacja złożoności czasowej po-

W wielu zastosowaniach, dla których czas rozwiązania układu równań nie dominuje znacząco nad czasem tworzenia macierzy układu, wysoka złożoność czasowa całkowania numerycznego stanowi istotną wadę metody elementów skończonych. Z obliczeniowego punktu widzenia metoda różnic skończonych (MRS) odpowiada metodzie elementów skończonych, dla której uprzednio dokonano całkowania numerycznego. Aby uczynić MES konkurencyjną z MRS, procedura całkowania numerycznego musi zostać poddana optymalizacji.

Najdalej idącą optymalizacją byłoby uniknięcie całkowania numerycznego w ogóle. Dla liniowych zagadnień o współczynnikach stałych w elemencie oraz dla siatek elementów będących afinicznymi odwzorowaniami elementów odniesienia całkowanie numeryczne można przeprowadzić jednorazowo dla każdej kombinacji typu elementu oraz stopnia aproksymacji, a następnie wykorzystywać przechowywane wartości. Dla obliczeń z dużą liczbą elementów w obszarach o złożonej geometrii korzystną techniką jest specjalna wersja wieloblokowej generacji siatki, w której dla bloków blisko brzegu obszaru generuje się siatki elementów izoparametrycznych, dobrze aproksymujących geometrię, natomiast wewnątrz obszaru tworzy się bloki elementów liniowych, dla których można zoptymalizować całkowanie numeryczne.

Dla problemów liniowych i nieliniowej aproksymacji geometrii elementów, dominującym kosztem jest sumowanie wyrazów macierzy elementowej i obliczanie pochodnych funkcji kształtu względem współrzędnych fizycznych (współrzędnych elementu rzeczywistego). Pochodne te są różne dla każdego punktu całkowania. Obniżenie złożoności można uzyskać poprzez jednorazowe obliczenie i stablicowanie wartości funkcji kształtu i ich pochodnych względem współrzędnych elementu odniesienia we wszystkich punktach całkowania.

W przypadku problemów nieliniowych i liniowej aproksymacji geometrii elementów stablicować można wszystkie składniki całek, z wyjątkiem nieliniowych współczynników. Nie udaje się jednak uniknąć sumowania wyrazów macierzy elementowej w potrójnej pętli.

Dla problemów nieliniowych i nieliniowej aproksymacji geometrii elementów konieczne jest wykonanie wszystkich obliczeń w punktach całkowania. Dla elementów niższego rzędu koszt obliczeń elementów składowych całek może przewyższać koszt sumowania wyrazów macierzy elementowej. Jednak dla

---

lega na zwiększeniu kosztu obliczenia współczynników całek (każdy współczynnik jest macierzą $N_u \times N_u$) oraz zwiększeniu złożoności sumowania, dla którego dochodzą pętle po składowych funkcji wektorowych. Ostatecznie złożoność całkowania ma postać $\left(3N_{\mathrm{C}}N_u^2N_{\mathrm{K}}^2 + N_{\mathrm{C}}N_{\mathrm{W}}N_u^2 + 34N_{\mathrm{G}} + 34N_{\mathrm{K}} + N_{\mathrm{J}}\right)N_{\mathrm{I}}$. Zważywszy na zwiększenie, w porównaniu z przypadkiem skalarnym, liczby stopni swobody, która wynosi N_uN_{K}, liczba operacji przypadająca na jeden stopień swobody zmniejsza się.

aproksymacji stopnia wyższego od trzech ten ostatni jest z reguły istotniejszy (wyjątkiem mogłyby być problemy z bardzo złożonymi współczynnikami).

W przypadku gdy koszt sumowania wyrazów macierzy jest dominujący, można w niektórych sytuacjach osiągnąć istotną redukcję złożoności czasowej, także dla przypadków nieliniowych fizycznie i geometrycznie. Dla funkcji kształtu będących iloczynami tensorowymi i kwadratur będących iloczynami tensorowymi można zredukować liczbę operacji potrzebnych do wysumowania wyrazów macierzy elementowej, poprzez połączenie całkowania i obliczania wartości funkcji kształtu oraz zastosowanie tablicowania wybranych wielkości. Procedura polega na tym, że nie dokonuje się tensorowego iloczynu funkcji kształtu, lecz pozostawia wielomiany z przestrzeni niższego wymiaru. Następnie dokonuje się całkowania w przestrzeni niższego wymiaru dla wszystkich kombinacji tak uzyskanych funkcji kształtu i ich pochodnych. Wyniki zapisuje się w tablicy, a następnie wykorzystuje do całkowania w pozostałych wymiarach przestrzennych, w trakcie których dokonuje się ostatecznego iloczynu tensorowego funkcji kształtu. Tym sposobem można zmniejszyć rząd złożoności czasowej sumowania wyrazów macierzy elementowej do $pN_{\mathrm{K}}N_{\mathrm{I}}$. Zmodyfikowany algorytm ma istotny narzut obliczeniowy związany z bardziej złożoną strukturą pętli i wykazuje zdecydowaną przewagę nad algorytmem klasycznym od stopni aproksymacji powyżej pięciu [119].

Dalsze oszczędności może przynieść użycie specjalnych, dostosowanych do siebie funkcji kształtu i kwadratur całkowania. Jeśli funkcje kształtu związane z wnętrzem elementu dobrane zostaną tak, że będą się zerować w większości punktów całkowania (praktycznie wszystkich, z wyjątkiem kilku), to większość wyrazów macierzy elementowej będzie się zerować, ich obliczanie będzie można pominąć, a rząd złożoności sumowania wyrazów macierzy elementowej ulegnie dalszej redukcji do wartości $N_{\mathrm{K}}N_{\mathrm{I}}$.

2.6.6 Analiza czasowej złożoności obliczeniowej całkowania numerycznego – przypadek szczególny jednorodnej aproksymacji nieciągłej dla problemu dyfuzji

Poniżej przedstawiony jest przykład analizy złożoności czasowej całkowania numerycznego dla konkretnej uproszczonej sytuacji:

- rozwiązywanym problemem jest skalarne nieliniowe równanie dyfuzji:

$$\left(k^{ij}(\nabla u)u_{,j}\right)_{,i} = s \tag{2.49}$$

- aproksymacja odbywa się za pomocą funkcji nieciągłych (zmodyfikowane sformułowanie 2.31), każda funkcja bazowa jest określona (niezerowa) we wnętrzu pojedynczego elementu,
- wykorzystane są geometrycznie liniowe elementy pryzmatyczne o jednakowym stopniu aproksymacji rozwiązania,
- dla aproksymacji stopnia p funkcjami kształtu w elemencie odniesienia są iloczyny tensorowe zupełnych wielomianów stopnia p na płaszczyźnie xy i jednowymiarowych wielomianów stopnia p w kierunku z.

Dla rozważanego problemu sformułowanie (2.31) upraszcza się do wzoru:

$$\begin{aligned}
&\int_{\Omega} k^{ij}(\nabla u) w_{,i} u_{,j} d\Omega + \\
&+ \int_{\Gamma_{\text{int}}} \left(< k^{ij}(\nabla u) n^i w_{,j} > [u] - [w] < k^{ij}(\nabla u) n^i u_{,j} > \right) d\Gamma + \\
&+ \int_{\Gamma_{\text{D}}} \left(k^{ij}(\nabla u) n^i w_{,j} u - k^{ij}(\nabla u) n^i w u_{,j} \right) d\Gamma = \\
&= \int_{\Omega} s w d\Omega + \int_{\Gamma_{\text{D}}} k^{ij}(\nabla u) n^i w_{,j} f_{\text{D}} d\Gamma
\end{aligned} \tag{2.50}$$

Dla oceny złożoności czasowej całkowania numerycznego istotne są tylko dwa pierwsze wyrazy (2.50). Wynika to z następujących faktów:

- przy obliczeniach wielkiej skali liczba boków wewnętrznych zdecydowanie przewyższa liczbę boków zewnętrznych z warunkiem Dirichleta,
- obliczenia dla pojedynczego boku wewnętrznego pochłaniają ponad czterokrotnie więcej operacji niż obliczenia dla pojedynczego boku zewnętrznego,
- całki po prawej stronie obliczane są dla pojedynczych funkcji kształtu i wymagają istotnie krótszego czasu obliczeń niż całki tworzące macierz elementową.

W rozważanym przypadku całkowanie numeryczne w pojedynczym elemencie nie wymaga obliczania współrzędnych punktów całkowania w elemencie rzeczywistym, posiada także stałe jakobiany transformacji, które oblicza się jednorazowo dla każdego elementu. To samo odnosi się do całkowania po brzegach elementów, przy czym dodatkowo oblicza się jednokrotnie dla każdego brzegu współrzędne jednostkowego wektora normalnego.

Tablica 2.3. Liczba punktów całkowania w kwadraturze Gaussa-Legendre'a dla elementu trójkątnego potrzebnych do dokładnego obliczenia całki z wielomianu stopnia $2(p-1)$

Stopień aproksymacji p	2	3	4	5	6	7	8	9	10
Liczba punktów całkowania	3	6	12	16	25	33	42	52	70

Dla pierwszego wyrazu wzoru (2.50), całkowania po obszarach elementów, liczba operacji będzie zależna od:

- liczby elementów N_{E},
- liczby całek do obliczenia N_{C} (każda całka odpowiada współczynnikowi k^{ij} i odpowiedniej kombinacji pochodnych funkcji bazowych). W ogólnym anizotropowym przypadku (wszystkie składowe k^{ij} różne od zera) N_{C} jest równe 9,
- nakładów N_{W} związanych z obliczeniem pojedynczego współczynnika k^{ij},
- stopnia aproksymacji p,
- liczby funkcji kształtu N_{K} w elemencie, $N_{\mathrm{K}} = \frac{1}{2}(p+1)^2(p+2)$,
- liczby punktów całkowania N_{I}, równej liczbie punktów na płaszczyźnie potrzebnych do dokładnego wycałkowania wielomianów stopnia $2(p-1)$, podanej w tabl. 2.3 i oznaczanej przez p_{xy}, pomnożonej przez liczbę p punktów w kierunku osi z[17]. Ostateczna liczba punktów całkowania jest rzędu p^3, bliska $\frac{2}{3}p^3$.

Dla drugiego składnika wzoru (2.50), sumy całek po wewnętrznych bokach elementów, funkcję liczby elementów pełni liczba boków wewnętrznych, a liczba punktów całkowania obliczana jest dla całek powierzchniowych. Przy zagęszczaniu siatki poprzez jednorodne izotropowe podziały elementów pryzmatycznych, liczba boków zmierza asymptotycznie do $\frac{5}{2}$ liczby elementów. Przy każdorazowym zagęszczeniu liczba boków wewnętrznych przyrasta dwa razy szybciej

[17] Ze względu na niewystępowanie wyrazów rzędu zerowego w sformułowaniu (2.50) rząd kwadratury jest niższy o jeden w porównaniu z przypadkiem ogólnym.

niż liczba boków zewnętrznych. Do oszacowań przyjęta jest, asymptotycznie dokładna, liczba boków wewnętrznych równa $\frac{5}{2}$ liczby elementów.

Liczba punktów całkowania jest różna dla boków prostokątnych i boków trójkątnych. Dla tych pierwszych wynosi p^2, przy założeniu użycia iloczynu tensorowego kwadratur, identycznych jak kwadratura zastosowana wzdłuż osi z wewnątrz elementu. Dla boków trójkątnych zastosowanie kwadratury Gaussa-Legendre'a prowadzi do liczby punktów całkowania bliskiej $\frac{2}{3}p^2$ (patrz tabl. 2.3).

W sformułowaniu (2.50) występują dwa wyrazy całkowane po bokach elementów i tylko jeden wycałkowywany we wnętrzu. Ponadto liczba par funkcji kształtu, dla których obliczane są całki na bokach elementów wynosi $4N_{\mathrm{K}}^2 = (p+1)^4(p+2)^2$. W całkach tych występują iloczyny skalarne wierszy macierzy współczynników k^{ij} i wektora normalnego do brzegu. W efekcie liczba całek jest równa 3, natomiast koszt obliczeniowy powyższych operacji dla pojedynczej całki wynosi $6N_{\mathrm{W}}$.

Ostatecznie złożoność czasowa obliczania całek wynosi:

- w przypadku całek po wnętrzach elementów:

$$\left(\frac{3}{4}N_{\mathrm{C}}(p+1)^4(p+2)^2 + N_{\mathrm{W}}N_{\mathrm{C}} + 17(p+1)^2(p+2) + 190\right)\frac{2}{3}p^3N_{\mathrm{E}} \qquad (2.51)$$

- natomiast dla całek po bokach elementów:

$$\left(6N_{\mathrm{C}}(p+1)^4(p+2)^2 + 2N_{\mathrm{W}}N_{\mathrm{C}} + 34(p+1)^2(p+2) + 380\right)\frac{13}{6}p^2N_{\mathrm{E}} \qquad (2.52)$$

W celu dalszego skonkretyzowania porównania rozważony jest przypadek zagadnienia dyfuzji (2.2). Dzięki znajomości rozwiązania dokładnego możliwe staje się dla tego przypadku obliczenie błędu rozwiązania przybliżonego i porównanie efektywności aproksymacji różnych stopni[18].

Rysunek 2.9 pokazuje przykładowe rozwiązanie problemu dla siatki z rys. 2.8 i stopnia aproksymacji $p = 3$. Tablica 2.4 oraz rys. 2.10 i 2.11 przedstawiają zbieżność aproksymacji NG dla różnych stopni aproksymacji od 2 do 7. W celu uzyskania zaprezentowanych tam wyników wykorzystane zostały siatki powstałe w efekcie równomiernego zagęszczania siatki z rys. 2.8. Pojedyncza adaptacja siatki powoduje dwukrotne zmniejszenie charakterystycznego liniowego rozmiaru elementu h. Oszacowania błędu dla nieciągłej aproksymacji

[18]Wprawdzie rozpatrywany problem nie jest już nieliniowy, ale analizie efektywności poddane zostanie całkowanie bez uproszczeń jakie to potencjalnie wnosi.

Tablica 2.4. Aproksymacja równania Laplace'a w sześcianie jednostkowym z zastosowaniem nieciągłej dyskretyzacji Galerkina: eksperymentalna szybkość zbieżności (wartość wykładnika potęgi h w oszacowaniu błędu) w normach L^2 i H^1, dla różnych stopni aproksymacji p

Stopień aproksymacji	2	3	4	5	6	7
Zbieżność L^2	1.948	3.970	4.588	5.907	6.334	7.694
Zbieżność H^1	1.972	3.042	4.004	5.026	5.402	7.289

Galerkina zawierają po prawej stronie wyrażenie postaci $C(u,p)h^s$. Liczby w tabl. 2.4 są uzyskanymi eksperymentalnie wartościami wykładnika s. Zbieżność metody elementów skończonych jest optymalna, jeśli $s = p+1$ dla normy L^2 i $s = p$ dla normy H^1. Oznacza to, że np. dla stopnia aproksymacji $p = 3$ jednokrotna równomierna adaptacja siatki powinna powodować 16-krotną redukcję błędu L^2 i 8-krotną redukcję błędu H^1.

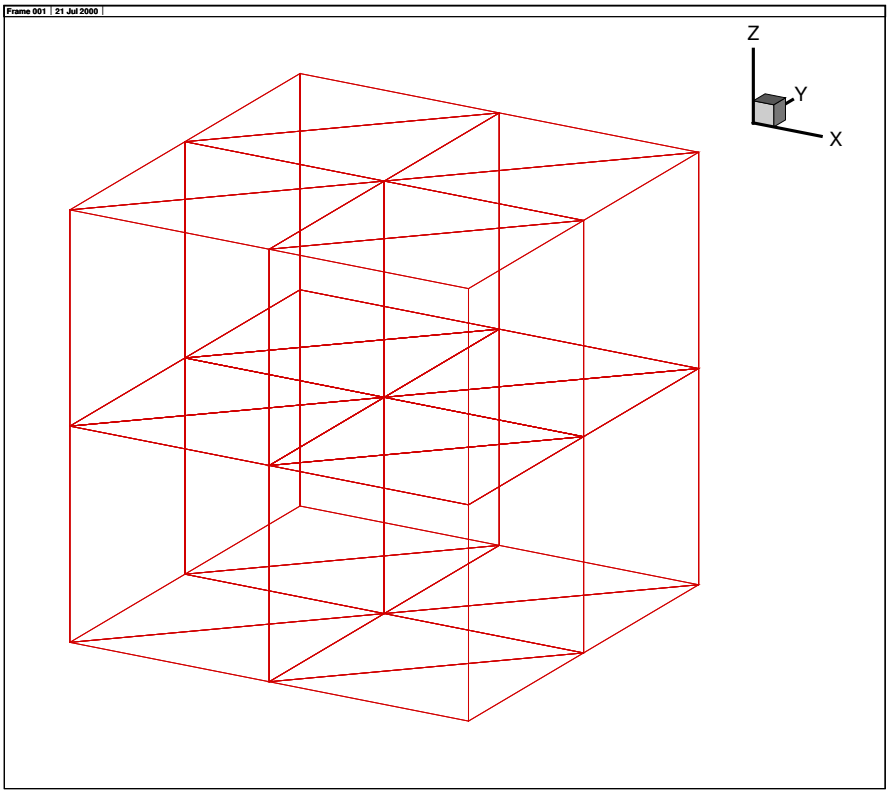
Eksperymenty potwierdzają uzyskane teoretycznie oszacowania błędu, w tym znany z teorii [125] fakt nieoptymalnej zbieżności w normie L^2 dla aproksymacji nieciągłej z parzystymi stopniami p. Na wykresach 2.10 i 2.11 widoczna jest wysoka dokładność aproksymacji wyższych stopni. Fakt ten jest typowy dla aproksymacji problemów o rozwiązaniach gładkich. Przy tej samej liczbie stopni swobody rozwiązania dla wysokich p dają błędy o kilka rzędów wielkości niższe niż w przypadku aproksymacji niskich stopni. W przeprowadzanych analizach dokładności pomija się jednak często wyższy koszt związany z aproksymacją wysokiego stopnia, mający wpływ na ogólną opłacalność metody. Poniżej przedstawiona jest analiza złożoności czasowej, uzupełniająca porównanie aproksymacji różnych stopni. Ostateczne konkluzje co do wyboru stopnia aproksymacji można podjąć dodając do analiz błędu aproksymacji i złożoności obliczeniowej całkowania numerycznego analizy złożoności obliczeniowej solwera układów równań liniowych. Analizy takie zawarte są w p. 2.7.

Przyjmując dla rozważanego przypadku $N_{\mathrm{W}} = 1$ i $N_{\mathrm{C}} = 9$, wzory na złożoność czasową całkowania numerycznego upraszczają się do postaci:

$$\left(\frac{9}{4}(p+1)^4(p+2)^2 + 17(p+1)^2(p+2) + 193\right)\frac{2}{3}p^3 N_{\mathrm{E}} \tag{2.53}$$

dla całek po wnętrzach elementów i wyrażenia:

$$\left(18(p+1)^4(p+2)^2 + 34(p+1)^2(p+2) + 386\right)\frac{13}{6}p^2 N_{\mathrm{E}} \tag{2.54}$$

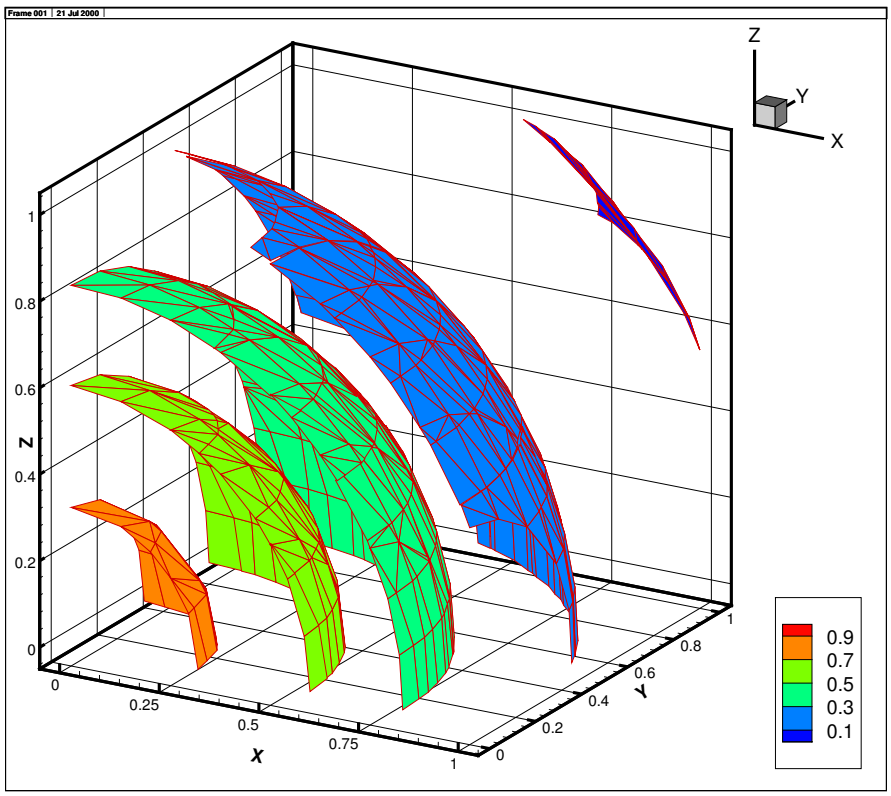


Rys. 2.8. Aproksymacja równania Laplace'a w sześcianie jednostkowym z zastosowaniem nieciągłej dyskretyzacji Galerkina: przykładowa siatka pryzmatyczna

dla całek po bokach elementów. Wzory powyższe pozwalają na porównanie kosztu całkowania przypadającego na pojedynczy element. Innym możliwym porównaniem jest przyjęcie stałej liczby stopni swobody w całym problemie i porównanie kosztu całkowania dla różnych stopni p. Globalna, całkowita liczba stopni swobody N jest równa liczbie elementów pomnożonej przez liczbę stopni swobody w elemencie. Obliczona z tej zależności liczba elementów wynosi:

$$N_{\mathrm{E}} = \frac{2N}{(p+1)^2(p+2)} \tag{2.55}$$

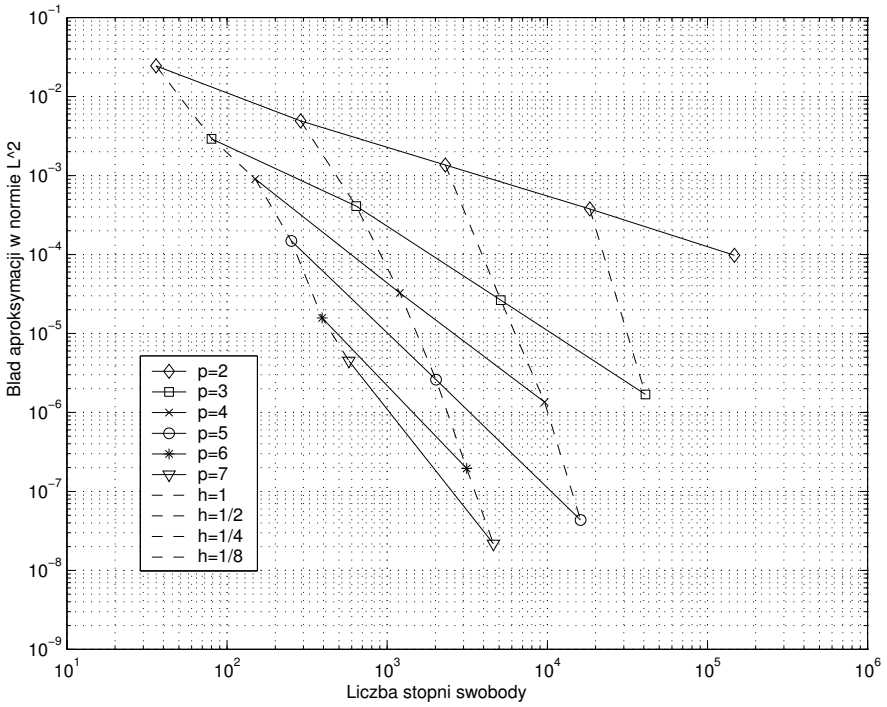
Po podstawieniu (2.55) do (2.53) i (2.54) otrzymuje się koszt całkowania numerycznego przypadający na pojedynczy stopień swobody w zależności od stopnia aproksymacji p.



Rys. 2.9. Aproksymacja równania Laplace'a w sześcianie jednostkowym z zastosowaniem nieciągłej dyskretyzacji Galerkina: rozwiązanie w postaci warstwic dla siatki z rys. 2.8 i stopnia aproksymacji $p = 3$

Na rysunku 2.12 przedstawione są wykresy funkcji (2.53) i (2.54) oraz ich odpowiedników po podstawieniu wzoru (2.55). Oznaczenia krzywych są następujące:

- EL/N_{E} – złożoność czasowa całkowania po wnętrzach elementów przypadająca na pojedynczy element,
- BOK/N_{E} – złożoność czasowa całkowania po bokach elementów przypadająca na pojedynczy element,
- EL/N – złożoność czasowa całkowania po wnętrzach elementów przypadająca na pojedynczy globalny stopień swobody,



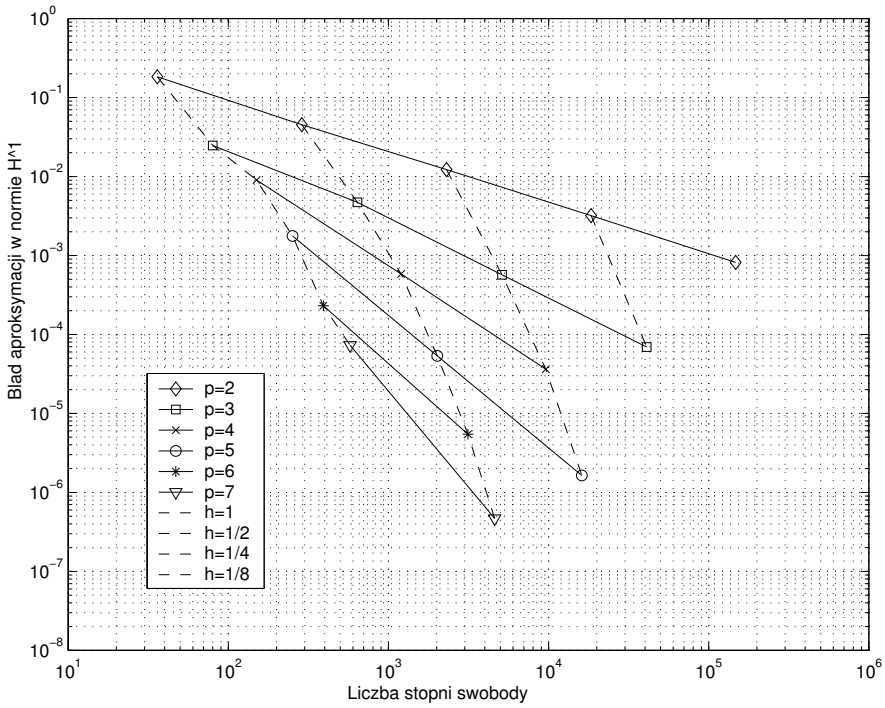
Rys. 2.10. Aproksymacja równania Laplace'a w sześcianie jednostkowym z zastosowaniem nieciągłej dyskretyzacji Galerkina: krzywe zbieżności w normie L^2 dla różnych stopni aproksymacji p

- BOK/N – złożoność czasowa całkowania po bokach elementów przypadająca na pojedynczy globalny stopień swobody.

W tablicy 2.5 przedstawione są liczby operacji EL/N i BOK/N dla stopni p od 2 do 7. I na rys. 2.12, i w tabl. 2.5 jednostką liczby operacji jest Mflop – milion operacji zmiennoprzecinkowych.

Okazuje się, że mimo niższego rzędu złożoności czasowej, dla stosowanych praktycznie stopni aproksymacji, koszt związany z całkowaniem po bokach elementów dominuje nad kosztem całkowania po wnętrzach elementów (złożoność całkowania po wnętrzach zaczyna przeważać dopiero dla $p > 30$). Wszelkie działania optymalizujące całkowanie numeryczne dla aproksymacji nieciągłej powinny więc szczególny nacisk kłaść na całki powierzchniowe.

Absolutne czasy całkowania dla liczby stopni swobody rzędu milionów mogą sięgać minut, a nawet godzin, dla pojedynczego współczesnego procesora (o wy-



Rys. 2.11. Aproksymacja równania Laplace'a w sześcianie jednostkowym z zastosowaniem nieciągłej dyskretyzacji Galerkina: krzywe zbieżności w normie H^1 dla różnych stopni aproksymacji p

dajności rzędu 1 Gflop/s). Wskazuje to na konieczność optymalizacji procedur całkowania numerycznego przy implementacji kodów MES.

Rysunki 2.10, 2.11 i 2.12 oraz tablica 2.5 pozwalają ocenić koszt obliczeniowy związany z całkowaniem numerycznym, wymagany do osiągnięcia założonego błędu aproksymacji, w zależności od stopnia aproksymacji. Na przykład, aby uzyskać przybliżenia o błędzie w normie H^1 równym 10^{-3} można użyć elementów stopnia 3 z ok. $3 \cdot 10^3$ stopniami swobody lub elementów stopnia 5 z ok. $3.4 \cdot 10^2$ stopniami swobody. I w pierwszym, i w drugim przypadku liczba operacji przy całkowaniu numerycznym wyniesie ok. 200 Mflop. Jeśli jednak założony błąd będzie wynosił 10^{-6}, to należy użyć elementów stopnia 3 z ok. $3 \cdot 10^6$ stopniami swobody lub elementów stopnia 5 z ok. $2.1 \cdot 10^4$ stopniami swobody. Teraz liczba operacji przy całkowaniu numerycznym będzie równa ok. 200 Gflop dla aproksymacji stopnia 3 i tylko ok. 12 Gflop dla aproksymacji

Tablica 2.5. Koszt całkowania numerycznego w przypadku zastosowania nieciągłej dyskretyzacji Galerkina do aproksymacji równania Laplace'a w sześcianie jednostkowym, w milionach operacji zmiennoprzecinkowych [Mflop] na jeden stopień swobody, przy różnych stopniach aproksymacji p

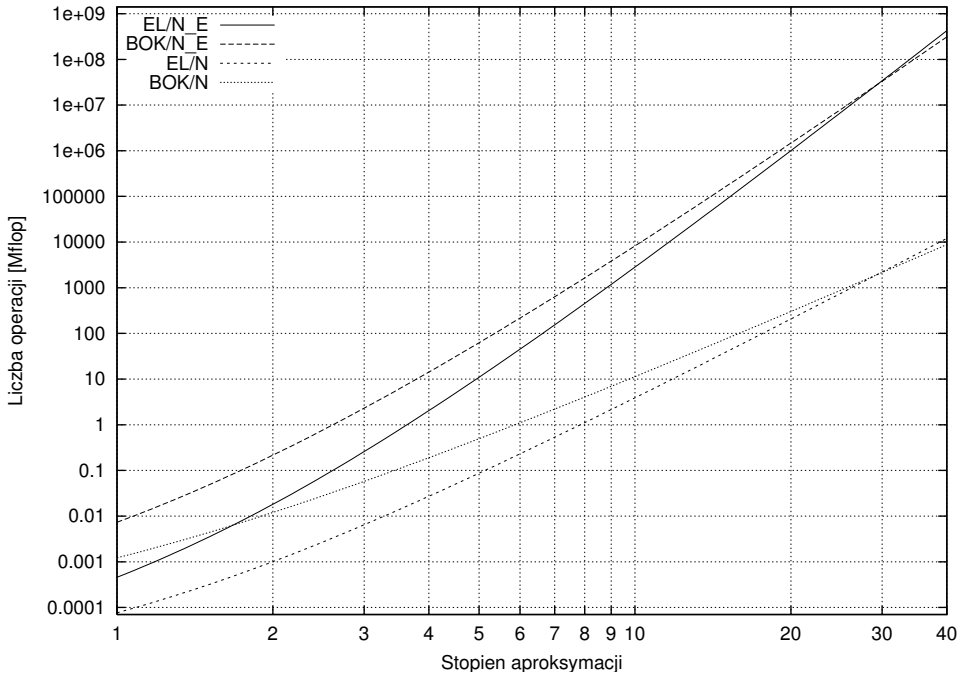
Stopień aproksymacji	2	3	4	5	6	7
Całki po elementach	0.0011	0.0072	0.0304	0.0975	0.2590	0.6006
Całki po bokach	0.0120	0.0576	0.1897	0.4952	1.1062	2.2089

stopnia 5.

Z wykresów widać, że dla małej liczby stopni swobody i małej liczby elementów wysoki koszt całkowania numerycznego może zadecydować o nieopłacalności aproksymacji wyższego stopnia. Jednak już dla stosunkowo niewielkiej liczby elementów eksponencjalna szybkość redukcji błędu wraz ze wzrostem p, w porównaniu z wielomianowym przyrostem liczby operacji, sprawia, że optymalna staje się aproksymacja możliwie wysokiego stopnia.

W przypadku adaptacji *hp* zależności rozważane powyżej, nawet dla tak prostego przykładu, są trudne do uzyskania analitycznie. Powodem jest złożoność jednoczesnego wpływu zagęszczania siatki i podnoszenia stopnia aproksymacji. Łatwiejsza jest analiza przypadku adaptacji typu h, zagęszczania siatki przy stałym stopniu aproksymacji. Każdorazowe podziały elementów powodują wzrost kosztu całkowania proporcjonalny do wzrostu liczby stopni swobody. Jednokrotne równomierne zagęszczenie siatki trójwymiarowej powoduje dwukrotne zmniejszenie rozmiaru liniowego elementów i 8-krotny wzrost liczby stopni swobody. Redukcja błędu aproksymacji jest zależna od stopnia aproksymacji i normy, w której mierzymy błąd. Dla normy L^2 i stopnia aproksymacji $p = 2$, w efekcie nieoptymalnej zbieżności uzyskuje się przy równomiernym zagęszczeniu siatki tylko 4-krotną redukcję błędu. Dla stopni aproksymacji większych od dwóch redukcja błędu przybliżenia w normie L^2 jest zawsze większa niż wzrost liczby stopni swobody. Dla normy H^1 przypadkiem rozgraniczającym jest $p = 3$, dla którego przyrost liczby stopni swobody i redukcja błędu zrównują się. W każdym przypadku opłacalność adaptacji wzrasta wraz ze wzrostem stopnia aproksymacji p[19].

[19]Należy podkreślić, że przedstawione rozważania dotyczą prostego zagadnienia w prostym obszarze obliczeniowym. Dla problemów nieliniowych i złożonych geometrii zależność błędu aproksymacji od stopnia aproksymacji znacznie się komplikuje. Dla wielu problemów apro-



Rys. 2.12. Złożoność czasowa całkowania numerycznego w funkcji stopnia aproksymacji p w przypadku zastosowania nieciągłej dyskretyzacji Galerkina do aproksymacji równania Laplace'a w sześcianie jednostkowym; EL – całki po wnętrzach elementów, BOK – całki po bokach elementów, liczba operacji na pojedynczy element (N_E) i liczba operacji na pojedynczy stopień swobody (N) (szczegółowe objaśnienia w tekście)

2.6.7 Wnioski

Z powyższych analiz wynika kilka wniosków istotnych dla implementacji programów MES:

- całkowanie numeryczne może stanowić istotny czynnik wpływający na efektywność obliczeń MES,
- złożoność czasowa całkowania numerycznego zależy od wielu czynników: rozwiązywanego problemu, typu i stopnia aproksymacji, użytej siatki MES, metody całkowania,

ksymacja wyższego stopnia przynosi korzyści tylko w połączeniu z lokalną adaptacją typu p lub hp.

- koszt całkowania numerycznego jest szczególnie wysoki dla aproksymacji wyższego stopnia,
- istnieją grupy zagadnień, dla których zysk zmniejszenia potrzebnej liczby elementów dla uzyskania założonej dokładności obliczeń w przypadku zastosowania aproksymacji wyższego stopnia przeważa nad zwiększonym kosztem całkowania w pojedynczym elemencie,
- istnieją techniki redukcji złożoności czasowej całkowania numerycznego dla wielu szczególnych przypadków,
- implementacja rdzenia obliczeniowego MES, pomimo złożoności zależności związanych z całkowaniem numerycznym, musi starać się zaoferować czytelną strukturę kodu przeprowadzającego całkowanie i zapewnić wysoką wydajność obliczeń,
- zadanie powyższe wydaje się szczególnie trudne przy założeniu modularności kodu MES i wielokrotnego użycia jego komponentów.

2.7 Rozwiązywanie układów równań liniowych

Rozwiązanie układu równań liniowych jest procedurą, która najczęściej decyduje o obliczeniowej efektywności aproksymacji MES. Z tego względu w programach MES szczególny nacisk kładzie się na implementację solwera liniowego (podsystemu rozwiązywania układów równań liniowych), w tym na implementację równoległą.

Najważniejszą zaletą metod bezpośredniego rozwiązywania układów równań liniowych jest gwarancja uzyskania wyniku dla każdego nieosobliwego układu równań (przy pominięciu wpływu błędów zaokrągleń). Najpoważniejszą wadą są znaczne nakłady czasu i pamięci potrzebne do realizacji algorytmów. Dlatego też w niniejszej pracy szczegółowej analizie poddane są wyłącznie algorytmy iteracyjne, oferujące w przypadku odpowiedniej zbieżności znacznie niższe złożoności obliczeniowe. Ogólna charakterystyka złożoności dla obu typów algorytmów zawarta jest w p. 2.7.7.

Analizy w niniejszej pracy przeprowadzone będą dla pewnego specyficznego algorytmu, będącego uogólnieniem metod Jacobiego i Gaussa-Seidla. Na jego podstawie można zbudować szereg metod iteracyjnego rozwiązywania układów równań liniowych, począwszy od klasycznych metod iteracji prostej aż po złożone metody podprzestrzeni Kryłowa z wielosiatkową poprawą uwarunkowania macierzy układu równań. Algorytm jest wydajny w praktyce i przystosowany

do szerokiej klasy zagadnień oraz związanych z nimi układów równań. Posiada prostą implementację równoległą, która skaluje się w praktyce do układów o liczbie stopni swobody rzędu dziesiątek i setek milionów.

2.7.1 Algorytmy Jacobiego i Gaussa-Seidla

Algorytmy Jacobiego i Gaussa-Seidla, które w niniejszej pracy zawsze rozważane są w wersjach blokowych [154], są reprezentantami klasycznych metod iteracji prostej. Pojedyncza iteracja blokowej metody Jacobiego lub Gaussa-Seidla zastosowana do rozwiązania układu równań liniowych (2.41) może zostać zapisana jako sekwencja problemów lokalnych postaci:

$$\mathbf{A}_{ll} \cdot \mathbf{u}_l^{m,l} = - \sum_{k=1,k\neq l}^{N_{\mathrm{bl}}} \mathbf{A}_{lk} \cdot \tilde{\mathbf{u}}_k + \mathbf{b}_l \tag{2.56}$$

Powyżej $\mathbf{u}_l$ oznacza podwektor stopni swobody, będący częścią globalnego wektora $\mathbf{u}$, związany z pojedynczym obiektem siatki, dowolną grupą obiektów lub grupą obiektów tworzących klasyczny podobszar Ω_l obszaru obliczeniowego Ω. Podobszary Ω_l mogą zachodzić na siebie (jeśli nie prowadzi to do zachodzenia na siebie podwektorów $\mathbf{u}_l$). N_{bl} oznacza liczbę podwektorów $\mathbf{u}_l$. Bloki $\mathbf{A}_{lk}$ odpowiadają parom podwektorów $\mathbf{u}_l$, a co za tym idzie parom obiektów siatki, grup obiektów siatki lub podobszarów, zależnie od definicji $\mathbf{u}_l$. Podwektory $\mathbf{b}_l$, fragmenty globalnego wektora $\mathbf{b}$, odpowiadają podwektorom $\mathbf{u}_l$. $\mathbf{u}_l^{m,l}$ oznacza podwektor $\mathbf{u}_l$ w trakcie wykonywania m-tej iteracji, po rozwiązaniu l problemów lokalnych (2.56). $\tilde{\mathbf{u}}$ jest aktualnym rozwiązaniem, uzyskiwanym różnie dla różnych metod. Dla iteracji Jacobiego uaktualnienie rozwiązania następuje po rozwiązaniu wszystkich problemów lokalnych, stąd $\tilde{\mathbf{u}}_k = \mathbf{u}_k^{m,0}$. Dla iteracji Gaussa-Seidla uaktualnienia dokonuje się po rozwiązaniu każdego problemu lokalnego, tak więc $\tilde{\mathbf{u}}_k = \mathbf{u}_k^{m,l-1}$. W obu wypadkach $\mathbf{u}^{m,0} = \mathbf{u}^{m-1}$ jest wektorem początkowym dla m-tej iteracji.

W klasycznej analizie metod iteracji prostej [85] zapisywane są one w postaci:

$$\mathbf{u}^m = \mathbf{S}^{-1}\mathbf{Q} \cdot \mathbf{u}^{m-1} - \mathbf{S}^{-1}\mathbf{b} \tag{2.57}$$

gdzie $\mathbf{S}$ i $\mathbf{Q}$ uzyskiwane są w efekcie rozkładu macierzy $\mathbf{A}$:

$$\mathbf{A} = \mathbf{S} - \mathbf{Q} \tag{2.58}$$

Dla metod Jacobiego i Gaussa-Seidla wykorzystuje się rozkład na części: trójkątną dolną $\mathbf{L}$, diagonalną $\mathbf{D}$ i trójkątną górną $\mathbf{U}$. Pojedyncza iteracja metody

Jacobiego zapisywana jest jako:

$$\mathbf{u}^m = -\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\cdot\mathbf{u}^{m-1} + \mathbf{D}^{-1}\mathbf{b} \tag{2.59}$$

natomiast pojedyncza iteracja metody Gaussa-Seidla przybiera postać:

$$\mathbf{u}^m = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\cdot\mathbf{u}^{m-1} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b} \tag{2.60}$$

Metody iteracji prostej są zbieżne, jeśli promień spektralny macierzy $\mathbf{S}^{-1}\mathbf{Q}$ jest mniejszy od 1.

W klasycznych sformułowaniach blokowych wariantów metod Jacobiego i Gaussa-Seidla podwektory $\mathbf{u}_l$ i odpowiadające im bloki macierzy $\mathbf{A}$ nie zachodzą na siebie. Jeśli zachodzenie występuje, nadal można stosować sekwencje rozwiązań problemów (2.56) dla iteracyjnego rozwiązania układu (2.41). Wynikające z tej techniki metody są uogólnieniami iteracji prostych, należącymi do metod Schwarza, czyli metod dekompozycji obszaru na nakładające się podobszary [162].

W analizie metod Schwarza wprowadza się dwa rodzaje operatorów: operatory zawężenia (restrykcji) $\mathbf{R}_l$ i operatory rozszerzenia (prolongacji) $\mathbf{R}_l^T$. Operator zawężenia $\mathbf{R}_l$ wybiera spośród wszystkich składowych wektora $\mathbf{u}$ podwektor stopni swobody $\mathbf{u}_l$ (w klasycznych metodach Schwarza odpowiadający podobszarowi Ω_l). Operator rozszerzenia $\mathbf{R}_l^T$ uzupełnia podwektor stopni swobody $\mathbf{u}_l$ zerami, do pełnego wymiaru wektora $\mathbf{u}$. W zapisie macierzowym $\mathbf{R}_l$ jest tablicą złożoną z jedynek i zer, a $\mathbf{R}_l^T$ jest, zgodnie z oznaczeniem, transpozycją $\mathbf{R}_l$, przy czym $\mathbf{R}_l\mathbf{R}_l^T = \mathbf{I}$.

Wykorzystując powyższą notację, lokalne problemy (2.56) dają się zapisać jako:

$$\begin{aligned}\mathbf{R}_l\mathbf{u}^{m,l} &= -\mathbf{A}_{ll}^{-1}\left(\sum_{k=1,k\neq l}^{N_{\mathrm{bl}}}\mathbf{A}_{lk}\mathbf{R}_k\tilde{\mathbf{u}} - \mathbf{R}_l\mathbf{b}\right) = \\ &= \mathbf{R}_l\tilde{\mathbf{u}} - \mathbf{A}_{ll}^{-1}\sum_{k=1}^{N_{\mathrm{bl}}}\mathbf{A}_{lk}\mathbf{R}_k\tilde{\mathbf{u}} + \mathbf{A}_{ll}^{-1}\mathbf{R}_l\mathbf{b}\end{aligned} \tag{2.61}$$

gdzie w celu uzyskania ostatniej równości zastosowano tożsamość:

$$\mathbf{R}_l\tilde{\mathbf{u}} = \mathbf{A}_{ll}^{-1}\mathbf{A}_{ll}\mathbf{R}_l\tilde{\mathbf{u}} \tag{2.62}$$

Ostateczny wektorowy zapis problemu (2.56) w notacji metod dekompozycji obszaru wygląda następująco:

$$\mathbf{u}^{m,l} = \tilde{\mathbf{u}} + \mathbf{R}_l^T\mathbf{A}_{ll}^{-1}\mathbf{R}_l(\mathbf{b}-\mathbf{A}\tilde{\mathbf{u}}) \tag{2.63}$$

W konsekwencji, pojedyncza iteracja addytywnej metody Schwarza, będącej uogólnieniem blokowej metody Jacobiego, uzyskuje formę:

$$\mathbf{u}^m = \mathbf{u}^{m-1} - \sum_{l=1}^{N_{\text{bl}}} \mathbf{R}_l^T \mathbf{A}_{ll}^{-1} \mathbf{R}_l \left(\mathbf{A}\mathbf{u}^{m-1} - \mathbf{b} \right) \tag{2.64}$$

natomiast pojedyncza iteracja multiplikatywnej metody Schwarza, będącej uogólnieniem blokowej metody Gaussa-Seidla, przybiera postać:

$$\mathbf{u}^m = \mathbf{u}^{m-1} - \left(I - \prod_{l=1}^{N_{\text{bl}}} \left(I - \mathbf{R}_l^T \mathbf{A}_{ll}^{-1} \mathbf{R}_l \mathbf{A} \right) \right) \mathbf{A}^{-1} \left(\mathbf{A}\mathbf{u}^{m-1} - \mathbf{b} \right) \tag{2.65}$$

Kluczem do analizy metod dekompozycji obszaru jest pojęcie rzutowania błędu aktualnego rozwiązania na lokalną przestrzeń (wektorową lub funkcyjną) związaną z podobszarem (podwektorem stopni swobody $\mathbf{u}_l$). Operator projekcji na lokalną przestrzeń wektorową definiuje się wzorem:

$$\mathbf{P}_l = \mathbf{R}_l^T \mathbf{A}_{ll}^{-1} \mathbf{R}_l \mathbf{A} \tag{2.66}$$

Jeśli przez $\mathbf{u}^*$ oznaczone zostanie rozwiązanie dokładne układu równań, tak że $\mathbf{b} = \mathbf{A} \cdot \mathbf{u}^*$, to lokalny problem (2.63) okazuje się być korektą aktualnego rozwiązania $\tilde{\mathbf{u}}$ o zrzutowany na lokalną przestrzeń globalny błąd rozwiązania $\tilde{\mathbf{e}}$:

$$\mathbf{u}^{m,l} = \tilde{\mathbf{u}} + \mathbf{P}_l(\mathbf{u}^* - \tilde{\mathbf{u}}) = \tilde{\mathbf{u}} + \mathbf{P}_l \tilde{\mathbf{e}} \tag{2.67}$$

W dalszych rozważaniach proces iteracyjny zapisywany jest w języku analizy funkcjonalnej. Słabe sformułowanie (2.30) lub (2.31) związane z rozwiązywanym układem równań, po uproszczeniu zapisu przez pominięcie indeksów konkretnych metod, przybiera formę:

Znajdź taką funkcję $u \in V_h$, aby spełnione było:

$$a(u, v) = l(v) \qquad \forall v \in V_h \tag{2.68}$$

Rozważa się dekompozycję obszaru obliczeniowego Ω na nakładające się podobszary Ω_l i lokalne problemy projekcji na przestrzenie funkcyjne $V_l \subset V_h$, związane z podobszarami Ω_l. Dla każdego podobszaru przestrzeń V_l jest rozpięta na lokalnych dla Ω_l funkcjach bazowych zerujących się na brzegu wewnętrznym podobszaru[20]. Dla dowolnej funkcji $u_l \in V_l$ określa się jej rozszerzenie u_l' do pełnej przestrzeni V_h (poprzez funkcję zerową). Dla dowolnej

[20]Brzegiem wewnętrznym podobszaru jest ta część jego brzegu, która nie pokrywa się z brzegiem obszaru obliczeniowego.

funkcji g problem rzutowania związany ze sformułowaniem (2.68) i rozwiązywanym układem równań ma postać:

Znajdź taką funkcję $g_l \in V_l$, aby spełnione było:

$$a_l(g_l, v_l) = a(g, v_l') \qquad \forall v_l \in V_l \tag{2.69}$$

Lokalną formę dwuliniową $a_l(.,.)$, odpowiadającą lokalnym problemom (2.56), uzyskuje się z formy globalnej poprzez zastąpienie obszaru Ω podobszarem Ω_l.

Jeśli funkcję z przestrzeni V_l odpowiadającą aktualnemu rozwiązaniu $\tilde{\mathbf{u}}$ oznaczymy przez $\tilde{u}$, a funkcję odpowiadającą rozwiązaniu układu $\mathbf{u}^*$ przez u^*, to dla funkcji błędu $\tilde{e} = u^* - \tilde{u}$ zachodzi równość:

$$a(\tilde{e}, v_l') = a(u^* - \tilde{u}, v_l') = l(v_l') - a(\tilde{u}, v_l') \tag{2.70}$$

Zagadnienie lokalnej projekcji globalnego błędu rozwiązania, zapisywane dla wektorów jako:

$$\tilde{\mathbf{e}}_l = \mathbf{P}_l \tilde{\mathbf{e}} \tag{2.71}$$

dla odpowiadających funkcji ma ostatecznie postać:

Znajdź taką funkcję $\tilde{e}_l \in V_l$, aby spełnione było:

$$a_l(\tilde{e}_l, v_l) = l(v_l') - a(\tilde{u}, v_l') \qquad \forall v_l \in V_l \tag{2.72}$$

Podobnie jak dla wektora niewiadomych, także dla odpowiadającej mu funkcji lokalne problemy przyjmują postać korekty aktualnych wartości za pomocą błędu zrzutowanego na lokalną przestrzeń:

$$u^{m,l} = \tilde{u} + \tilde{e}_l \tag{2.73}$$

Zinterpretowane jako metody dekompozycji obszaru klasyczne metody blokowe Jacobiego i Gaussa-Seidla ukazują się jako serie lokalnych projekcji błędu i następujących (w różnych momentach dla różnych metod) korekt aktualnego rozwiązania. Istotnym elementem w rozważanych metodach dekompozycji (metodach Schwarza) jest nakładanie się podobszarów, z którymi związane są lokalne podprzestrzenie. Zbieżność iteracji jest w tych metodach zależna od rozmiaru podobszarów, ich kształtu, rozmiaru strefy nakładania się podobszarów i, oczywiście, od własności problemu [67, 66, 68, 145]. W standardowych przypadkach, im większe podobszary i im większa strefa nakładania się podobszarów, tym zbieżność metod szybsza. Zbieżność ta jest jednak na tyle wolna, że w praktycznych zastosowaniach zawsze stosuje się techniki jej przyspieszania.

2.7.2 Metody dwusiatkowe

Jedną z przyczyn wolnej zbieżności metod Jacobiego i Gaussa-Seidla jest fakt, że lokalne projekcje błędu pozwalają tylko na lokalną poprawę rozwiązania, która jest związana ze składnikami wysokiej częstotliwości w falowym rozkładzie błędu [162]. Szczególnie widoczne jest to dla silnie rozdrobnionych siatek, dla których rozmiar podobszarów jest mały w stosunku do rozmiaru całego obszaru. Lokalne korekty pozwalają na efektywne wygładzenie błędu, nie redukują jednak jego składowych o niskiej częstotliwości.

Aby zredukować także wolnozmienne składowe błędu, konstruuje się dodatkową, zgrubną korektę błędu. W klasycznej teorii metod dekompozycji obszaru rozważa się pojedynczą korektę związaną ze zgrubną przestrzenią V_C, obejmującą cały obszar obliczeniowy, ale o znacząco mniejszej liczbie stopni swobody niż oryginalna przestrzeń V_h. Przestrzeń V_C najczęściej jest uzyskiwana poprzez aproksymację zadania w obszarze obliczeniowym przy użyciu nowej, rzadszej („zgrubnej") siatki, o znacząco większym rozmiarze elementów. Otrzymana w ten sposób metoda jest metodą dwusiatkową (*two-grid*), którą uogólnia się następnie do metod wielosiatkowych (*multigrid*).

W metodzie dwusiatkowej definiuje się dla przestrzeni V_C, podobnie jak wcześniej dla przestrzeni V_l, zagadnienie projekcji dowolnej funkcji g:

Znajdź taką funkcję $g_C \in V_C$, aby spełnione było:

$$a(g_C, v_C) = a(g, v_C) \qquad \forall v_C \in V_C \tag{2.74}$$

(oznaczenia są analogiczne do zdefiniowanych dla przestrzeni V_l). Zgrubna korekta błędu $\tilde{e}_C$ dla aktualnego rozwiązania $\tilde{u}$ obliczana jest, podobnie jak lokalna korekta $\tilde{e}_l$ dla V_l, jako rozwiązanie problemu:

Znajdź taką funkcję $\tilde{e}_C \in V_C$, aby spełnione było:

$$a(\tilde{e}_C, v_C) = l(v_C) - a(\tilde{u}, v_C) \qquad \forall v_C \in V_C \tag{2.75}$$

Uzyskana w ten sposób korekta $\tilde{e}_C$ użyta jest do poprawy aktualnego rozwiązania, ponownie w analogiczny sposób jak korekty lokalne $\tilde{e}_l$. Aktualne rozwiązanie, które jest argumentem problemu (2.75) i które podlega korekcie, może być związane z początkowym wektorem iteracji, tym samym, na którym operują lokalne korekty. Mamy wtedy do czynienia z addytywną wersją metody. Jeśli rozwiązaniem aktualnym jest funkcja po naniesieniu lokalnych poprawek $\tilde{e}_l$, metoda jest multiplikatywna.

Wyrażany dotychczas w notacji funkcyjnej problem obliczenia zgrubnej korekty błędu może być także zapisany w notacji wektorowej:

$$\tilde{\mathbf{e}}_C = \mathbf{R}_C^T \mathbf{A}_C^{-1} \mathbf{R}_C (\mathbf{b} - \mathbf{A}\tilde{\mathbf{u}}) \tag{2.76}$$

Dwusiatkowa metoda addytywna ma postać:

$$\mathbf{u}_C^m = \mathbf{u}^m + \mathbf{R}_C^T \mathbf{A}_C^{-1} \mathbf{R}_C (\mathbf{b} - \mathbf{A}\mathbf{u}^{m-1}) \tag{2.77}$$

a metodę multiplikatywną można zapisać jako:

$$\mathbf{u}_C^m = \mathbf{u}^m + \mathbf{R}_C^T \mathbf{A}_C^{-1} \mathbf{R}_C (\mathbf{b} - \mathbf{A}\mathbf{u}^m) \tag{2.78}$$

gdzie: $\mathbf{u}^{m-1}$ – rozwiązanie początkowe dla iteracji, $\mathbf{u}^m$ – rozwiązanie po serii poprawek lokalnych (wzory (2.64) i (2.65)), a $\mathbf{u}_C^m$ – rozwiązanie po serii poprawek lokalnych i poprawce zgrubnej.

2.7.3 Metody wielosiatkowe

Uogólnieniem metod dwusiatkowych są metody wielosiatkowe. Rozważa się w nich sekwencję przestrzeni V_{C_i}, $C_i = C_0, C_1, ..., C_{\max}$, gdzie $V_{C_{\max}}$ odpowiada przestrzeni V_h. Ponownie najczęstszą realizacją metod wielosiatkowych jest użycie ciągu siatek o coraz większym rozmiarze elementów, pokrywających cały obszar obliczeniowy. Każda przestrzeń V_{C_i} jest rozpinana na funkcjach bazowych odpowiadających kolejnej siatce. W utworzonej hierarchii siatek na poziomie najwyższym znajduje się siatka ostateczna, odpowiadająca przestrzeni V_h, następnie kolejne, rzadsze siatki na niższych poziomach aż do siatki najrzadszej na poziomie najniższym. Dla każdej pary przestrzeni V_{C_i} i $V_{C_{i-1}}$ siatka odpowiadająca przestrzeni V_{C_i} będzie nazywana siatką gestą, a siatka odpowiadająca przestrzeni zgrubnej $V_{C_{i-1}}$ siatką rzadką.

Dla każdej pary przestrzeni V_{C_i} i $V_{C_{i-1}}$ rozważa się algorytm wygładzania błędu w przestrzeni V_{C_i} (na siatce gęstej) i algorytm korekty błędu w przestrzeni zgrubnej $V_{C_{i-1}}$. Obliczenia korekty zgrubnej (2.76) nie są przeprowadzane dokładnie. Dla zbieżności metody wystarczy, aby w miejsce macierzy $\mathbf{A}_{C_i}^{-1}$ zastosować operator liniowy będący jej przybliżeniem. Tym przybliżeniem może być ponowne wygładzenie błędu, tym razem w przestrzeni zgrubnej (na siatce rzadkiej) i korekta błędu w kolejnej w ciągu przestrzeni $V_{C_{i-2}}$. Jedynym warunkiem jest, aby obie operacje odpowiadały operatorom liniowym. Schemat wygładzania i korekty można powtarzać aż do osiągnięcia ostatniej przestrzeni V_{C_0}. Tutaj rozwiązanie zadania korekty zgrubnej musi już być dokładne.

Tak opisany proces prowadzi do klasycznego cyklu V metody wielosiatkowej [84, 176, 45, 118]. Dodatkowo rozważa się w nim wygładzanie błędu także po dodaniu korekty z przestrzeni zgrubnej. Algorytmem odpowiadającym cyklowi V metody wielosiatkowej jest rekurencyjny algorytm MG:

Algorytm 2.1 (MG)

```
MG(y, d, C_i, N_pre, N_post)
{
  jeżeli (C_i = C_0) {
    rozwiąż dokładnie zadanie dla siatki najrzadszej:  /
    y := A_{C_0}^{-1} d
  } w pozostałych przypadkach {
    dla j = 1, 2, ..., N_pre {
      przeprowadz wygładzanie wstępne:  /
      y := SM(y, d, C_i)
    }
    oblicz residuum na siatce gęstej i zrzutuj na rzadką:  /
    g := R_{C_i}(d - A_{C_i} y)
    zainicjuj rozwiązanie początkowe dla problemu korekty:  /
    z := 0
    znajdz przybliżone rozwiązanie problemu korekty na /
    siatce rzadkiej za pomocą wielosiatkowego cyklu V: /
    z := MG(z, g, C_{i-1}, N_pre, N_post)
    popraw aktualne rozwiązanie:  /
    y := y + R_{C_i}^T z
    dla j = 1, 2, ..., N_post {
      przeprowadz wygładzanie koncowe:  /
      y := SM(y, d, C_i)
} } }
```

Argumentami algorytmu MG są pewien wektor początkowy $\mathbf{y}$ i wektor prawej strony rozwiązywanego układu $\mathbf{d}$, określone dla przestrzeni V_{C_i}. $\mathbf{R}_{C_i}$ i $\mathbf{R}_{C_i}^T$ oznaczają operatory zawężenia z przestrzeni V_{C_i} do przestrzeni $V_{C_{i-1}}$ i rozszerzenia z $V_{C_{i-1}}$ do V_{C_i}. Operacja $\mathtt{SM}(\mathbf{y}, \mathbf{d}, C_i)$ odpowiada wygładzaniu błędu w przestrzeni V_{C_i}. Stosownie dobierane parametry N_{pre} i N_{post} określają liczbę iteracji wygładzających przed korektą i po korekcie w przestrzeni zgrubnej.

Jeżeli do wygładzania błędu użyta jest sekwencja lokalnych projekcji (czyli pojedyncza iteracja addytywnej lub multiplikatywnej metody Schwarza), to operacja wygładzania odpowiada wzorom (2.64) i (2.65). Zastosowanie algorytmu SM dla uzyskania kolejnej m-tej iteracji globalnego rozwiązania $\mathbf{u}$ daje się wtedy zapisać jako:

$$\mathbf{u}^m = \mathtt{SM}(\mathbf{u}^{m-1}, \mathbf{b}, C_{\max}) = \mathbf{u}^{m-1} + \mathbf{M}_{\text{SM}}^{-1}\left(\mathbf{b} - \mathbf{A}\mathbf{u}^{m-1}\right) \tag{2.79}$$

gdzie $\mathbf{M}_{\mathrm{SM}}^{-1}$ oznacza liniowy operator implikowany przez algorytm wygładzania na siatce ostatecznej.

Wywołanie algorytmu `MG` dla uzyskania m-tej iteracji $\mathbf{u}$ można wyrazić jako:

$$\mathbf{u}^m = \mathtt{MG}(\mathbf{u}^{m-1}, \mathbf{b}, C_{\max}, N_{\mathrm{pre}}, N_{\mathrm{post}}) = \mathbf{u}^{m-1} + \mathbf{M}_{\mathrm{MG}}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}^{m-1}) \quad (2.80)$$

gdzie tym razem $\mathbf{M}_{\mathrm{MG}}^{-1}$ jest operatorem liniowym odpowiadającym wygładzaniu na wszystkich siatkach wyższych poziomów i korekcie na siatce najrzadszej.

Algorytm `MG` może być stosowany jako solwer układów równań liniowych o znacznie szybszej zbieżności niż pojedyncze iteracje proste. W punkcie 2.7.5 opisane są techniki dalszego przyspieszania zbieżności obu tych metod.

2.7.4 Realizacja metod wielosiatkowych dla adaptacyjnej MES

Realizacja konkretnego algorytmu wielosiatkowego polega na podaniu dla każdej pary indeksów C_i i C_{i-1}, $C_i = C_1, ..., C_{\max}$, odpowiadających parze siatek – gęstej i rzadkiej, następujących definicji:

- przestrzeni zgrubnej $V_{C_{i-1}}$ ($V_{C_{\max}} = V_h$ jest dane),
- operatora zawężenia $\mathbf{R}_{C_i}$,
- operatora rozszerzenia $\mathbf{R}_{C_i}^T$,
- macierzy $\mathbf{A}_{C_{i-1}}$ ($\mathbf{A}_{C_{\max}} = \mathbf{A}$ jest dane).

W opisywanym w niniejszej pracy solwerze iteracyjnym definicje $V_{C_{i-1}}$, $\mathbf{R}_{C_i}$, $\mathbf{R}_{C_i}^T$ i $\mathbf{A}_{C_{i-1}}$ związane są z adaptacjami siatki MES.

Przestrzenie zgrubne $V_{C_{i-1}}$

Założeniem opisywanej realizacji jest, że każda przestrzeń funkcyjna V_{C_i}, $C_i = C_0, C_1, ..., C_{\max}$, jest rozpięta na funkcjach bazowych związanych z pewną siatką MES. Definicja sekwencji przestrzeni dla metody wielosiatkowej polega na podaniu dla każdej przestrzeni V_{C_i} i związanej z nią siatki gęstej sposobu konstrukcji siatki rzadkiej i związanej z nią przestrzeni zgrubnej $V_{C_{i-1}}$.

Zakłada się, że siatka gęsta jest otrzymywana w efekcie jedno- lub kilkakrotnego zastosowania procedury adaptacji typu h lub hp, polegającej m.in. na podziałach elementów. Efektem podziału pojedynczego elementu (elementu-rodzica) jest kilka elementów o mniejszych rozmiarach (elementów-dzieci). Funkcje bazowe związane z elementami-dziećmi rozpinają pewną przestrzeń

funkcji. Dla tej przestrzeni, przestrzeń funkcji rozpiętych przez funkcje bazowe związane z elementem-rodzicem jest przestrzenią zgrubną[21]. W całej siatce gęstej, jeśli powstała w wyniku wielokrotnych adaptacji, mogą występować elementy siatki początkowej oraz elementy powstałe w wyniku podziału innych elementów. Tworząc siatkę rzadką, postępuje sie następująco:

- każdy element siatki początkowej występujący w siatce gęstej pozostawia się w siatce rzadkiej,
- grupę elementów z siatki gęstej będących dziećmi pojedynczego elementu-rodzica zastępuje się w siatce rzadkiej tymże rodzicem.

Operatory $\mathbf{R}_{C_i}$ i $\mathbf{R}^T_{C_i}$

Konstrukcja operatora zawężenia wykorzystuje zależności pomiędzy funkcjami bazowymi $\phi^i_{C_{i-1}}$[22], związanymi z elementem-rodzicem, i funkcjami bazowymi $\phi^i_{C_i}$, związanymi z jego dziećmi. Każda zgrubna funkcja bazowa $\phi^k_{C_{i-1}}$ (funkcja bazowa na siatce rzadkiej) może zostać przedstawiona jako kombinacja liniowa funkcji $\phi^j_{C_i}$:

$$\phi^k_{C_{i-1}} = \sum_j R^{kj}_{C_i} \phi^j_{C_i} \tag{2.81}$$

W standardowy sposób elementy przestrzeni V_{C_i} i $V_{C_{i-1}}$ reprezentowane są jako kombinacje liniowe funkcji bazowych:

$$v_{C_i} = \sum_{l=1}^{N_{C_i}} v^l_{C_i} \phi^l_{C_i} \qquad v_{C_{i-1}} = \sum_{k=1}^{N_{C_{i-1}}} v^k_{C_{i-1}} \phi^k_{C_{i-1}} \tag{2.82}$$

Po zastosowaniu powyższych podstawień problem korekty na siatce rzadkiej, analogiczny do (2.75), przekształca się do postaci:

Znajdź taką funkcję $\tilde{e}_{C_{i-1}} \in V_{C_{i-1}}$, aby dla każdego $v_{C_{i-1}} \in V_{C_{i-1}}$ spełnione było:

$$a(\tilde{e}_{C_{i-1}}, v_{C_{i-1}}) = \sum_{k=1}^{N_{C_{i-1}}} v^k_{C_{i-1}} \sum_j R^{kj}_{C_i} \left(l(\phi^j_{C_i}) - \sum_{l=1}^{N_{C_i}} \tilde{u}^l_{C_i} a(\phi^l_{C_i}, \phi^j_{C_i}) \right) \tag{2.83}$$

[21] Tak zdefiniowana sekwencja zgrubnych przestrzeni odpowiada klasycznej geometrycznej metodzie wielosiatkowej [84].

[22] W punkcie tym indeksy związane z poszczególnymi funkcjami bazowymi i składowymi wektora niewiadomych umieszczane są u góry, dla uproszczenia notacji w przypadku występowania siatek gęstych i rzadkich.

Obliczenie prawej strony w powyższym równaniu odpowiada wyliczeniu residuum dla aktualnego rozwiązania na siatce gęstej:

$$l(\phi^j_{C_i}) - \sum_{l=1}^{N_{C_i}} \tilde{u}^l_{C_i} a(\phi^l_{C_i}, \phi^j_{C_i}) = b^j_{C_i} - \sum_{l=1}^{N_{C_i}} A^{j,l}_{C_i} \tilde{u}^l_{C_i} \tag{2.84}$$

i zastosowaniu zawężenia do przestrzeni siatki rzadkiej. Współczynniki $R^{kj}_{C_i}$ uzyskane lokalnie dla elementów-rodziców i ich dzieci stają się wyrazami operatorów restrykcji $\mathbf{R}_{C_i}$.

Klasyczne sformułowania metod wielosiatkowych dopuszczają odrębną konstrukcję operatorów restrykcji (zawężenia) i prolongacji (rozszerzenia). W opracowanym modelu implementacji kodu ta dowolność jest zachowana. W modelu przyjętym do analizy zakłada się, że, zgodnie z notacją, operator rozszerzenia jest transpozycją operatora zawężenia.

Macierze $\mathbf{A}_{C_{i-1}}$

Zastosowanie podstawień (2.82) do lewej strony równania (2.83) daje wyrażenie:

$$\sum_{k=1}^{N_{C_{i-1}}} v^k_{C_{i-1}} \sum_{l=1}^{N_{C_{i-1}}} \tilde{e}^l_{C_{i-1}} a(\phi^l_{C_{i-1}}, \phi^k_{C_{i-1}}) \tag{2.85}$$

Wyrazy $a(\phi^l_{C_{i-1}}, \phi^k_{C_{i-1}})$ macierzy $\mathbf{A}_{C_{i-1}}$ można obliczać z oryginalnego sformułowania słabego (2.68), uwzględniając odpowiednie obszary całkowania i funkcje kształtu. Można także wykorzystać wzory (2.81) oraz dwuliniowość formy a i uzyskać macierze $\mathbf{A}_{C_{i-1}}$ za pomocą operatorów zawężenia oraz rozszerzenia (operacja ta jest nazywana projekcją Galerkina – *Galerkin projection*):

$$\mathbf{A}_{C_{i-1}} = \mathbf{R}_{C_i} \mathbf{A}_{C_i} \mathbf{R}^T_{C_i} \tag{2.86}$$

2.7.5 Metody podprzestrzeni Kryłowa z poprawą uwarunkowania macierzy układu

Metody podprzestrzeni Kryłowa (zwane często w skrócie metodami Kryłowa) są obecnie jednymi z najpopularniejszych metod iteracyjnego rozwiązywania układów równań liniowych [35]. Ich istota polega na poszukiwaniu przybliżenia rozwiązania w specjalnych podprzestrzeniach wektorowych związanych z rozważanym układem równań. Zbieżność metod podprzestrzeni Kryłowa jest tym szybsza, im uwarunkowanie macierzy układu jest bliższe jedności [154]. Dlatego w praktyce prawie wyłącznie stosuje się metody podprzestrzeni Kryłowa w

połączeniu z poprawą uwarunkowania macierzy układu. Uwarunkowanie macierzy $\mathbf{A}$ definiuje się jako iloczyn $\|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, który osiąga optymalną wartość równą 1 dla macierzy jednostkowej. W przypadku tzw. lewostronnej poprawy uwarunkowania zamiast oryginalnego układu (2.41) rozwiązuje się układ:

$$\mathbf{M}^{-1}\mathbf{A} \cdot \mathbf{u} = \mathbf{M}^{-1}\mathbf{b} \tag{2.87}$$

Macierz poprawy uwarunkowania $\mathbf{M}^{-1}$ dobiera się tak, aby iloczyn $\mathbf{M}^{-1}\mathbf{A}$ był bliski macierzy jednostkowej. Można to osiągnąć przez przybliżone rozwiązanie oryginalnego układu (2.41) za pomocą metody, którą da się przedstawić w postaci operatora liniowego. W takim, typowym w praktyce, przypadku nie konstruuje się operatora $\mathbf{M}^{-1}$, a jego postać jest implikowana przez stosowany algorytm.

Dla metod podprzestrzeni Kryłowa zastosowanych do zmodyfikowanego układu (2.87) ciąg podprzestrzeni, dla których poszukuje się rozwiązania przybliżonego, jest dany jako:

$$\mathbf{u}_0 + \mathbf{K}_{N_\mathrm{k}} \tag{2.88}$$

gdzie $\mathbf{u}_0$ jest dowolnym wektorem początkowym. Oznaczając przez $\mathbf{r}_0$ początkowe residuum układu równań:

$$\mathbf{r}_0 = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}_0) \tag{2.89}$$

podprzestrzeń Kryłowa $\mathbf{K}_{N_\mathrm{k}}$ o wymiarze N_k definiuje się jako:

$$\mathbf{K}_{N_\mathrm{k}} = \mathrm{span}\left\{\mathbf{r}_0, \mathbf{M}^{-1}\mathbf{A}\mathbf{r}_0, (\mathbf{M}^{-1}\mathbf{A})^2\mathbf{r}_0, ..., (\mathbf{M}^{-1}\mathbf{A})^{(N_\mathrm{k}-1)}\mathbf{r}_0\right\} \tag{2.90}$$

Algorytmy poszczególnych metod Kryłowa polegają na znalezieniu, w kolejno tworzonych podprzestrzeniach $\mathbf{K}_{N_\mathrm{k}}$, najlepszego, w sensie wybranej miary, przybliżenia dla rozwiązania dokładnego $\mathbf{u}^*$. Dobór miary przybliżenia i dobór strategii poszukiwań decydują o różnicach między metodami.

Metoda sprzężonych gradientów

Najpopularniejszą z metod podprzestrzeni Kryłowa jest metoda sprzężonych gradientów, mająca zastosowanie do problemów symetrycznych i dodatnio określonych [89]. Dla tej grupy problemów rozwiązanie układu równań liniowych jest równoważne minimalizacji pewnego funkcjonału. Dzięki temu w metodzie sprzężonych gradientów poszukiwanie przybliżenia w podprzestrzeniach Kryłowa sprowadza się do ciągu problemów minimalizacji błędu wzdłuż

wybranych kierunków $\mathbf{p}_k$. Wektory wyznaczające kolejne kierunki $\mathbf{p}_k$, rozpinające podprzestrzeń $\mathbf{K}_k$, są z sobą sprzężone, $\mathbf{p}_k \cdot \mathbf{A} \cdot \mathbf{p}_j = 0$, $j \neq k$, dzięki czemu mogą być obliczane z wzorów rekurencyjnych na podstawie znajomości poprzedniego kierunku poszukiwań. Metoda minimalizuje błąd $\|\mathbf{u}^* - \mathbf{u}_k\|$, gdzie $\mathbf{u}_k$ jest przybliżeniem po k-tej iteracji, należącym do $\mathbf{K}_k$. Ostateczny algorytm przedstawiony jest w następującej definicji:

Algorytm 2.2 (`CG`)

`CG(`$\mathbf{u}_0$`, u,` tol`)`
`{`
  `oblicz residuum oryginalnego układu rownan:` $\tilde{\mathbf{r}}_0 := \mathbf{A}\mathbf{u}_0 - \mathbf{b}$
  `oblicz residuum zmodyfikowanego układu rownan:` $\mathbf{r}_0 := \mathbf{M}^{-1}\tilde{\mathbf{r}}_0$
  `ustal początkowy kierunek:` $\mathbf{p}_0 := \mathbf{r}_0$
  `dla` $k = 0, 1, \ldots.$ `{`
    `oblicz współczynnik poprawki:` $\alpha_k := -\mathbf{r}_k \cdot \tilde{\mathbf{r}}_k / \mathbf{p}_k \cdot \mathbf{A} \cdot \mathbf{p}_k$
    `popraw rozwiązanie:` $\mathbf{u}_{k+1} := \mathbf{u}_k - \alpha_k \mathbf{p}_k$
    `popraw residuum:` $\tilde{\mathbf{r}}_{k+1} := \tilde{\mathbf{r}}_k + \alpha_k \mathbf{A}\mathbf{p}_k$
    `jeżeli (`$\alpha_k \|\mathbf{p}\| <$ tol`) {`
      `podstaw:` $\mathbf{u} := \mathbf{u}_{k+1}$
      `wroc do programu głownego`
    `}`
    `oblicz residuum układu zmodyfikowanego:` $\mathbf{r}_{k+1} := \mathbf{M}^{-1}\tilde{\mathbf{r}}_{k+1}$
    `oblicz poprawkę kierunku poszukiwan:` $\beta := \mathbf{r}_{k+1} \cdot \tilde{\mathbf{r}}_{k+1} / \mathbf{r}_k \cdot \tilde{\mathbf{r}}_k$
    `znajdz nowy kierunek:` $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta \mathbf{p}_k$
`} }`

Zatrzymanie iteracji następuje w momencie, gdy norma poprawki aktualnego rozwiązania jest mniejsza od pewnej założonej tolerancji tol, będącej argumentem procedury `CG`.

Metoda GMRES

Metoda GMRES (*Generalized Minimal RESidual* [155]) ma zastosowanie do rozwiązywania niesymetrycznych, dodatnio określonych układów równań. Ze względu na brak symetrii układom takim nie odpowiada problem minimalizacyjny, w związku z czym kolejne kierunki poszukiwań nie są z sobą sprzężone i nie da się obliczać ich rekurencyjnie. Stąd bazę podprzestrzeni Kryłowa konstruuje się jawnie za pomocą zmodyfikowanej procedury ortonormalizacji Grama-Schmidta.

Poniższa definicja przedstawia algorytm tzw. restartowanej metody GMRES z poprawą uwarunkowania macierzy:

Algorytm 2.3 (GMRES)

```
GMRES(u_0, u, tol)
{
  powtarzaj {
    oblicz początkowe residuum układu:  r_0 := M^{-1}(b - A u_0)
    znormalizuj residuum:  r̄_0 := r_0/||r_0||
    dla i = 1,2,...,N_k {
      oblicz iloczyn:  r̂_i := M^{-1} A r̄_{i-1}
      zortonormalizuj r̂_i względem r̄_j, j=0,1,..,i-1 za pomocą /
      zmodyfikowanej procedury Grama-Schmidta:  {
        dla j = 0,1,...,i-1 {
          H_ji := r̂_i · r̄_j  ;  r̂_i := r̂_i - H_ji r̄_j
        }
        H_ii := ||r̂_i||
        jeżeli (H_ii ≠ 0) podstaw:  r̄_i := r̂_i/H_ii
      }
      rozwiąż problem minimalizacyjny GMRES: min_{y∈R^i} ||βe_1 - Hy||
      jeżeli (||M^{-1}(b - A(u_0 + z))|| < tol lub i = N_k lub H_ii = 0) {
        oblicz poprawkę z:  z := Σ_{j=0}^{i-1} y_j r̄_j
        popraw rozwiązanie:  u := u_0 + z
        jeżeli (i ≠ N_k) wroc do programu głownego
    } }
    podstaw nowy wektor początkowy:  u_0 := u
} }
```

Algorytm GMRES polega na znalezieniu poprawki $\mathbf{z}$ rozwiązania $\mathbf{u}$, będącej kombinacją liniową uzyskanych wektorów bazowych $\bar{\mathbf{r}}_j$. Współczynniki tej kombinacji tworzą wektor $\mathbf{y}$, który minimalizuje wyrażenie $\|\beta\mathbf{e}_1 - \mathbf{H}\mathbf{y}\|$, $\beta = \|\mathbf{r}_0\|$, $\mathbf{e}_1 = (1,0,0,...,0)^T \in \mathbb{R}^{k+1}$. W wyrażeniu tym k jest aktualnym wymiarem podprzestrzeni Kryłowa, a $\mathbf{H}$ jest prostokątną macierzą zawierającą iloczyny skalarne postaci:

$$H_{ji} = \mathbf{M}^{-1}\mathbf{A}\bar{\mathbf{r}}_{i-1} \cdot \bar{\mathbf{r}}_j \quad i = 1,...,k, \;\; j = 0,...,k \tag{2.91}$$

Zagadnienie znalezienia wektora $\mathbf{y}$ nazywane jest problemem minimalizacyjnym GMRES. Na skutek własności wektorów bazowych $\bar{\mathbf{r}}_j$, użycie $\mathbf{y}$ do poprawy rozwiązania jest równoważne minimalizacji residuum układu równań dla danej podprzestrzeni Kryłowa. Aby sprawdzić spełnienie warunku:

$$\|\mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}(\mathbf{u}_0 + \mathbf{z}))\| < \text{tol} \tag{2.92}$$

nie jest konieczne obliczanie wyrażenia po lewej stronie. Jeśli do rozwiązania problemu minimalizacyjnego GMRES używa się ciągu obrotów Givensa,

to wyrażenie to jest uzyskiwane jako jeden z wyrazów przetransformowanego wektora prawej strony $\beta\mathbf{e}_1$.

W algorytmie restartowanej metody GMRES maksymalny wymiar podprzestrzeni, w której poszukuje się minimum residuum, jest ograniczony przez liczbę N_k i jeśli zbieżność nie zostaje osiągnięta dla tego wymiaru, rozpoczyna się konstrukcję nowej podprzestrzeni Kryłowa, zaczynając tym razem od ostatnio uzyskanego przybliżenia, jako wektora początkowego. Zastosowanie tego wariantu ma na celu redukcję wysokiego kosztu zapamiętania wszystkich wektorów bazowych podprzestrzeni Kryłowa, koniecznych do realizacji procedury Grama-Schmidta.

Jako miary szybkości zbieżności metody GMRES używa się ilorazu norm residuów dla dwóch kolejnych iteracji, $\|\hat{\mathbf{r}}_i\|/\|\hat{\mathbf{r}}_{i-1}\|$, lub tegoż ilorazu uśrednionego dla N_it iteracji:

$$(\|\hat{\mathbf{r}}_{N_\mathrm{it}}\|/\|\hat{\mathbf{r}}_0\|)^{1/N_\mathrm{it}} \tag{2.93}$$

Realizacja poprawy uwarunkowania macierzy układu w metodach podprzestrzeni Kryłowa

Do efektywnej realizacji metod podprzestrzeni Kryłowa z poprawą uwarunkowania macierzy układu konieczne jest wydajne wykonywanie operacji: obliczania normy wektora i iloczynu skalarnego wektorów, obliczania iloczynu macierzy układu $\mathbf{A}$ z dowolnym wektorem i przybliżonego rozwiązywania oryginalnego układu (2.41), operacji będącej odpowiednikiem iloczynu $\mathbf{M}^{-1}\mathbf{v}$ dla dowolnie wybranego wektora $\mathbf{v}$. Realizacja iloczynu $\mathbf{A}$ z dowolnym wektorem zależy od schematu przechowywania macierzy $\mathbf{A}$ i w większości przypadków dokonywana jest standardowymi technikami, podobnie jak operacje wektorowe. Kluczem do efektywności metod Kryłowa jest szybkie i gwarantujące dobre przybliżenie rozwiązania układu (2.41) wykonanie algorytmu odpowiadającego iloczynowi $\mathbf{M}^{-1}\mathbf{v}$.

Jedną z najpopularniejszych technik poprawy uwarunkowania macierzy układu w metodach Kryłowa jest wykorzystanie omawianych w pp. 2.7.1 i 2.7.3 metod jedno- lub wielosiatkowego wygładzania błędu. Wzory (2.79) i (2.80), przedstawiające efekty działania tych metod, mają wspólną postać:

$$\mathbf{u}^m = \mathbf{u}^{m-1} + \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{u}^{m-1}) \tag{2.94}$$

gdzie: $\mathbf{M}^{-1} = \mathbf{M}_\mathrm{MG}^{-1}$ lub $\mathbf{M}^{-1} = \mathbf{M}_\mathrm{SM}^{-1}$. W celu obliczenia iloczynu $\mathbf{M}^{-1}\mathbf{v}$ można wykonać jedną iterację (2.94) z wektorem $\mathbf{v}$, jako wektorem prawej strony $\mathbf{b}$, i zerowym wektorem początkowym $\mathbf{u}^{m-1}$.

Dla metody GMRES obie operacje macierzowe, obliczenie iloczynu $\mathbf{Av}$ i iloczynu $\mathbf{M}^{-1}\mathbf{v}$, występują łącznie albo pod postacią obliczenia residuum $\mathbf{M}^{-1}(\mathbf{b}-\mathbf{Au}_0)$, albo iloczynu $\mathbf{M}^{-1}\mathbf{A}\bar{\mathbf{r}}_{i-1}$. Dzięki temu można je zrealizować za pomocą pojedynczej iteracji (2.94), stosując odpowiednie podstawienia za wektor początkowy oraz wektor prawej strony i dokonując następnie odjęcia wektora początkowego $\mathbf{u}^{m-1}$ od wektora wynikowego $\mathbf{u}^m$ [110].

Algorytmy niekompletnego rozkładu macierzy układu

Podstawową wadą metod eliminacji Gaussa, zastosowanych do rozwiązywania układów równań liniowych MES, jest naruszenie, typowej dla MES, rzadkiej struktury macierzy układu. Metody eliminacji Gaussa wypełniają niezerowymi elementami pasmo wokół przekątnej głównej macierzy układu. W ogólnym przypadku (patrz p. 2.7.7), szerokość pasma rośnie wraz z liczbą stopni swobody, natomiast liczba niezerowych wyrazów w pojedynczym wierszu macierzy układu jest stała (patrz p. 2.5.5). Powoduje to stały wzrost proporcji wyrazów wypełnionych wartościami niezerowymi w trakcie działania algorytmu, w stosunku do wyrazów będących niezerowymi pierwotnie.

Modyfikacją standardowych algorytmów eliminacji są algorytmy, w których próbuje się kontrolować wyżej wymienioną proporcję. Algorytmy te należą do technik niekompletnego rozkładu, Choleskiego (*Incomplete Cholesky factorization*) dla przypadku macierzy symetrycznych [99] i LU (*Incomplete Lower-Upper factorization*) dla macierzy niesymetrycznych [50]. W przypadku najprostszych metod, takich które zachowują strukturę niezerowych wyrazów macierzy układu, wykonując algorytmy standardowych rozkładów pomija się te operacje, które dotyczą wyrazów zerowych. Otrzymany wynik jest przybliżeniem rozkładu ścisłego, a procedura rozwiązania z użyciem macierzy pochodzącej z niekompletnego rozkładu polega na znalezieniu pewnego przybliżenia odwrotności macierzy układu.

To właśnie jest potrzebne i wystarczające do realizacji tak algorytmów metod Kryłowa, jak i algorytmów metod wielosiatkowych [180]. W metodach Kryłowa niekompletna eliminacja stosowana jest do poprawy uwarunkowania macierzy, a więc obliczenia iloczynu $\mathbf{M}^{-1}\mathbf{v}$. Rozwiązanie z zastosowaniem niekompletnego rozkładu daje się zapisać w postaci operatora liniowego, który służy w tym wypadku do określenia macierzy $\mathbf{M}^{-1}$.

W przypadku zastosowania w algorytmie `MG` metody wielosiatkowej (s. 74), niekompletna eliminacja służy do realizacji algorytmu `SM` wygładzania błędu. Dla siatki każdego poziomu (z wyjątkiem siatki najrzadszej) rozwiązuje się w sposób przybliżony układ z macierzą $\mathbf{A}_{C_i}$, którą poddaje się niekompletnemu

rozkładowi, i z residuum $\mathbf{d} - \mathbf{A}_{C_i}\mathbf{y}$ jako wektorem prawej strony, a następnie wynik dodaje do wektora $\mathbf{y}$. Operatory liniowe wynikające z zastosowania niekompletnego rozkładu na danym poziomie siatki wykorzystywane są do definicji $\mathbf{M}_{\mathrm{MG}}^{-1}$ z wzoru (2.80). Metoda wielosiatkowa z wygładzaniem błędu stosującym niekompletny rozkład może stanowić niezależny solwer, może także zostać użyta do poprawy uwarunkowania macierzy dla dowolnej metody Kryłowa.

Wariantami metod niekompletnego rozkładu są algorytmy operujące na blokach macierzy układu, zamiast na pojedynczych jej wyrazach. Poniższa definicja przedstawia algorytm takiego właśnie blokowego wariantu niekompletnego rozkładu LU [154]:

Algorytm 2.4 (`ILU(0)`)

```
dla i = 1, ..., N_bl {
  dla k = 1, ..., i - 1 {
    jeżeli (A^E_{i,k} ≠ 0) {
      A^E_{i,k} := A^E_{i,k} · A^E_{k,k}
      dla j = k + 1, ..., N_bl {
        jeżeli (A^E_{k,j} ≠ 0) {
          A^E_{i,j} := A^E_{i,j} - A^E_{i,k} · A^E_{k,j}
  } } } }
  A^E_{i,i} := (A^E_{i,i})^{-1}
}
```

Bloki wykorzystywane w algorytmie są blokami elementarnymi macierzy $\mathbf{A}$. Algorytm działa w miejscu, zastępując bloki macierzy $\mathbf{A}$ blokami pochodzącymi z rozkładu. Bloki na przekątnej głównej zostają zamienione na ich odwrotności i następnie użyte w algorytmie. Algorytm, będący blokową wersją metody ILU(0), zachowuje strukturę wyrazów niezerowych oryginalnej macierzy $\mathbf{A}$. Cechą charakterystyczną zaprezentowanej wersji algorytmu (wykorzystującej tzw. realizację *ikj* eliminacji Gaussa) jest przeprowadzanie obliczeń w kolejnych poziomych pasmach macierzy $\mathbf{A}$. Wyrazy ponad pasmem są wykorzystywane, ale nie modyfikowane, natomiast wyrazy poniżej nie są wykorzystywane. Powoduje to uzależnienie ostatecznego wyniku od kolejności wykonywania operacji. Zależność ta może mieć istotny wpływ na zbieżność metody iteracyjnej używającej rozkładu ILU(0) [38].

Przy użyciu niekompletnej eliminacji Gaussa w ramach metod iteracyjnych, algorytm `ILU(0)` stosuje się jednokrotnie, a następnie wielokrotnie przeprowadza postępującą redukcję (*forward reduction*) i podstawienie powrotne (*back substitution*) dla kolejnych iteracji metod. Algorytm postępującej redukcji i

podstawienia powrotnego, podobnie jak algorytm metody ILU(0), jest modyfikowany, w porównaniu z wersją standardową, dla uwzględnienia rzadkiej struktury macierzy $\mathbf{A}$ i blokowego charakteru obliczeń. Przykład takiego algorytmu, ponownie wykorzystującego bloki elementarne macierzy $\mathbf{A}$, przedstawia poniższa definicja:

Algorytm 2.5 (FR-BS)

```
dla i = 1, ..., N_bl {
  u_i := b_i
  dla j = 1, ..., i - 1 {
    jeżeli (A^E_{i,j} ≠ 0) {
      u_i := u_i - A^E_{i,j} · u_j
} } }
dla i = N_bl, ..., 1;krok= -1 {
  dla j = i + 1, ..., N_bl {
    jeżeli (A^E_{i,j} ≠ 0) {
      u_i := u_i - A^E_{i,j} · u_j
  } }
  u_i := A^E_{i,i} · u_i
}
```

Algorytm powyższy działa dla macierzy pochodzącej z niekompletnego blokowego rozkładu ILU(0) dokonanego algorytmem z s. 83. Podobnie jak w algorytmie `ILU(0)` bloki na przekątnej głównej są odwrotnościami odpowiadających bloków oryginalnej macierzy $\mathbf{A}$.

Metody rozwiązywania równań liniowych bez tworzenia macierzy układu

Cechą metod podprzestrzeni Kryłowa, charakterystyczną także dla innych algorytmów iteracyjnych, jest występowanie macierzy $\mathbf{A}$ wyłącznie w ramach iloczynów ze specyficznymi dla danej metody wektorami. Umożliwia to opracowanie bezmacierzowych wariantów metod. Stosuje się je np. do rozwiązywania układów równań liniowych w ramach algorytmów przybliżonych metod Newtona (por. p. 2.3 i wzory (2.9), (2.11) i (2.12)). W dokładnej metodzie Newtona dla problemu (2.9) macierz układu równań liniowych $\mathbf{A}$ jest macierzą jakobianową $\partial\mathbf{F}/\partial\mathbf{u}$ nieliniowej funkcji $\mathbf{F}$ wektora niewiadomych $\mathbf{u}$. W przybliżonych wariantach bezmacierzowych, każdorazowe wystąpienie w algorytmie iloczynu $\mathbf{A}$ z dowolnym wektorem zamieniane jest na aproksymację (2.12), wykorzystującą wzór na pochodną kierunkową. Powoduje to, że macierz układu

nie tylko nie jest tworzona, ale nie jest w ogóle znana. W dalszych analizach ta nieznana macierz, odpowiadająca wykonywanym operacjom, oznaczana będzie przez $\mathbf{A}_J$ dla zaznaczenia, że jest ona pewnym przybliżeniem macierzy jakobianowej.

Także w przypadku wariantów bezmacierzowych istotne przyspieszenie zbieżności solwera liniowego uzyskuje się poprzez poprawę uwarunkowania macierzy układu. Jak jednak poprawić uwarunkowanie macierzy, której się nie tworzy? Najczęściej rozważa się pomocnicze przybliżenie macierzy jakobianowej, o którym zakłada się, że jest także przybliżeniem macierzy układu $\mathbf{A}_J$. Niekompletną dekompozycję takiej przybliżonej macierzy lub jej diagonalne bloki stosuje się w algorytmach poprawy uwarunkowania.

2.7.6 Bezmacierzowe rozwiązywanie układów równań liniowych w symulacjach przepływów ściśliwych

Algorytmy techniki bezmacierzowej dla aproksymacji równań Eulera

Omawiana w niniejszym punkcie technika jest uzupełnieniem metodologii zaprezentowanej w p. 2.3.4. Do aproksymacji stacjonarnych równań Eulera używana była tam technika całkowania pseudoczasowego i niejawny nieliniowy schemat Eulera. Na każdym kroku czasowym rozwiązywany był układ równań nieliniowych niedokładną (ograniczoną do jednej lub dwóch iteracji), przybliżoną (wykorzystującą w miejsce dokładnej przybliżoną macierz jakobianową) metodą Newtona. W każdej iteracji metody Newtona rozwiązywany był układ równań liniowych (także z założoną niską dokładnością) za pomocą metody GMRES z poprawą uwarunkowania macierzy układu. Rozważane były dwa przybliżenia macierzy jakobianowej układu. Jedno, oznaczane przez $\tilde{\mathbf{J}}$, zostało uzyskane przez prostą linearyzację sformułowania słabego MES, prowadzącą do wzoru (2.14). W procesie iteracyjnym rozwiązywany był układ równań:

$$\tilde{\mathbf{J}}(\mathbf{u}) \cdot \mathbf{u} = -\mathbf{F}(\mathbf{u}) \tag{2.95}$$

Drugie przybliżenie wynikało z zastosowania wzorów (2.12) w bezmacierzowym rozwiązywaniu równań metodą GMRES. Układ odpowiadający temu przypadkowi oznaczany będzie jako:

$$\mathbf{A}_J(\mathbf{u}) \cdot \mathbf{u} = -\mathbf{F}(\mathbf{u}) \tag{2.96}$$

gdzie: $\mathbf{A}_J(\mathbf{u}) \cdot \mathbf{u} \approx (\partial \mathbf{F}/\partial \mathbf{u}) \cdot \mathbf{u}$, a macierz $\mathbf{A}_J$ nie jest jawnie tworzona.

Poniżej przedstawione są cztery techniki realizacji podwójnych iloczynów macierzowych $\hat{\mathbf{r}}_i := \mathbf{M}^{-1}\mathbf{A}\bar{\mathbf{r}}_{i-1}$, występujących w metodzie GMRES dla powyższych symulacji. We wszystkich stosuje się łączne obliczenie podwójnego iloczynu za pomocą pojedynczej iteracji metod Schwarza (rozwiązania ciągu problemów 2.56), po której następuje odpowiednie odejmowanie wektorów. Dwie pierwsze realizacje są standardowymi technikami, opisanymi w p. 2.7.1, zastosowanymi do rozwiązania układu (2.95). Dwie następne odpowiadają układowi (2.96) i technice bezmacierzowej, w której wykorzystuje się wzory (2.12) i używa macierzy $\tilde{\mathbf{J}}(\mathbf{u})$ do poprawy uwarunkowania macierzy $\mathbf{A}_J$. Ze względu na zastosowanie pojedynczej iteracji metod Schwarza jej indeks jest opuszczony przy oznaczaniu wektorów $\hat{\mathbf{r}}_i$ i $\bar{\mathbf{r}}_{i-1}$.

Algorytmy odpowiadające poszczególnym przypadkom są oznaczane i wyglądają następująco:

- NLE–BJ – addytywna metoda Schwarza dla standardowego algorytmu GMRES

  Algorytm 2.6 (`NLE-BJ`)

  $(\hat{\mathbf{r}}_i)^0 := \bar{\mathbf{r}}_{i-1}$
  `dla` $l = 1, 2, ..., N_{\mathrm{bl}}$
  $\quad (\hat{\mathbf{r}}_i)_l := \tilde{\mathbf{J}}_{ll}^{-1} \cdot \left(- \sum_{m \neq l} \tilde{\mathbf{J}}_{lm} \cdot (\bar{\mathbf{r}}_{i-1})_m \right)$
  $\hat{\mathbf{r}}_i := -\hat{\mathbf{r}}_i + \bar{\mathbf{r}}_{i-1}$

- NLE–GS – multiplikatywna metoda Schwarza dla standardowego algorytmu GMRES

  Algorytm 2.7 (`NLE-GS`)

  $(\hat{\mathbf{r}}_i)^0 := \bar{\mathbf{r}}_{i-1}$
  `dla` $l = 1, 2, ..., N_{\mathrm{bl}}$
  $\quad (\hat{\mathbf{r}}_i)_l^l := \tilde{\mathbf{J}}_{ll}^{-1} \cdot \left(- \sum_{m \neq l} \tilde{\mathbf{J}}_{lm} \cdot (\hat{\mathbf{r}}_i)_m^{l-1} \right)$
  $\hat{\mathbf{r}}_i := -\hat{\mathbf{r}}_i + \bar{\mathbf{r}}_{i-1}$

- NKS–BJ – addytywna metoda Schwarza dla bezmacierzowego algorytmu GMRES

Algorytm 2.8 (NKS-BJ)

$\tilde{\mathbf{r}}_i := \frac{1}{\epsilon}\left(\mathbf{F}(\mathbf{u} + \epsilon\bar{\mathbf{r}}_{i-1}) - \mathbf{F}(\mathbf{u})\right)$
dla $l = 1, 2, ..., N_{\text{bl}}$
  $(\hat{\mathbf{r}}_i)_l := \tilde{\mathbf{J}}_{ll}^{-1} \cdot (\tilde{\mathbf{r}}_i)_l$

- NKS–GS – multiplikatywna metoda Schwarza dla bezmacierzowego algorytmu GMRES

Algorytm 2.9 (NKS-GS)

$(\hat{\mathbf{r}}_i)^0 := \bar{\mathbf{r}}_{i-1}$
dla $l = 1, 2, ..., N_{\text{bl}}$
  $(\hat{\mathbf{r}}_i)_l^l := (\hat{\mathbf{r}}_i)_l^{l-1} - \tilde{\mathbf{J}}_{ll}^{-1} \cdot \frac{1}{\epsilon}\left([\mathbf{F}(\mathbf{u} + \epsilon(\hat{\mathbf{r}}_i)^{l-1})]_l - [\mathbf{F}(\mathbf{u})]_l\right)$
$\hat{\mathbf{r}}_i := -\hat{\mathbf{r}}_i + \bar{\mathbf{r}}_{i-1}$

W dwóch pierwszych algorytmach zastosowane zostały wzory (2.56), jako wydajniejsze w implementacji od wzorów (2.63), wprowadzonych w celu analizy metod. We wzorach tych nie występuje iloczyn macierzy z wektorem, a tylko sekwencja odpowiednich problemów lokalnych. Podejście takie nie jest możliwe w przypadku metod bezmacierzowych. Wzór (2.12) przybliża iloczyn macierzy z dowolnym *globalnym* wektorem, a zatem konieczne staje się wykorzystanie wzorów (2.63).

Realizacja algorytmów bezmacierzowych dla techniki addytywnej jest prosta: oblicza się najpierw globalny iloczyn macierzy z wektorem, a następnie rozwiązuje serię problemów lokalnych. W metodzie multiplikatywnej wykonanie *jednego* globalnego iloczynu nie jest możliwe, gdyż wektor będący argumentem tego iloczynu zmienia się po każdym problemie lokalnym. Stąd realizacja poprawy uwarunkowania odbywa się dokładnie według wzorów (2.63).

Ma to istotne konsekwencje dla efektywności wykonania. Wyrażenie (2.12) musi być obliczane przed każdym problemem lokalnym (jednak tylko dla stopni swobody występujących w tym problemie, tzn. stopni swobody związanych z niezerowymi blokami w danym poziomym paśmie macierzy $\mathbf{A}_J$). Dlatego zwiększanie rozmiaru bloków podwójnie przyspiesza osiągnięcie zbieżności. Po pierwsze, metoda Schwarza dla większych bloków (podobszarów) jest szybciej

zbieżna, po drugie, kosztowne obliczanie wyrażeń (2.12) wykonywane jest rzadziej, gdyż zmniejsza się liczba lokalnych problemów. Ceną, jaką płaci się za to, jest większy koszt związany z odwracaniem większych bloków diagonalnych $\tilde{\mathbf{J}}_{ll}$ macierzy $\tilde{\mathbf{J}}$ i zwiększone zapotrzebowanie na pamięć operacyjną, związane z przechowywaniem bloków $\tilde{\mathbf{J}}_{ll}^{-1}$.

Przykład numeryczny – przepływ okołodźwiękowy wokół profilu lotniczego

Ilustracją omawianych zagadnień jest porównanie czterech przedstawionych powyżej technik dla przypadku rozwiązania pojedynczego układu równań liniowych w trakcie symulacji przepływu okołodźwiękowego wokół profilu lotniczego NACA0012. Jest to kontynuacja przykładu z p. 2.3.4 dotyczącego symulacji przepływu o liczbie Macha 0.85 i kącie natarcia 1^o. Rozważany układ równań liniowych powstał tam w efekcie zastosowania opisywanej metodologii całkowania pseudoczasowego i rozwiązywania rownań nieliniowych oraz aproksymacji MES na siatce adaptacyjnej zawierającej 5143 węzły (ilustracje użytej siatki i otrzymanego rozwiązania przedstawione są na rys. 2.4 i 2.5).

Tablica 2.6 przedstawia rezultaty osiągnięte dla czterech omawianych kombinacji sposobów przybliżenia macierzy jakobianowej w metodzie Newtona i poprawy uwarunkowania macierzy układu w metodzie GMRES oraz dla różnych rozmiarów bloków macierzy wykorzystanych przy poprawie uwarunkowania. Porównane są wyniki dla różnych wartości parametru CFL (2.7), decydującego o stopniu nieliniowości zagadnienia rozwiązywanego metodą Newtona. W zadaniu stosowana jest aproksymacja liniowa dla każdej z czterech składowych niewiadomej funkcji wektorowej. Bloki elementarne macierzy układu związane z pojedynczymi węzłami siatki mają więc cztery stopnie swobody. W algorytmach poprawy uwarunkowania stosuje się podwektory stopni swobody związane z grupami kilku węzłów. Odpowiada to dekompozycji obszaru obliczeniowego na podobszary, zawierające odpowiednie węzły jako węzły wewnętrzne, ze strefą nakładania się szerokości jednego elementu (mimo iż podwektory stopni swobody nie nakładają się na siebie). Liczba węzłow związanych z pojedynczym podwektorem (pojedynczym podobszarem) jest w tabl. 2.6 miarą wielkości bloków. Czasy obliczeń uzyskano na procesorze MIPS R10000 komputera Origin200 (patrz s. 20).

Z danych w tablicy widać, że dla wartości parametru CFL = 4 standardowe techniki dają nieco szybszą zbieżność GMRES (wyrażaną miarą (2.93)), przy nieco większym zapotrzebowaniu na pamięć i znacznie krótszym czasie działania. Wyższość techniki bezmacierzowej ukazuje się, gdy wartość parametru

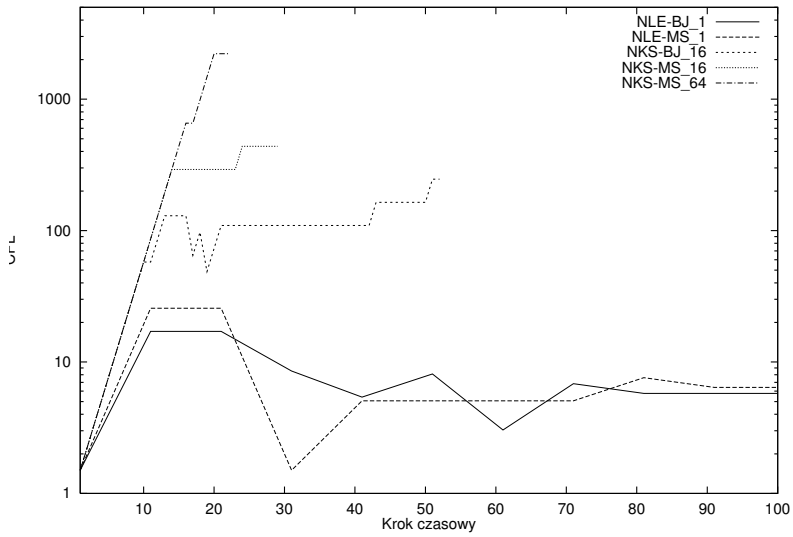
Tablica 2.6. Porównanie czasu działania, szybkości zbieżności (wyrażanej miarą (2.93)) i wymaganej pamięci dla metody GMRES wykorzystującej dwa przybliżenia macierzy jakobianowej – standardowe (NLE) i bezmacierzowe (NKS) – oraz dwa warianty metody Schwarza poprawy uwarunkowania macierzy układu równań liniowych – addytywny (BJ) i multiplikatywny (MS)

Algorytm	Rozmiar bloków	Pamięć [MBajt]	Zbieżność GMRES		Czas CPU [s]	
			CFL=4	CFL=64	CFL=4	CFL=64
NLE–BJ	4	12.95	0.602	—	7.51	—
	16	20.66	0.416	—	8.24	—
NLE–MS	4	12.95	0.365	—	6.29	—
	16	20.66	0.207	—	7.40	—
NKS–BJ	4	9.25	0.629	0.914	17.54	85.37
	16	16.95	0.478	0.843	13.87	49.96
	64	47.74	0.412	0.790	24.94	44.32
NKS–MS	4	9.25	0.393	0.842	19.02	93.86
	16	16.95	0.304	0.787	13.36	45.33
	64	47.74	0.245	0.713	17.22	38.01

Dane dla pojedynczego równania o 20 572 niewiadomych w wybranym kroku czasowym symulacji przepływu okołodźwiękowego wokół profilu NACA0012. Rozmiar bloków odpowiada przeciętnej liczbie węzłów siatki w pojedynczym podobszarze użytym do poprawy uwarunkowania metodą Schwarza. Czas CPU jest czasem rozwiązania pojedynczego układu równań.

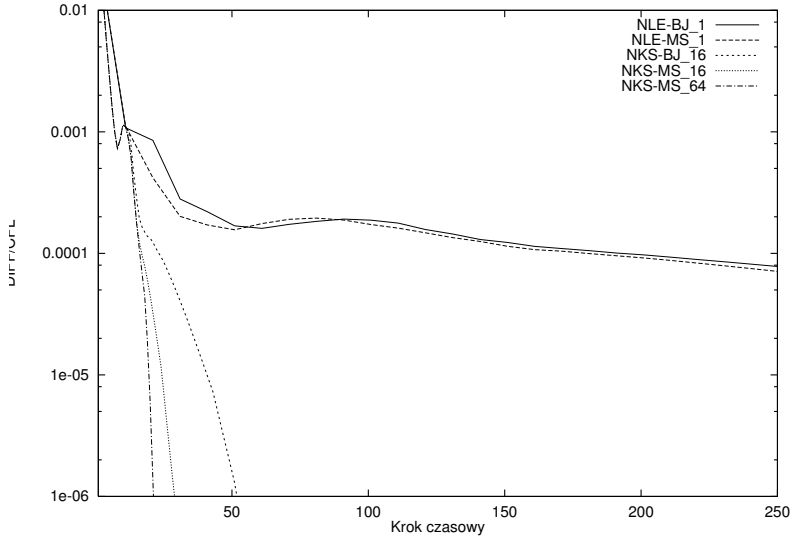
CFL wzrasta, a co za tym idzie wzrasta stopień nieliniowości zadania i pogarsza się uwarunkowanie macierzy układu liniowego. Dla technik standardowych ograniczenie wartości CFL w symulacjach wynika z konieczności zapewnienia zbieżności metody Newtona. Dla technik bezmacierzowych przybliżenie macierzy jakobianowej jest tak dobre, że zbieżność metody Newtona nie stanowi problemu. Ograniczeniem staje się zapewnienie zbieżności metody GMRES. Przyczyna tego tkwi w fakcie, że do poprawy uwarunkowania układu stosuje się macierz $\tilde{\mathbf{J}}$ źle przybliżającą macierz jakobianową, a więc dla dużych wartości CFL istotnie różną od macierzy $\mathbf{A}_J$ wynikającej z techniki bezmacierzowej.

Ograniczenie wartości CFL dla technik bezmacierzowych jest znacznie mniej restrykcyjne niż dla technik standardowych. Rysunek 2.13 przedstawia długość kroku czasowego, dla różnych technik i metod poprawy uwarunkowania, w trak-



Rys. 2.13. Symulacja okołodźwiękowego przepływu wokół profilu NACA0012: zmienność wartości parametru CFL w trakcie symulacji dla różnych wersji metody GMRES rozwiązywania układów równań liniowych; oznaczenia sposobu przybliżenia macierzy jakobianowej: NLE – za pomocą linearyzacji, NKS – techniką bezmacierzową; oznaczenia metod poprawy uwarunkowania macierzy układu: BJ – addytywna metoda Schwarza, GS - multiplikatywna metoda Schwarza; liczba w symbolu metody oznacza rozmiar podobszarów użytych w algorytmach metod Schwarza, mierzony liczbą węzłów wewnętrznych (szczegółowy opis metod w tekście)

cie całej symulacji przepływu, w przypadku zastosowania strategii maksymalizacji wartości parametru CFL na każdym kroku czasowym [19]. Oznaczenie każdej z metod wzbogacono o przeciętną liczbę węzłów w blokach macierzy $\tilde{\mathbf{J}}_{ll}^{-1}$, użytych do poprawy uwarunkowania. Widać, że dla technik bezmacierzowych wartość parametru CFL może być o kilka rzędów wielkości większa niż dla technik standardowych. Pozwala to uzyskać zdecydowane przyspieszenie zbieżności całej symulacji, zilustrowane na rys. 2.14 i 2.15 (miarą zbieżności jest ponownie iloraz DIFF/CFL (2.8)). Liczba kroków czasowych w całej symulacji jest dla technik bezmacierzowych kilka rzędów wielkości mniejsza niż dla technik standardowych. Biorąc pod uwagę większy koszt obliczeniowy technik bezmacierzowych, ich przewaga w całkowitym czasie symulacji maleje, wciąż

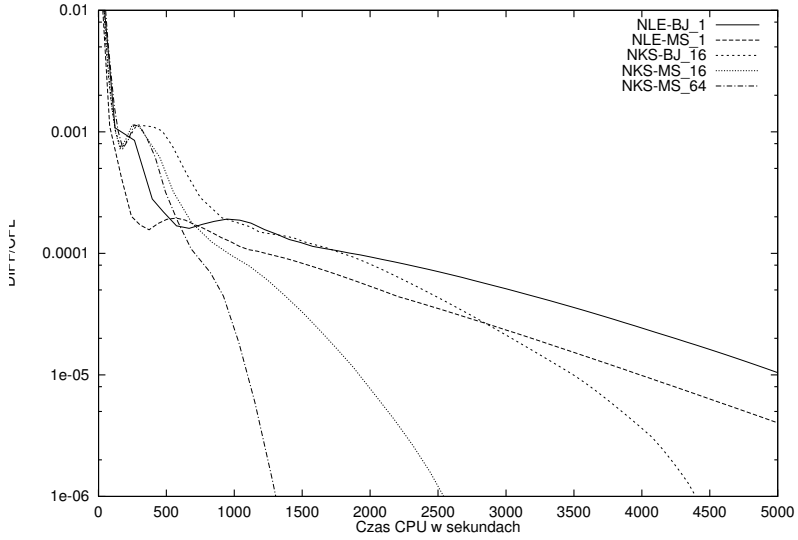


Rys. 2.14. Symulacja okołodźwiękowego przepływu wokół profilu NACA0012: globalna zbieżność (wyrażana miarą (2.8)) jako funkcja liczby kroków czasowych dla poszczególnych metod rozwiązywania układów równań liniowych (opis metod jak dla rys. 2.13)

jednak pozostaje znacząca.

2.7.7 Złożoność obliczeniowa rozwiązywania układów równań liniowych

Macierze układów równań liniowych powstające w wyniku dyskretyzacji MES są nie tylko macierzami rzadkimi, ale najczęściej (przy zastosowaniu odpowiedniej numeracji stopni swobody) także macierzami pasmowymi (wstęgowymi) [184, 91, 42]. Szerokość pasma macierzy $\mathbf{A}$, w którym mogą występować wyrazy niezerowe, oznaczana przez N_{d}, jest parametrem decydującym o złożoności obliczeniowej metod bezpośredniego rozwiązywania układów równań liniowych MES. W trakcie obliczeń za pomocą rozmaitych wariantów eliminacji Gaussa, pasmo, początkowo zawierające przewagę wyrazów zerowych, jest wypełniane liczbami różnymi od zera, dla których wykonuje się operacje algorytmów. N_{d} jest najczęściej wielkością rzędu $N^{2/3}$ w przypadku symulacji trójwymiarowych (3W) i rzędu $N^{1/2}$ dla problemów dwuwymiarowych (2W) (dla szczególnie korzystnych geometrii obszaru obliczeniowego lub technik apro-



Rys. 2.15. Symulacja okołodźwiękowego przepływu wokół profilu NACA0012: globalna zbieżność (wyrażana miarą (2.8)) jako funkcja całkowitego czasu symulacji dla poszczególnych metod rozwiązywania układów równań liniowych (opis metod jak dla rys. 2.13)

ksymacji wartości te mogą być niższe – przypadki te nie są uwzględniane w dalszych analizach).

Jeśli za złożoność pamięciową przyjmie się wielkość potrzebną do przechowania całego pasma, to będzie ona rzędu $N^{5/3}$ (3W) lub $N^{3/2}$ (2W). Dla liczby stopni swobody w problemie ponad milion daje to rozmiary wymaganej pamięci rzędu kilkudziesięciu i kilkuset Gigabajtów. Dlatego w realizacji algorytmów bezpośredniego rozwiązywania układów równań liniowych popularne są techniki tzw. *out-of-core*, korzystające w trakcie wykonywania obliczeń z pamięci dyskowej, co jednak znacznie spowalnia działanie w porównaniu z algorytmami korzystającymi wyłącznie z pamięci operacyjnej.

Metody iteracyjne, szczegółowo analizowane w niniejszej pracy, wymagają w przypadku rzadkich macierzy układu znacznie mniej pamięci niż metody bezpośrednie. Potrzebny rozmiar, równy złożoności jednej iteracji metody, jest ograniczony do niewielkiej wielokrotności rozmiaru zajmowanego przez niezerowe wyrazy macierzy, który to rozmiar jest (patrz p. 2.5.5) tego samego rzędu, co liczba niewiadomych (jako że liczba niezerowych wyrazów w pojedynczym wierszu macierzy układu jest ograniczona). W efekcie metody iteracyjne mają

liniowe złożoności pamięciowe, co umożliwia, w zdecydowanej większości przypadków, realizację algorytmów *in-core* – wyłącznie w pamięci operacyjnej.

Rząd złożoności czasowej wariantów eliminacji Gaussa wynosi NN_{d}^2, co dla przypadków różnych wymiarów problemów daje $N^{7/3}$ (3W) lub N^2 (2W). Dla miliona stopni swobody i procesora o wydajności 1 Gflop/s czas rozwiązania układu wynosi kilka godzin (dla metod iteracyjnych w przypadku odpowiedniej szybkości zbieżności – kilkanaście do kilkudziesięciu iteracji metody dla uzyskania założonej tolerancji – czas ten może wynosić kilka minut, patrz p. 2.7.11).

Dodatkową wadą eliminacji Gaussa jest brak efektywnych realizacji równoległych [78]. Przy liczbie stopni swobody w zakresie dziesiątek i setek milionów, nawet najbardziej wyrafinowane równoległe warianty eliminacji Gaussa stają się praktycznie nierealizowalne[23].

Inaczej prezentują się charakterystyki solwerów iteracyjnych. Złożoność czasowa metod iteracyjnych zależy w głównej mierze od zbieżności procesu iteracyjnego, jej istnienia i szybkości. W ogólnym przypadku zbieżność zależy i od cech rozwiązywanego problemu, i od metod aproksymacji. Mimo istnienia wielu typów zagadnień i metod dyskretyzacji, dla których nie opracowano jeszcze efektywnych algorytmów iteracyjnych, istnieje grupa metod iteracyjnych o szerokim spektrum zastosowań i wydajnej realizacji, tak sekwencyjnej, jak i równoległej. Do grupy tej należą, omawiane w niniejszej pracy, metody wielosiatkowe (patrz p. 2.7.3 oraz algorytm `MG`) i metody podprzestrzeni Kryłowa (patrz p. 2.7.5 oraz algorytmy `CG` i `GMRES`) z odpowiednio dobraną poprawą uwarunkowania macierzy układu. Zbieżność metod Kryłowa zależy w decydujący sposób od uwarunkowania macierzy układu. Najefektywniejszą metodą poprawy uwarunkowania jest metoda wielosiatkowa. W dalszej części niniejszego punktu oraz dwóch punktach następnych analizowana będzie złożoność takiego właśnie, należącego do najefektywniejszych obecnie, solwera, powstałego z połączenia metody Kryłowa i wielosiatkowej poprawy uwarunkowania macierzy.

Dla metody wielosiatkowej, stosowanej jako niezależny solwer układów równań liniowych, istnieją dowody optymalnej liniowej złożoności obliczeniowej [84, 176, 45, 118]. W większości przypadków dowody te uzyskiwane są dla dyskretyzacji liniowych zagadnień eliptycznych, w prostych obszarach oblicze-

[23] Na liście Top500 najpotężniejszych systemów komputerowych na świecie z czerwca 2004 r. pierwsze miejsce zajmował komputer Earth Simulator, będący wieloprocesorową (5120 procesorów) maszyną złożoną z procesorów wektorowych, wyprodukowaną przez firmę NEC. Potrafi on rozwiązać za pomocą eliminacji Gaussa układ równań o ok. 10^6 stopni swobody w ok. 6 godzin. Algorytm rozwiązywania nie wykorzystuje ewentualnej pasmowości lub rzadkości macierzy.

niowych i dla prostych aproksymacji (najczęściej dokonywanych metodą różnic skończonych). Dla bardziej złożonych problemów (np. nieliniowych) i aproksymacji (np. wyższego rzędu, anizotropowych) trudniej o teoretyczne dowody zbieżności, a praktyczne eksperymenty pokazują, że w wielu przypadkach zbieżność pogarsza się. Nadal jednak może pozostać zbliżona do optymalnej, jeśli liczba cykli V koniecznych do uzyskania zbieżności solwera nieznacznie tylko zmienia się wraz ze zwiększaniem się liczby stopni swobody (w przypadku obliczeń wielkiej skali adaptacyjną MES najczęściej związanym ze zmniejszaniem się rozmiaru elementów). Dotyczy to jednakowo cykli V metody wielosiatkowej, jako niezależnego solwera, jak i metod podprzestrzeni Kryłowa z pojedynczym cyklem V, jako techniką poprawy uwarunkowania macierzy układu. Eksperymentalne badanie zbieżności tego ostatniego solwera w przypadku nieciągłej aproksymacji MES zawiera p. 2.7.11.

2.7.8 Analiza złożoności obliczeniowej rozwiązywania układów równań liniowych – przypadek ogólny

Rozważona zostanie złożoność obliczeniowa metody GMRES z wielosiatkową poprawą uwarunkowania macierzy. W stosunku do metody sprzężonych gradientów dodatkowym, istotnym z punktu widzenia złożoności, elementem algorytmu jest jawne obliczanie bazy przestrzeni Kryłowa. Poza tym złożoność obu metod (w przypadku stosowania tych samych metod poprawy uwarunkowania) jest zbliżona.

Porównane będą dwa warianty algorytmu. W pierwszym, oznaczanym dalej symbolem GMRES–MG–MS, zastosowane będzie wygładzanie błędu multiplikatywną metodą Schwarza, w drugim, oznaczanym jako GMRES–MG–ILU, do tego celu użyta zostanie niekompletna eliminacja Gaussa w postaci techniki rozkładu ILU(0).

Złożoność obliczeniowa algorytmu `GMRES`

Pojedynczy restart algorytmu `GMRES` składa się z jednego obliczenia residuum układu, kilku mniej istotnych obliczeniowo operacji (w tym obliczenia ostatecznej postaci rozwiązania) i pętli po wektorach Kryłowa (wektorach bazowych podprzestrzeni Kryłowa). W pojedynczej pętli liczba iteracji, nazywanych dalej iteracjami GMRES, jest ograniczona przez maksymalną liczbę wektorów Kryłowa w restarcie – N_{k}.

Pojedyncza iteracja algorytmu GMRES składa się z następujących, istotnych z punktu widzenia kosztu obliczeniowego, operacji:

- obliczenie iloczynu $\mathbf{M}^{-1}\mathbf{A}\bar{\mathbf{r}}_{i-1}$,
- odpowiedni fragment zmodyfikowanej procedury Grama-Schmidta.

Rozwiązywany w każdej iteracji problem minimalizacyjny metody GMRES (z prostokątną macierzą układu równań liniowych, stąd do jego rozwiązania stosuje się specjalne techniki, jak np. rozkład QR) ma rozmiar równy liczbie wektorów Kryłowa w aktualnym momencie. Jako że liczba ta jest niewielka, koszt rozwiązania problemu minimalizacyjnego jest pomijalny.

Zmodyfikowana ortonormalizacja Grama-Schmidta

Fragment procedury ortonormalizacji Grama-Schmidta związany z pojedynczym wektorem bazowym polega na obliczeniu iloczynu skalarnego tego wektora ze wszystkimi uprzednio obliczonymi wektorami bazy, a następnie odjęciu odpowiedniej kombinacji liniowej obliczonych wektorów od wektora rozważanego. Jej wariant, zwany procesem Arnoldiego, użyty w algorytmie `GMRES` na s. 80, polega na jednoczesnym obliczaniu iloczynów skalarnych i odejmowaniu od rozważanego wektora. Celem tej modyfikacji jest zwiększenie odporności na błędy zaokrągleń, które nieodłącznie pojawiają się w trakcie procedury, zmniejszając dokładność ortonormalizacji. W przypadku restartowanej metody GMRES i wymiarów podprzestrzeni Kryłowa rzędu kilkudziesięciu, błędy te nie wpływają znacząco na zbieżność algorytmu GMRES.

Złożoność pamięciowa procedury ortonormalizacji Grama-Schmidta jest determinowana przez konieczność przechowywania wszystkich wektorów Kryłowa utworzonych w danym restarcie. Koszt pamięciowy wynosi więc NN_{k}, gdzie N jak zwykle oznacza całkowitą liczbę stopni swobody w problemie, będącą jednocześnie długością pojedynczego wektora Kryłowa. Koszt ten jest porównywalny z kosztem przechowania oryginalnej macierzy układu $\mathbf{A}$ i bloków macierzy związanych z poprawą uwarunkowania $\mathbf{A}$. Dla wielu problemów (m.in. opisywanych w niniejszej pracy problemów dyskretyzacji czasowej i pseudoczasowej dla równań Eulera [18, 19]) zbieżność metody GMRES jest na tyle dobra, że wykorzystuje się tylko kilka wektorów w pojedynczym restarcie, a restartowanie nie powoduje istotnego wydłużenia czasu działania. Nierzadko jednak wprowadzenie restartu zaburza zbieżność algorytmu GMRES, co wymusza użycie podprzestrzeni Kryłowa o wymiarach rzędu kilkudziesięciu (jest tak np. w przypadku zastosowania nieciągłej aproksymacji Galerkina do problemów z nieciągłymi współczynnikami o dużych skokach [29]).

Złożoność czasowa w przypadku ortonormalizacji pojedynczego wektora $\hat{\mathbf{r}}_i$

jest równa:

$$\sum_{j=0}^{i-1} 4N \tag{2.97}$$

Koszt ortonormalizacji dla pełnego restartu wynosi więc:

$$4N \sum_{i=1}^{N_\mathrm{k}} i = 2NN_\mathrm{k}(N_\mathrm{k}+1) \tag{2.98}$$

Kwadratowa złożoność czasowa procedury Grama-Schmidta (jako funkcji wymiaru przestrzeni Kryłowa) może powodować, że koszt ortonormalizacji wektorów Kryłowa przewyższa koszt poprawy uwarunkowania macierzy (zwłaszcza jeśli ten skaluje się liniowo wraz ze wzrostem liczby stopni swobody N).

Powyższe fakty pokazują, że w przypadku słabej zbieżności metody GMRES, koszt obliczeniowy związany z jawnym tworzeniem bazy podprzestrzeni Kryłowa może dominować w całkowitej złożoności algorytmu GMRES. Dlatego zawsze stara się dobrać podprzestrzeń o minimalnym wymiarze (często metodą prób i błędów) i stosować efektywną poprawę uwarunkowania macierzy układu, poprawiającą zbieżność metody GMRES.

Poprawa uwarunkowania macierzy układu

Obliczenie iloczynu $\mathbf{M}^{-1}\mathbf{A}\bar{\mathbf{r}}_{i-1}$, realizującego poprawę uwarunkowania macierzy $\mathbf{A}$, jest dokonywane w analizowanej metodzie algorytmem wielosiatkowym `MG` (s. 74). W obu rozpatrywanych wariantach (GMRES–MG–MS i GMRES––MG–ILU) nie tworzy się macierzy poprawy uwarunkowania $\mathbf{M}^{-1}$, lecz wykorzystuje przetworzone w odpowiedni sposób bloki macierzy $\mathbf{A}$.

Złożoność obliczeniowa pojedynczego cyklu V metody wielosiatkowej jest sumą złożoności dla kolejnych poziomów siatki. Obliczenia na poziomach wyższych składają się z następujących etapów:

- wygładzanie błędu na siatce gęstej,
- obliczenie residuum,
- rzutowanie residuum na siatkę rzadką,
- rozszerzenie rozwiązania problemu korekty na siatce rzadkiej do przestrzeni związanej z siatką gęstą,
- ponowne wygładzanie błędu na siatce gęstej.

Poza tym występują standardowe operacje wektorowe (*axpy*) o relatywnie niskiej złożoności obliczeniowej.

W przypadku siatki najrzadszej układ równań musi zostać rozwiązany dokładnie. Koszt rozwiązania zależy od właściwości problemu i rodzaju aproksymacji. Istnieją zadania, które wymagają zastosowania metod bezpośrednich (np. dla problemów nieokreślonych dodatnio). Dla innych, wybór solwera bezpośredniego lub iteracyjnego będzie zależał od efektywności obliczeniowej. Jeżeli liczba stopni swobody dla siatki najrzadszej jest niewielka (rzędu tysięcy), zastosowanie metod bezpośrednich staje się opłacalne. W rozważaniach niniejszej pracy zakłada się, że układ odpowiadający siatce najrzadszej jest na tyle mały, że koszt obliczeniowy związany z jego rozwiązaniem jest niższego rzędu niż koszt wygładzania błędu na siatce najgęstszej (ostatecznej).

Jeżeli złożoność obliczeń na siatce ostatecznej jest liniowa, to analiza złożoności obliczeniowej cyklu V prowadzi do rekurencji, której rozwiązaniem jest także funkcja liniowa względem liczby niewiadomych w układzie. Z tego powodu analiza złożoności obliczeniowej w dalszych punktach przeprowadzona zostanie dla operacji związanych z siatką ostateczną.

Złożoność obliczeniowa uzyskiwania residuum układu

Złożoność obliczeniowa uzyskiwania residuum układu zależy od niezerowej struktury i sposobu przechowywania macierzy $\mathbf{A}$. W niniejszej analizie zakłada się, że macierz $\mathbf{A}$ przechowywana jest z wykorzystaniem bloków elementarnych $\mathbf{A}^{\mathrm{E}}_{L,M}$. Daje to optymalną złożoność pamięciową (przechowywane są wyłącznie elementy niezerowe macierzy), przy jednoczesnym grupowaniu elementów, co usprawnia zarządzanie pamięcią w trakcie operacji macierzowych (mniej adresowania pośredniego) i przyczynia się do lepszego wykorzystania pamięci podręcznej komputera.

Liczba operacji potrzebnych do obliczenia residuum układu równań jest zdominowana przez liczbę operacji przy realizacji iloczynu z macierzą układu. Obliczenie iloczynu odbywa się w pętli po podwektorach elementarnych i związanych z nimi poziomych pasmach macierzy $\mathbf{A}$. Dla każdego podwektora elementarnego wykonuje się pętlę po wszystkich podwektorach sąsiadujących, a więc takich, dla których w danym paśmie macierzy $\mathbf{A}$ istnieją niezerowe wyrazy (bloki elementarne pozadiagonalne). Oznaczając liczbę sąsiadów i-tego podwektora przez N^i_{sbl} oraz liczbę stopni swobody w podwektorze i przez N^i_{ssb},

uzyskuje się złożoność czasową dla pojedynczego podwektora elementarnego:

$$2(N^i_{\mathrm{ssb}})^2 + \sum_{j=1}^{N^i_{\mathrm{sbl}}} 2N^i_{\mathrm{ssb}} N^{\mathrm{gls}(i,j)}_{\mathrm{ssb}} \tag{2.99}$$

i w efekcie dla mnożenia $\mathbf{A}$ z dowolnym wektorem globalnym:

$$\sum_{i=1}^{N^{\mathrm{E}}_{\mathrm{bl}}} 2N^i_{\mathrm{ssb}} \left(N^i_{\mathrm{ssb}} + \sum_{j=1}^{N^i_{\mathrm{sbl}}} N^{\mathrm{gls}(i,j)}_{\mathrm{ssb}} \right) \tag{2.100}$$

gdzie $N^{\mathrm{E}}_{\mathrm{bl}}$ oznacza całkowitą liczbę bloków elementarnych, a $\mathrm{gls}(i,j)$ globalny numer (identyfikator) j-tego sąsiada i-tego podwektora elementarnego.

Ze względu na nienakładanie się bloków elementarnych (patrz p. 2.5.5) wzór (2.100) określa (podwojoną) liczbę niezerowych wyrazów macierzy $\mathbf{A}$. Określa tym samym złożoność pamięciową obliczania residuum układu równań w metodzie GMRES. Jak widać, złożoność ta zależy od struktury siatki i typu aproksymacji MES, które determinują liczbę stopni swobody w pojedynczym podwektorze elementarnym i liczbę sąsiadów pojedynczego podwektora.

Złożoność obliczeniowa prolongacji i restrykcji w metodzie wielosiatkowej

Obie operacje związane z przenoszeniem rozwiązania pomiędzy siatkami gęstą i rzadką realizowane są z tą samą złożonością obliczeniową. Niezależnie czy formuje się jawnie macierz $\mathbf{R}_{C_i}$, czy dokonuje operacji rzutowania, algorytmy rozszerzenia i zawężenia są podobne, zmienia się natomiast układ działań. Złożoność pozostaje złożonością mnożenia wektora stopni swobody na odpowiednim poziomie przez macierz $\mathbf{R}_{C_i}$ lub $\mathbf{R}^T_{C_i}$. Przy realizacji mnożenia wykonuje się operacje tylko na niezerowych elementach macierzy, a liczba tych ostatnich jest jedynym czynnikiem determinującym złożoność obliczeniową rozszerzenia i zawężenia (prolongacji i restrykcji). W efekcie złożoność ta (tak pamięciowa, jak i czasowa) jest zawsze liniową funkcją liczby stopni swobody w problemie.

W przypadku operatorów otrzymanych za pomocą wzorów (2.81) liczba niezerowych wyrazów w macierzy $\mathbf{R}_{C_i}$ jest zależna od proporcji i rozkładu stopni swobody na siatce gęstej i rzadkiej. Każdy i-ty wiersz macierzy $\mathbf{R}_{C_i}$ odpowiada pojedynczemu stopniowi swobody na siatce rzadkiej, a więc i pojedynczej funkcji bazowej $\phi^i_{C_{i-1}}$ na tej siatce. Liczba niezerowych wyrazów w pojedynczym wierszu zależy od liczby funkcji bazowych na siatce gęstej, które użyte są do interpolacji $\phi^i_{C_{i-1}}$. Funkcje te nie zerują się wspólnie z $\phi^i_{C_{i-1}}$ w co najmniej

jednym elemencie siatki gęstej i są równe zeru wszędzie tam, gdzie $\phi^i_{C_{i-1}}$ jest równa zeru.

W efekcie liczba niezerowych wyrazów w macierzy $\mathbf{R}_{C_i}$ zależy od typu i stopnia aproksymacji oraz od sposobu konstrukcji siatki rzadkiej. W przypadku opisywanej metody wielosiatkowej, gdzie kolejne siatki są związane z elementami pochodzącymi z podziałów elementów-rodziców, proporcja stopni swobody na siatce rzadkiej i gęstej zależy od liczby elementów-dzieci dla pojedynczego dzielonego elementu oraz od stopni aproksymacji we wszystkich uczestniczących w podziale elementach.

Zagadnieniem odrębnym od użycia macierzy $\mathbf{R}_{C_i}$ jest problem obliczenia jej współczynników. Obliczenia tego dokonuje się jednorazowo dla danej pary siatek (macierz $\mathbf{R}_{C_i}$ może być wykorzystywana wielokrotnie dla kolejnych wywołań solwera). Wyrazy macierzy $\mathbf{R}_{C_i}$ najczęściej odpowiadają współczynnikom interpolacji funkcji bazowych siatki rzadkiej na siatce gęstej. Jeśli taka prosta interpolacja nie jest możliwa, alternatywą jest rozwiązanie odpowiedniego zadania projekcji dla każdej funkcji bazowej. Dla aproksymacji wyższego stopnia użycie w takich przypadkach nieoptymalnych (nieortogonalnych) funkcji bazowych może spowodować znaczny wzrost złożoności obliczeniowej – rzędu p^9.

Złożoność obliczeniowa wygładzania błędu za pomocą pojedynczej iteracji multiplikatywnej metody Schwarza

Różnice między technikami poprawy GMRES–MG–MS i GMRES–MG–ILU polegają na różnym sposobie przeprowadzenia wygładzania błędu na siatce danego poziomu. Dla multiplikatywnej metody Schwarza rozważony zostanie algorytm odpowiadający blokowej metodzie Gaussa-Seidla (wzór (2.56) i algorytm NLE–GS z przykładu w p. 2.7.6). W celu uogólnienia rozważań założone zostanie, że bloki macierzy $\mathbf{A}$ wykorzystane do poprawy uwarunkowania odpowiadają grupom obiektów siatki, a więc składają się z wielu bloków elementarnych i mogą zachodzić na siebie. Bloki takie nazywane będą blokami poprawy uwarunkowania i oznaczane przez $\mathbf{A}^{\mathrm{P}}_{KM}$. Każdemu takiemu blokowi odpowiada para podwektorów stopni swobody, nazywanych dalej podwektorami poprawy uwarunkowania (blokom diagonalnym odpowiada pojedynczy podwektor). Pojedynczy K-ty podwektor poprawy uwarunkowania składa się z N^K_{bl} odpowiednich elementarnych podwektorów stopni swobody.

Ilustracją konstrukcji bloków $\mathbf{A}^{\mathrm{P}}_{KM}$ jest rys. 2.16. Przedstawia on fragment macierzy $\mathbf{A}$ związany z trzema podwektorami poprawy uwarunkowania. Każdemu podwektorowi K odpowiada poziome pasmo macierzy $\mathbf{A}$ o szerokości

(wysokości) równej liczbie stopni swobody w podwektorze:

$$N_{\mathrm{ssB}}^{K} = \sum_{i=1}^{N_{\mathrm{bl}}^{K}} N_{\mathrm{ssb}}^{\mathrm{GL}(i,K)} \tag{2.101}$$

gdzie $\mathrm{GL}(i,K)$ określa globalny numer i-tego kolejnego podwektora elementarnego składającego się na K-ty podwektor poprawy uwarunkowania. Rysunek 2.16 przedstawia fragment macierzy $\mathbf{A}$ wokół przekątnej głównej, w przypadku gdy podwektory poprawy uwarunkowania nie zachodzą na siebie. Bloki poprawy uwarunkowania na rys. 2.16 utworzone są z bloków elementarnych macierzy $\mathbf{A}$. Z każdym diagonalnym blokiem poprawy uwarunkowania związany jest podwektor poprawy uwarunkowania. Każdemu podwektorowi poprawy uwarunkowania odpowiada podobszar obszaru Ω, w którym funkcje bazowe związane ze stopniami swobody podwektora są niezerowe. Jeśli podobszary związane z dwoma podwektorami poprawy uwarunkowania częściowo nakładają się na siebie (sąsiadują z sobą), wtedy istnieją w macierzy $\mathbf{A}$ niezerowe wyrazy odpowiadające obu podwektorom. Wyrazy takie tworzą pozadiagonalne bloki poprawy uwarunkowania. Rysunek 2.16 pokazuje sytuację, gdy dla dwóch spośród trzech reprezentowanych podwektorów poprawy uwarunkowania, związane z nimi podobszary sąsiadują z sobą lub nakładają się. W efekcie powstają dwa pozadiagonalne bloki poprawy uwarunkowania, po jednym w odpowiednich poziomych pasmach macierzy $\mathbf{A}$.

Sytuacja komplikuje się, jeśli podobszary związane z poszczególnymi blokami poprawy uwarunkowania nakładają się na tyle, że związane z nimi podwektory poprawy uwarunkowania częściowo się pokrywają. Wtedy pasma macierzy $\mathbf{A}$ związane z podwektorami poprawy uwarunkowania także pokrywają się i niektóre z bloków elementarnych występują jednocześnie w różnych blokach poprawy uwarunkowania.

Algorytm wygładzania błędu jest realizowany w pętli po podwektorach poprawy uwarunkowania. Dla każdego podwektora dokonuje się mnożenia fragmentu odpowiadającego mu poziomego pasma macierzy układu z globalnym wektorem stopni swobody. W każdym pasmie uwzględnia się wszystkie niezerowe pozadiagonalne bloki elementarne (stosownie do wzoru (2.56)). Oznacza to pętlę po N_{bl}^{K} podwektorach elementarnych tworzących podwektor poprawy uwarunkowania i wykonanie mnożenia o złożoności czasowej danej wzorem:

$$\sum_{i=1}^{N_{\mathrm{bl}}^{K}} \sum_{j=1}^{N_{\mathrm{sbl,off}}^{\mathrm{GL}(i,K)}} 2 N_{\mathrm{ssb}}^{\mathrm{GL}(i,K)} N_{\mathrm{ssb}}^{\mathrm{gls}(\mathrm{GL}(i,K),j)} \tag{2.103}$$

$$\begin{pmatrix} \mathbf{A}^{\mathrm{E}}_{1,1} & \mathbf{A}^{\mathrm{E}}_{1,2} & \mathbf{A}^{\mathrm{E}}_{1,3} \\ \mathbf{A}^{\mathrm{E}}_{2,1} & \mathbf{A}^{\mathrm{E}}_{2,2} & \mathbf{A}^{\mathrm{E}}_{2,3} \\ \mathbf{A}^{\mathrm{E}}_{3,1} & \mathbf{A}^{\mathrm{E}}_{3,2} & \mathbf{A}^{\mathrm{E}}_{3,3} \end{pmatrix} \quad \begin{pmatrix} \mathbf{A}^{\mathrm{E}}_{1,6} & \mathbf{A}^{\mathrm{E}}_{1,7} \\ \mathbf{A}^{\mathrm{E}}_{2,6} & \mathbf{A}^{\mathrm{E}}_{2,7} \\ \mathbf{A}^{\mathrm{E}}_{3,6} & \mathbf{A}^{\mathrm{E}}_{3,7} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{A}^{\mathrm{E}}_{4,4} & \mathbf{A}^{\mathrm{E}}_{4,5} \\ \mathbf{A}^{\mathrm{E}}_{5,4} & \mathbf{A}^{\mathrm{E}}_{5,5} \end{pmatrix} \tag{2.102}$$

$$\begin{pmatrix} \mathbf{A}^{\mathrm{E}}_{6,1} & \mathbf{A}^{\mathrm{E}}_{6,2} & \mathbf{A}^{\mathrm{E}}_{6,3} \\ \mathbf{A}^{\mathrm{E}}_{7,1} & \mathbf{A}^{\mathrm{E}}_{7,2} & \mathbf{A}^{\mathrm{E}}_{7,3} \end{pmatrix} \quad \begin{pmatrix} \mathbf{A}^{\mathrm{E}}_{6,6} & \mathbf{A}^{\mathrm{E}}_{6,7} \\ \mathbf{A}^{\mathrm{E}}_{7,6} & \mathbf{A}^{\mathrm{E}}_{7,7} \end{pmatrix}$$

Rys. 2.16. Konstrukcja bloków poprawy uwarunkowania macierzy $\mathbf{A}$ układu równań liniowych MES (zaznaczonych nawiasami okrągłymi), na bazie bloków elementarnych $\mathbf{A}^{\mathrm{E}}_{L,M}$

gdzie $N^{\mathrm{GL}(i,K)}_{\mathrm{sbl,off}}$ oznacza liczbę sąsiadów podwektora elementarnego $\mathrm{GL}(i,K)$, nie należących wraz z podwektorem $\mathrm{GL}(i,K)$ do K-tego podwektora poprawy uwarunkowania.

Kolejnym etapem algorytmu blokowej metody Gaussa-Seidla jest rozwiązanie sekwencji problemów lokalnych (2.56). Rozwiązanie to dokonywane jest wielokrotnie w trakcie iteracji, za każdym razem z tymi samymi macierzami lokalnych układów równań, lecz różnymi wektorami prawych stron. Zakładając, że macierze układów (2.56) są gęste, można do rozwiązania wykorzystać eliminację Gaussa z klasycznym rozkładem LU. W takim wypadku dla pojedynczej macierzy lokalnej faza rozkładu ma złożoność czasową $2/3(N^K_{\mathrm{ssB}})^3$, natomiast rozwiązanie można uzyskać wykonując $4(N^K_{\mathrm{ssB}})^2$ operacji (jeśli rozwiązanie otrzymywane jest w wyniku mnożenia przez odwrotność macierzy układu, złożoność zmniejsza się do $2(N^K_{\mathrm{ssB}})^2$). Fazę rozkładu LU bloków poprawy uwarunkowania można przeprowadzić odrębnie, przed rozpoczęciem iteracji, a jej złożoność dla wszystkich $N^{\mathrm{P}}_{\mathrm{bl}}$ bloków poprawy uwarunkowania wynosi:

$$\sum_{K=1}^{N^{\mathrm{P}}_{\mathrm{bl}}} 2/3(N^K_{\mathrm{ssB}})^3 \tag{2.104}$$

Ostatecznie pojedyncza iteracja uogólnionej blokowej metody Gaussa-Seidla (metody multiplikatywnej Schwarza), będąca sekwencją rozwiązań lokalnych układów równań (2.56), gdzie najpierw dokonuje się mnożenia (2.103), następnie dodaje odpowiedni fragment wektora prawej strony i w końcu przeprowadza fazę rozwiązania układu z uprzednio odwróconą macierzą diagonalną $\mathbf{A}^{\mathrm{P}}_{KK}$, ma

złożoność czasową:

$$\sum_{K=1}^{N_{\mathrm{bl}}^{\mathrm{P}}} \left(2(N_{\mathrm{ssB}}^{K})^2 + N_{\mathrm{ssB}}^{K} + \sum_{i=1}^{N_{\mathrm{bl}}^{K}} \sum_{j=1}^{N_{\mathrm{sbl,off}}^{\mathrm{GL}(i,K)}} 2N_{\mathrm{ssb}}^{\mathrm{GL}(i,K)} N_{\mathrm{ssb}}^{\mathrm{gls}(\mathrm{GL}(i,K),j)} \right) \tag{2.105}$$

W przypadku gdy bloki $\mathbf{A}_{KK}^{\mathrm{P}}$ nie nakładają się na siebie, powyższa złożoność jest, z dokładnością do mnożenia podwektorów elementarnych przez bloki elementarne tworzące diagonalne bloki poprawy uwarunkowania i uwzględnienia wektora prawej strony, równa sumie liczby operacji mnożenia macierz–wektor danej przez (2.99) i liczby operacji rozwiązania problemów lokalnych równej:

$$\sum_{K=1}^{N_{\mathrm{bl}}^{\mathrm{P}}} 2(N_{\mathrm{ssB}}^{K})^2 \tag{2.106}$$

Jeśli bloki poprawy uwarunkowania nakładają się na siebie, liczba operacji może się znacznie, nawet kilkadziesięciokrotnie, zwiększyć.

Złożoność pamięciowa wygładzania błędu metodami dekompozycji obszaru jest determinowana przez koszt przechowywania diagonalnych bloków poprawy uwarunkowania macierzy układu używanych w algorytmach. Mnożenie poziomych pasm macierzy $\mathbf{A}$ przez globalne wektory realizowane jest przy wykorzystaniu przechowywania macierzy układu w postaci bloków elementarnych. Koszt ten przypisany jest w niniejszej analizie obliczaniu residuum układu równań. W trakcie rozwiązywania problemów lokalnych w pętli po podwektorach poprawy uwarunkowania nie wykorzystuje się pozadiagonalnych bloków poprawy uwarunkowania. W metodach dekompozycji obszaru nie muszą być one tworzone. Koszt przechowania diagonalnych bloków poprawy uwarunkowania wynosi:

$$\sum_{K=1}^{N_{\mathrm{bl}}^{\mathrm{P}}} (N_{\mathrm{ssB}}^{K})^2 \tag{2.107}$$

W przypadku dużych, nakładających się na siebie bloków poprawy uwarunkowania koszt ten może znacznie przewyższać koszt przechowywania oryginalnej macierzy $\mathbf{A}$. Uzasadnieniem ponoszenia tak znacznych wydatków obliczeniowych może stać się konieczność zagwarantowania lub istotnego poprawienia całościowej zbieżności solwera.

Złożoność obliczeniowa wygładzania błędu z zastosowaniem niekompletnego rozkładu LU

Złożoność algorytmu wygładzania błędu z zastosowaniem niekompletnego blokowego rozkładu LU zależy od sposobu konstrukcji bloków poprawy uwarunkowania. W większości przypadków nie zachodzi potrzeba użycia bloków większych niż bloki elementarne [29], dlatego analiza zostanie przeprowadzona dla tego właśnie przypadku.

Na podstawie algorytmu `FR-BS` z s. 84 widać, że wygładzanie błędu z zastosowaniem rozkładu ILU(0) odbywa się w dwóch pętlach po wszystkich podwektorach poprawy uwarunkowania. Dla każdego podwektora o określonym indeksie (identyfikatorze) wykonuje się iteracje po sąsiednich podwektorach poprawy uwarunkowania, w jednym przypadku tylko o indeksach mniejszych, a w drugim tylko o indeksach większych. Na zakończenie mnoży się podwektor poprawy uwarunkowania przez diagonalny blok poprawy uwarunkowania (będący odwrotnością bloku diagonalnego w oryginalnej macierzy). W efekcie liczba koniecznych operacji jest równa liczbie operacji dla mnożenia macierz–wektor (2.103). Koszt pamięciowy jest równy kosztowi przechowania oryginalnej macierzy $\mathbf{A}$.

Przygotowanie macierzy $\mathbf{A}$ do przeprowadzenia wygładzanie błędu poprzez realizację algorytmu `ILU(0)` z s. 83 jest, przy takim samym rozmiarze bloków poprawy uwarunkowania, bardziej czasochłonne niż przeprowadzenie rozkładu lub odwracanie diagonalnych bloków poprawy uwarunkowania, wymagane w algorytmie blokowym Gaussa-Seidla. Złożoność czasowa tej operacji, zgodnie z algorytmem `ILU(0)` i założeniem, że bloki poprawy uwarunkowania pokrywają się z blokami elementarnymi, wynosi:

$$\sum_{i=1}^{N_{\mathrm{bl}}^{\mathrm{E}}} N_{\mathrm{ssb}}^{i} \left(\frac{8}{3} (N_{\mathrm{ssb}}^{i})^2 + \sum_{k=1}^{N_{\mathrm{sbl},<i}^{i}} 2 N_{\mathrm{ssb}}^{\mathrm{gls}(i,k)} \left(N_{\mathrm{ssb}}^{\mathrm{gls}(i,k)} + \sum_{j=1}^{N_{\mathrm{sbl},>k}^{k}} N_{\mathrm{ssb}}^{\mathrm{gls}(j',k)} \right) \right) \quad (2.108)$$

gdzie $j' = N_{\mathrm{sbl},<k}^{k} + j$. W powyższym wzorze założono, że bloki sąsiadujące z danym k-tym blokiem elementarnym są tak pogrupowane, że najpierw na liście znajduje się $N_{\mathrm{sbl},<k}^{k}$ bloków o globalnych identyfikatorach mniejszych od k, a następnie pozostałe $N_{\mathrm{sbl},>k}^{k}$ bloków sąsiadujących.

2.7.9 Analiza złożoności obliczeniowej rozwiązywania układów równań liniowych – przypadek szczególny jednorodnej aproksymacji nieciągłej

Rozważanym przypadkiem szczególnym jest zastosowanie solwera przy aproksymacji nieciągłej Galerkina wykorzystującej równomierną siatkę pryzmatycznych elementów skończonych o jednorodnym stopniu aproksymacji p. Siatka taka może powstać np. przez szereg równomiernych adaptacji siatki z rys. 2.8.

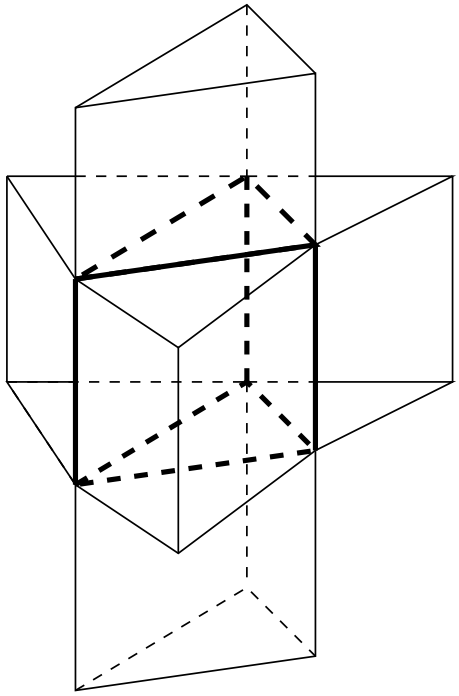
Porównywane są: wykorzystanie blokowego niekompletnego rozkładu ILU(0) opartego na blokach elementarnych i różne strategie konstruowania bloków poprawy uwarunkowania macierzy dla blokowej metody Gaussa-Seidla. Konstrukcja bloków poprawy uwarunkowania związana jest z podziałem obszaru obliczeniowego na małe podobszary – łaty elementów.

Każdy blok poprawy uwarunkowania odpowiada parze podwektorów poprawy uwarunkowania. Każdy podwektor poprawy uwarunkowania grupuje stopnie swobody związane z kilkoma obiektami siatki. Dla nieciągłej dyskretyzacji Galerkina jedynymi obiektami siatki, z którymi wiąże się stopnie swobody, są elementy. Grupa obiektów siatki przyporządkowana pojedynczemu podwektorowi poprawy uwarunkowania jest więc pewnym zbiorem elementów. Elementy te dla efektywności poprawy uwarunkowania powinny sąsiadować z sobą tworząc łatę elementów.

Na początek rozważane są dwa przypadki: w pierwszym, podwektory poprawy uwarunkowania są podwektorami elementarnymi związanymi z pojedynczymi elementami, w drugim, podwektory poprawy uwarunkowania związane są z podobszarami przedstawionymi na rys. 2.17. W tym ostatnim przypadku zakłada się, że łatę elementów (podwektor poprawy uwarunkowania) tworzy się dla każdego elementu siatki oraz że siatka zawiera na tyle dużo elementów, że przy szacowaniu złożoności można pominąć te elementy na brzegu, dla których łata bedzie miała kształt inny niż przedstawiony na rys. 2.17. Dzięki pierwszemu z założeń we wszystkich rozważanych strategiach konstrukcji bloków poprawy uwarunkowania ich liczba jest równa liczbie elementów, będącej także liczbą bloków elementarnych, $N_{\mathrm{bl}}^{\mathrm{P}} = N_{\mathrm{bl}}^{\mathrm{E}} = N_{\mathrm{E}}$.

Liczby stopni swobody charakteryzujące podwektory i bloki poprawy uwarunkowania oraz ostateczne złożoności algorytmów wygładzania błędu dla poszczególnych przypadków wyglądają następująco:

- algorytm blokowej metody Gaussa-Seidla z blokami elementarnymi użytymi do poprawy uwarunkowania:



Rys. 2.17. Podobszar obszaru obliczeniowego – łata elementów – wykorzystywany do poprawy uwarunkowania macierzy układów równań liniowych dla nieciągłej aproksymacji Galerkina (linie pogrubione – pojedynczy element pryzmatyczny, linie cienkie – wszyscy jego sąsiedzi)

- $N_{\mathrm{bl}}^{K} = 1$,
- $N_{\mathrm{ssB}}^{K} = N_{\mathrm{ssb}}^{i} = \frac{1}{2}(p+1)^2(p+2) = N_{\mathrm{K}}$,
- $N_{\mathrm{sbl,off}}^{i} = 5$,
- złożoność czasowa przygotowania macierzy do wygładzania:

$$\frac{2}{3}N_{\mathrm{K}}^{3}N_{\mathrm{E}} = \frac{2}{3}N_{\mathrm{K}}^{2}N \tag{2.109}$$

- złożoność czasowa wygładzania błędu:

$$(12N_{\mathrm{K}}^{2} + N_{\mathrm{K}})N_{\mathrm{E}} = (12N_{\mathrm{K}} + 1)N \tag{2.110}$$

- złożoność pamięciowa:

$$N_{\mathrm{K}}^{2}N_{\mathrm{E}} = N_{\mathrm{K}}N \tag{2.111}$$

- algorytm blokowej metody Gaussa-Seidla z blokami poprawy uwarunkowania odpowiadającymi podobszarom z rys. 2.17:
  - $N_{\mathrm{bl}}^K = 6$,
  - $N_{\mathrm{ssB}}^K = 6N_{\mathrm{ssb}}^i = 3(p+1)^2(p+2) = 6N_{\mathrm{K}}$,
  - $N_{\mathrm{sbl,off}}^i = 3\frac{1}{3}$ (wartość średnia, pięć tworzących elementów ma po czterech sąsiadów poza blokiem, element środkowy nie ma żadnego),
  - złożoność czasowa przygotowania macierzy do wygładzania:
$$144N_{\mathrm{K}}^3 N_{\mathrm{E}} = 144N_{\mathrm{K}}^2 N \tag{2.112}$$
  - złożoność czasowa wygładzania błędu:
$$(112N_{\mathrm{K}}^2 + 6N_{\mathrm{K}})N_{\mathrm{E}} = (112N_{\mathrm{K}} + 6)N \tag{2.113}$$
  - złożoność pamięciowa:
$$36N_{\mathrm{K}}^2 N_{\mathrm{E}} = 36N_{\mathrm{K}} N \tag{2.114}$$
- algorytm wykorzystujący blokowy niekompletny rozkład LU z blokami elementarnymi, jako blokami poprawy uwarunkowania:
  - $N_{\mathrm{bl}}^K = 1$,
  - $N_{\mathrm{ssB}}^K = N_{\mathrm{ssb}}^i = \frac{1}{2}(p+1)^2(p+2) = N_{\mathrm{K}}$,
  - $N_{\mathrm{sbl},<i}^i = 2.5$ (wartość uśredniona dla siatek regularnych takich jak na rys. 2.8; jeśli obliczenia odbywają się w naturalnej kolejności wzdłuż linii współrzędnych przestrzennych, elementy mają na przemian dwóch i trzech sąsiadów o indeksach mniejszych od własnego),
  - złożoność czasowa przygotowania macierzy do wygładzania:
$$\frac{242}{12}N_{\mathrm{K}}^3 N_{\mathrm{E}} = 20\frac{1}{6}N_{\mathrm{K}}^2 N \tag{2.115}$$
  - złożoność czasowa wygładzania błędu:
$$(12N_{\mathrm{K}}^2 + N_{\mathrm{K}})N_{\mathrm{E}} = (12N_{\mathrm{K}} + 1)N \tag{2.116}$$
  - złożoność pamięciowa:
$$6N_{\mathrm{K}}^2 N_{\mathrm{E}} = 6N_{\mathrm{K}} N \tag{2.117}$$

Ze względu na znacznie zwiększoną liczbę operacji w przypadku użycia dużych bloków poprawy uwarunkowania z rys. 2.17 rozważa się także przypadek zmodyfikowany. Łaty elementów tworzy się dla każdego elementu, ale dołącza się do nich tylko te elementy, dla których łata nie została jeszcze utworzona. Łaty tworzone dla siatek regularnych w naturalnej kolejności, opisanej dla rozkładu LU, mają na przemian trzy lub cztery elementy. Prowadzi to do następujących charakterystyk metody:

- $N_{\mathrm{bl}}^{K} = \frac{7}{2}$,
- $N_{\mathrm{ssB}}^{K} = \frac{7}{2} N_{\mathrm{ssb}}^{i} = \frac{7}{4}(p+1)^2(p+2) = 3\frac{1}{2} N_{\mathrm{K}}$,
- $N_{\mathrm{sbl,off}}^{i} = \frac{43}{12}$ – wartość uśredniona dla siatek regularnych,
- złożoność czasowa przygotowania macierzy do wygładzania:

$$\frac{325}{12} N_{\mathrm{K}}^3 N_{\mathrm{E}} = 27 \frac{1}{12} N_{\mathrm{K}}^2 N \tag{2.118}$$

- złożoność czasowa wygładzania błędu:

$$(\frac{595}{12} N_{\mathrm{K}}^2 + N_{\mathrm{K}}) N_{\mathrm{E}} = (49 \frac{7}{12} N_{\mathrm{K}} + 1) N \tag{2.119}$$

- złożoność pamięciowa:

$$\frac{49}{4} N_{\mathrm{K}}^2 N_{\mathrm{E}} = 12 \frac{1}{4} N_{\mathrm{K}} N \tag{2.120}$$

Przy omawianiu procedur służących do realizacji metody wielosiatkowej, jako istotne z punktu widzenia obliczeniowego, wymienione zostały w p. 2.7.8 także poniższe dodatkowe operacje, których złożoność czasowa dla omawianego szczególnego przypadku wynosi:

- obliczenie residuum – złożoność równa $(12N_{\mathrm{K}}^2 + N_{\mathrm{K}})N_{\mathrm{E}} = (12N_{\mathrm{K}} + 1)N$, na podstawie wzorów (2.100);
- rzutowanie residuum na siatkę rzadką – liczba niezerowych elementów w pojedynczym wierszu odpowiadającym funkcji kształtu na siatce gęstej jest równa liczbie funkcji kształtu w pojedynczym elemencie siatki rzadkiej, w omawianym przypadku równej N_{K}. Złożoność restrykcji wynosi więc $N_{\mathrm{K}}N$;
- rozszerzenie rozwiązania na siatce rzadkiej do przestrzeni związanej z siatką gęstą – złożoność prolongacji jest identyczna ze złożonością restrykcji.

2.7.10 Wnioski

Podstawą omawianych algorytmów iteracyjnego rozwiązywania układów równań liniowych jest pojedyncza pętla wygładzania błędu na zadanej siatce, realizowana jako sekwencja rozwiązań problemów lokalnych (2.56). Złożoność czasowa tej operacji jest rzędu liczby stopni swobody związanych z siatką. W przypadku stosowania bardzo ogólnych strategii adaptacji, dla których jedynym ograniczeniem jest nieprzekraczanie założonego maksymalnego stopnia aproksymacji, warunkiem otrzymania optymalnej, rzędu N, złożoności procedury rozwiązywania układów równań liniowych jest utrzymanie stałej liczby iteracji gwarantujących zbieżność solwera liniowego. W praktyce jedynymi metodami rozwiązywania równań liniowych zapewniającymi dla powyższych strategii adaptacji zbieżność przy liczbie iteracji stałej lub nieznacznie rosnącej są metody wielosiatkowe. Zaprezentowane w pracy szczegółowe algorytmy należą więc do wąskiej grupy algorytmów wykazujących dla szeregu problemów złożoność równą lub bliską optymalnej. Ich optymalna złożoność ma szczególne znaczenie dla obliczeń wielkiej skali, gdy liczba stopni swobody przekracza milion.

Innym problemem, poruszanym już wcześniej, jest zależność złożoności obliczeniowej rozważanych solwerów równań liniowych od stopnia aproksymacji. Zależność ta wynika z wpływu stopnia aproksymacji na zbieżność solwera i złożoności pojedynczej operacji wygładzania błędu, determinującej złożoność pojedynczych iteracji solwera. Podnoszenie stopnia aproksymacji z reguły osłabia zbieżność solwera, jednak wpływ ten zależy w dużym stopniu od charakteru problemu i rodzaju aproksymacji.

Wpływ stopnia aproksymacji na złożoność wygładzania błędu jest określony wzorami wyprowadzonymi w pp. 2.7.8 i 2.7.9. W obu przypadkach charakter zależności jest podobny, jednak znacznie bardziej wyrazisty jest w przypadku wzorów wyprowadzonych dla konkretnego przypadku szczegółowego (p. 2.7.8). Złożoność pojedynczej pętli wygładzania jest tutaj kwadratową funkcją liczby stopni swobody w pojedynczym elemencie.

W przypadku ogólnym (p. 2.7.9) bloki elementarne związane z obiektami siatki należacymi do pojedynczego elementu sąsiadują z sobą w sensie tworzenia niezerowych wyrazów w wierszach macierzy układu. Stąd liczba niezerowych wyrazów w pojedynczym wierszu macierzy układu jest liniową funkcją liczby stopni swobody w pojedynczym elemencie (będącej sumą liczb stopni swobody związanych z poszczególnymi obiektami siatki tworzącymi element). W konsekwencji złożoność wygładzania jest ponownie kwadratową funkcją liczby stopni swobody w elemencie.

Różne strategie konstrukcji bloków poprawy uwarunkowania mogą zmienić złożoność obliczeniową wygładzania o pewien stały czynnik, niezależny od stopnia aproksymacji i całkowitej liczby stopni swobody w problemie. Podobnie ma się sprawa z przygotowaniem macierzy do wygładzania. Tym razem jednak czynnik zwiększający złożoność czasową jest proporcjonalny do kwadratu czynnika zwiększającego złożoność czasową wygładzania.

Tłumacząc to na zależności od stopnia aproksymacji, otrzymuje się w przypadku trójwymiarowym złożoność pamięciową rzędu p^3 oraz złożoność czasową wygładzania rzędu p^3 i złożoność czasową przygotowania do wygładzania rzędu p^6. Zależności te należy uwzględnić dobierając całościową stretegię aproksymacji MES. Zyski zwiększonej dokładności, wynikające z zastosowania aproksymacji wyższego rzędu, nawet tak znaczne, jak prezentowane w p. 2.6.6 dla gładkiego problemu w regularnym obszarze, są przeciwważone istotnie wzrastającymi nakładami obliczeniowymi: rzędu co najmniej p^3 dla czasowej złożoności całkowania numerycznego (lub nawet p^6 w przypadku zastosowania nieoptymalnych procedur standardowych) i rzędu co najmniej p^6 dla czasowej złożoności rozwiązywania układów równań liniowych, nawet przy zastosowaniu optymalnych metod wielosiatkowych z opisywanymi w niniejszej pracy blokowymi algorytmami wygładzania błędu. Istotny, a w wielu wypadkach decydujący, jest także rząd złożoności pamięciowej solwera równań liniowych wynoszący p^3.

Dla metod nieblokowych, takich jak standardowe iteracje proste Jacobiego, Gaussa-Seidla czy SSOR, poprawa uwarunkowania jest słabsza i może nie gwarantować zbieżności. Jednak w wypadku jej uzyskania wzrost złożoności czasowej jest rzędu tylko p^3, przy stałej złożoności pamięciowej.

2.7.11 Weryfikacja eksperymentalna analiz złożoności obliczeniowej algorytmów rozwiązywania układów równań liniowych

Weryfikacja przedstawionych uprzednio analiz złożoności obliczeniowej solwerów równań liniowych dokonana jest dla omawianego w p. 2.6.6 przykładu zastosowania nieciągłej aproksymacji Galerkina do rozwiązania równania Laplace'a w sześcianie jednostkowym. Rozważana jest sekwencja rozwiązań dla różnych stopni aproksymacji p oraz siatek otrzymanych poprzez równomierne zagęszczanie siatki z rys. 2.8. Każdorazowo używana jest restartowana metoda GMRES z maksymalnie dwudziestoma wektorami Kryłowa.

Tablice 2.7 i 2.8 przedstawiają porównanie kosztów obliczeniowych dla omawianych uprzednio wersji solwera blokowego oraz dwóch stopni aproksymacji

Tablica 2.7. Charakterystyki solwerów równań liniowych w przypadku zastosowania nieciągłej aproksymacji Galerkina stopnia $p = 2$ do rozwiązania równania Laplace'a w sześcianie jednostkowym

Solwer		Liczba stopni swobody			
		288	2304	18 432	147 456
SBGS-s	liczba iteracji	18	40	91	193
	pamięć [MBajt]	0.25	2.35	20.08	165.73
	czas obliczeń [s]	0.082	0.20	4.46	84.12
SBGS-l	liczba iteracji	9	13	18	30
	pamięć [MBajt]	0.49	5.22	47.31	401.68
	czas obliczeń [s]	0.014	0.23	2.92	40.96
SBILU	liczba iteracji	10	15	25	57
	pamięć [MBajt]	0.37	3.62	31.51	262.22
	czas obliczeń [s]	0.013	0.18	3.25	152.32
MBGS-s	liczba iteracji	17	498	∞	∞
	pamięć [MBajt]	0.30	2.60	22.32	185.10
	czas obliczeń [s]	0.023	6.32	—	—
MBGS-l	liczba iteracji	8	11	12	12
	pamięć [MBajt]	0.54	5.72	52.66	451.38
	czas obliczeń [s]	0.020	0.31	3.23	28.73
MBILU	liczba iteracji	8	11	13	13
	pamięć [MBajt]	0.42	4.00	35.14	294.40
	czas obliczeń [s]	0.019	0.25	2.90	47.80

Objaśnienia: S – metody jednosiatkowe, M – metody wielosiatkowe, B – metody blokowe, GS – uogólniony algorytm Gaussa-Seidla, ILU – algorytm ILU(0), s – wykorzystanie bloków elementarnych, l – wykorzystanie bloków odpowiadających zmodyfikowanym łatom elementów (szczegółowy opis metod w tekście).

Tablica 2.8. Charakterystyki solwerów równań liniowych w przypadku zastosowania nieciągłej aproksymacji Galerkina stopnia $p = 5$ do rozwiązania równania Laplace'a w sześcianie jednostkowym (objaśnienia jak dla tabl. 2.7)

Solwer		Liczba stopni swobody		
		2016	16 128	129 024
SBGS-s	liczba iteracji	99	∞	∞
	pamięć [MBajt]	10.07	96.11	830.49
	czas obliczeń [s]	1.08	—	—
SBGS-l	liczba iteracji	9	16	53
	pamięć [MBajt]	21.71	235.77	2157.20
	czas obliczeń [s]	1.39	14.57	166.33
SBILU	liczba iteracji	12	19	78
	pamięć [MBajt]	15.89	158.13	1388.65
	czas obliczeń [s]	1.36	13.14	156.81
MBGS-s	liczba iteracji	23	79	∞
	pamięć [MBajt]	11.00	106.80	934.51
	czas obliczeń [s]	1.17	25.87	—
MBGS-l	liczba iteracji	8	9	9
	pamięć [MBajt]	23.01	258.46	2412.88
	czas obliczeń [s]	1.54	18.18	154.78
MBILU	liczba iteracji	9	11	11
	pamięć [MBajt]	17.06	174.87	1560.74
	czas obliczeń [s]	1.58	17.06	145.58

$p = 2$ i $p = 5$. Porównanie obejmuje liczbę iteracji koniecznych do osiągnięcia względnej redukcji residuum o 10^{-6}, rozmiar pamięci zajmowanej przez strukturę danych solwera (przede wszystkim macierz układu, macierz poprawy uwarunkowania i 20 wektorów Kryłowa) oraz czas rozwiązania układu równań, bez czasu tworzenia macierzy układu (całkowania numerycznego). Liczba iteracji ∞ oznacza brak zbieżności solwera.

Porównywanymi metodami poprawy uwarunkowania macierzy układu są:

- SBGS-s – jednosiatkowa blokowa metoda Gaussa-Seidla z blokami elementarnymi, jako blokami poprawy uwarunkowania,
- SBGS-l – jednosiatkowa blokowa metoda Gaussa-Seidla z blokami po-

prawy uwarunkowania, odpowiadającymi zmodyfikowanym łatom elementów,

- SBILU – jednosiatkowy blokowy niekompletny rozkład LU z blokami elementarnymi, jako blokami poprawy uwarunkowania,
- MBGS-s – wielosiatkowa wersja algorytmu SBGS-s,
- MBGS-l – wielosiatkowa wersja algorytmu SBGS-l,
- MBILU – wielosiatkowa wersja algorytmu SBILU.

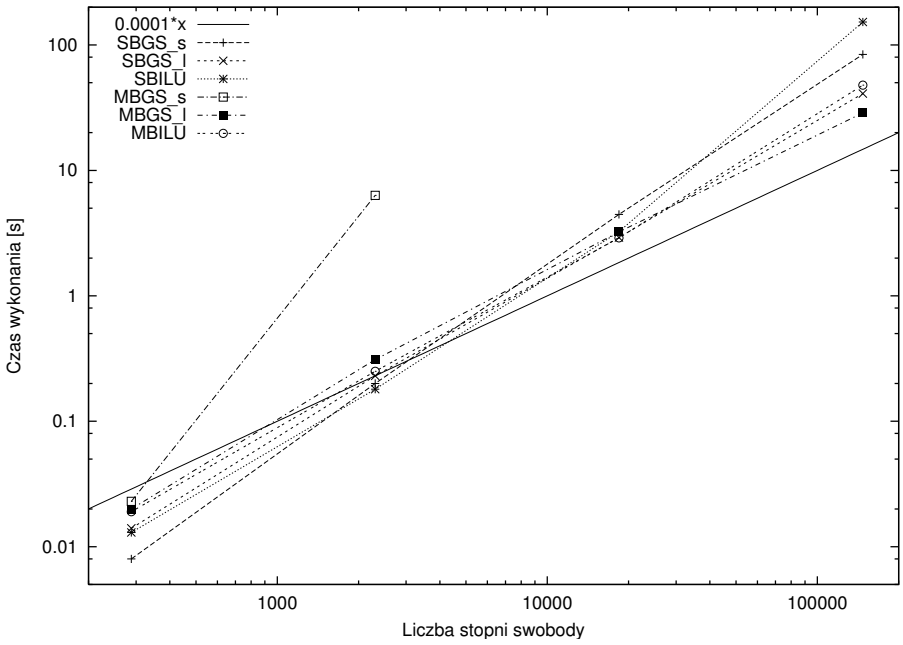
Wyniki pokazują, jak przy zagęszczaniu siatki wzrasta liczba iteracji koniecznych do osiągnięcia zbieżności w przypadku metod jednosiatkowych. Dla metod wielosiatkowych liczba ta pozostaje względnie stała (jej wzrost jest spowodowany stosowaniem tylko jednej iteracji wygładzania dla siatki najdrobniejszej). Jednocześnie zwiększenie rozmiaru wymaganej pamięci przy przejściu od wersji jednosiatkowej do wielosiatkowej w przypadku każdej z metod nie przekracza nigdy kilkunastu procent. Praktyczne obliczenia potwierdzają, że tylko metody wielosiatkowe gwarantują optymalność solwera układów równań liniowych dla obliczeń wielkiej skali.

Ponadto, rezultaty potwierdzają konieczność używania w uogólnionej metodzie Gaussa-Seidla powiększonych bloków poprawy uwarunkowania w przypadku stosowania metody dla nieciągłej dyskretyzacji Galerkina. Przyspieszenie zbieżności w porównaniu z metodami wykorzystującymi bloki elementarne okupione jest w takim wypadku znacznym zwiększeniem rozmiaru wykorzystywanej pamięci.

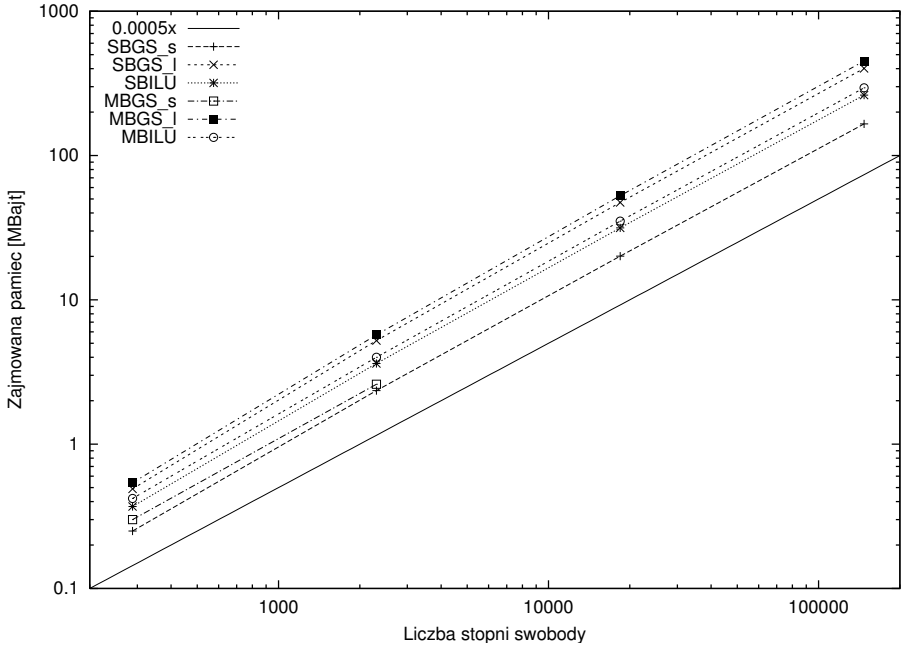
Wykresy zamieszczone na rys. 2.18 i 2.19 obrazują zachowanie solwerów w przypadku stałego stopnia aproksymacji, $p = 2$, i wzrastającej liniowo na skutek jednorodnego zagęszczania siatki liczby stopni swobody. I pod względem rozmiaru wymaganej pamięci, i czasu obliczeń solwery wielosiatkowe wykazują optymalną lub zbliżoną do optymalnej złożoność obliczeniową.

Wykresy przedstawione na rys. 2.20 i 2.21 podsumowują koszty obliczeniowe związane z nieciągłą aproksymacją Galerkina i solwerem GMRES, wykorzystującym poprawę uwarunkowania metodą MBGS-l, zastosowanymi do rozwiązania zadania Laplace'a w sześcianie jednostkowym. Podobne wyniki można uzyskać dla drugiego rozważanego solwera liniowego – metody GMRES z poprawą uwarunkowania MBILU.

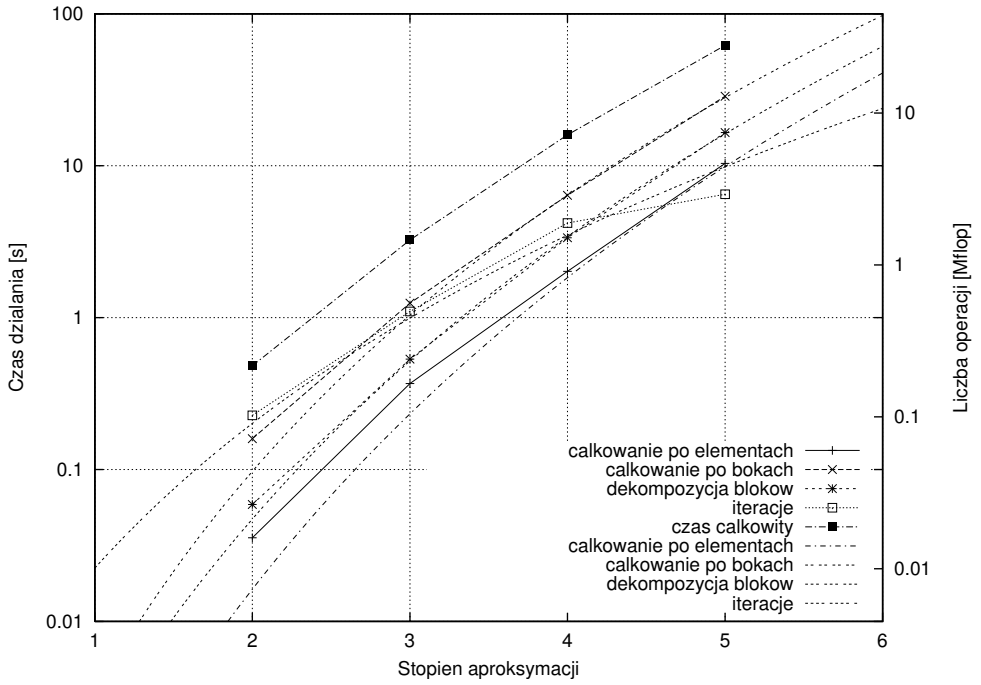
Pierwszy z wykresów (rys. 2.20) przedstawia wpływ zmiany stopnia aproksymacji na czasy realizacji poszczególnych etapów obliczeń w trakcie symu-



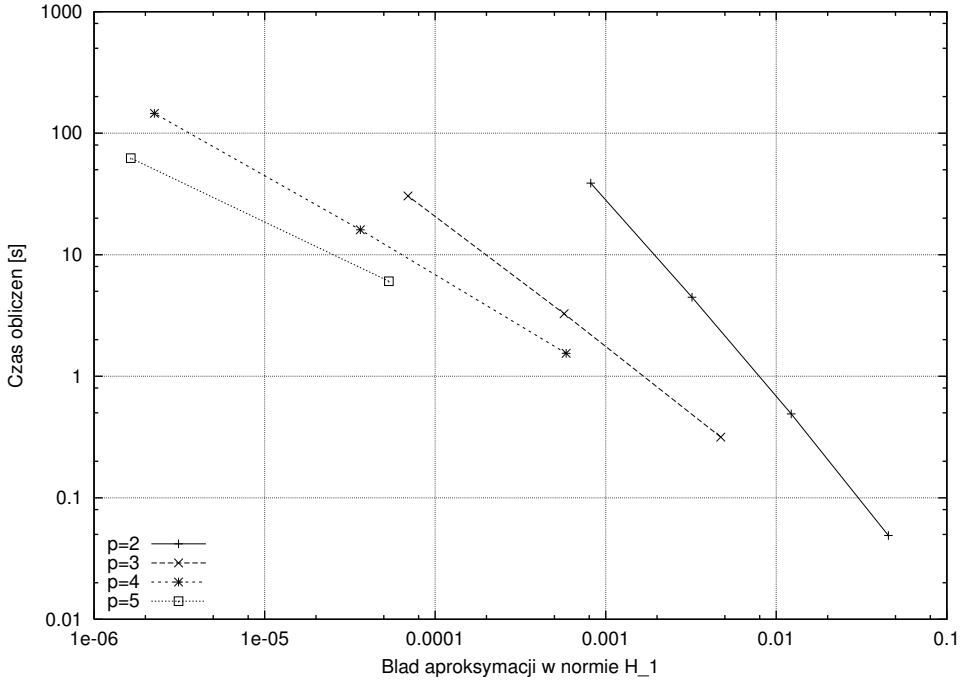
Rys. 2.18. Czas rozwiązania układu równań liniowych w funkcji liczby stopni swobody w przypadku zastosowania nieciągłej aproksymacji Galerkina stopnia $p = 2$ do rozwiązania równania Laplace'a w sześcianie jednostkowym (opis metod jak dla tabl. 2.7)



Rys. 2.19. Wymagany rozmiar pamięci dla solwera równań liniowych w funkcji liczby stopni swobody w przypadku zastosowania nieciągłej aproksymacji Galerkina stopnia $p = 2$ do rozwiązania równania Laplace'a w sześcianie jednostkowym (opis metod jak dla tabl. 2.7)



Rys. 2.20. Złożoność czasowa poszczególnych etapów obliczeń w przypadku zastosowania nieciągłej aproksymacji Galerkina do rozwiązania równania Laplace'a w sześcianie jednostkowym, wyniki uzyskane przy tej samej liczbie elementów i zmieniającym się stopniu aproksymacji; krzywe z symbolami – wyniki eksperymentalne, krzywe bez symboli – wyniki teoretyczne odpowiadające wzorom (2.53), (2.54), (2.118) i (2.119) (solwer GMRES + MBGS-l)



Rys. 2.21. Ilustracja opłacalności użycia nieciągłej aproksymacji Galerkina wyższego stopnia do rozwiązania zagadnienia Laplace'a w sześcianie jednostkowym przedstawiona w postaci zależności czasu obliczeń od uzyskanego błędu aproksymacji dla różnych stopni aproksymacji p (solwer GMRES + MBGS-l)

lacji z tą samą liczbą elementów (co oznacza, że wzrost stopnia aproksymacji powoduje wzrost liczby stopni swobody i zmniejszenie błędu aproksymacji). Wyniki eksperymentów nałożonone są na wykresy krzywych uzyskanych analitycznie. Wykres ukazuje złożoność czasową całkowania numerycznego po wnętrzach elementów, całkowania numerycznego po bokach elementów, przygotowania macierzy poprawy uwarunkowania oraz 10 iteracji GMRES. Widać, jak dla przedstawionego zakresu zmienności p zmienia się procentowy udział poszczególnych etapów w całości obliczeń. Dla niskich p dominuje czas iteracji, następnie wyższy rząd złożoności całkowania numerycznego i przygotowania macierzy poprawy uwarunkowania powoduje zwiększenie procentowego udziału tych ostatnich. Dla wysokich p czas całkowania numerycznego jest ponad dwukrotnie dłuższy od czasu rozwiązywania układu równań.

Wykres na rys. 2.21 pokazuje, dla szczególnego przypadku gładkiego rozwiązania i regularnego obszaru, opłacalność stosowania aproksymacji różnych stopni. Pomimo istotnego wzrostu nakładów obliczeniowych dla pojedynczego elementu, aproksymacja wyższego stopnia zawsze pozwala uzyskać założony poziom błędu rozwiązania w krótszym czasie niż aproksymacja niższego stopnia.

Dla innych problemów obraz ten mogą zmienić dwa czynniki: szybkość zbieżności solwera w przypadku aproksymacji różnych stopni i szybkość zbieżności rozwiązania przybliżonego MES do rozwiązania dokładnego. Pozostałe elementy, tzn. nakłady obliczeniowe na całkowanie numeryczne i nakłady na pojedynczą iterację solwera liniowego pozostaną takie, jak wyprowadzone dla niniejszego przykładu.

2.8 Adaptacja

Adaptacja jest niezwykle ważnym etapem całościowej procedury aproksymacji metodą elementów skończonych. Dokonuje się jej w celu uzyskania optymalnej siatki MES oraz, w metodzie p i hp, optymalnego rozkładu stopnia aproksymacji tak, aby osiągnąć założony poziom błędu rozwiązania jak najmniejszym kosztem obliczeniowym.

Istnieje wiele metod szacowania *a posteriori* błędu aproksymacji i opartych na nich strategii adaptacji [142]. Ze względu na różnorodność tych metod i ich znaczne zróżnicowanie, jeśli chodzi o zakres stosowalności i złożoność obliczeniową, proces szacowania błędu nie jest przedmiotem szczegółowych analiz w niniejszej pracy. Analizowane są natomiast wszelkie konsekwencje, jakie dla pozostałych faz obliczeń z użyciem dyskretyzacji przestrzennej MES niesie sto-

sowanie adaptacji. Konsekwencje te obejmują tak dobór odpowiednich metod numerycznych, jak i struktur danych oraz sposobów organizacji obliczeń. Specyfika niniejszej pracy polega na uwzględnieniu różnorodnych metod adaptacji (h, p, hp) przy doborze i analizie algorytmów oraz sposobów ich implementacji w programach MES przeznaczonych do obliczeń wielkiej skali. W tym kontekście istotny jest fakt, że zdecydowana większość stosowanych w praktyce algorytmów szacowania błędu wykorzystuje sekwencję obliczeń lokalnych (obejmujących małe podobszary obszaru obliczeniowego), co prowadzi do liniowych złożoności obliczeniowych względem liczby stopni swobody w problemie.

W przypadku samego procesu modyfikacji siatki, liczba operacji jest liniową funkcją liczby stopni swobody. Podział dowolnego obiektu siatki (jeśli jest możliwy) wykonywany jest w stałej liczbie operacji. Jedynym etapem modyfikacji siatki zależnym od rodzaju aproksymacji jest przypisywanie wartości stopni swobody nowo tworzonym obiektom. Przy klasycznej aproksymacji dokonuje się tego najczęściej poprzez interpolację funkcji na siatce adaptowanej przy wykorzystaniu funkcji bazowych nowo utworzonej siatki. Przy aproksymacji nieciągłej i bardziej złożonych adaptacjach dla aproksymacji ciągłej zachodzi konieczność dokonania jawnej projekcji rozwiązania z siatki adaptowanej na wynikową. Dla stałej liczby stopni swobody w problemie i dyskretyzacjach różniących się stopniem aproksymacji złożoność czasowa projekcji, przypadająca na pojedynczy stopień swobody, jest ponownie kwadratową funkcją liczby stopni swobody w elemencie, a więc jej rząd wynosi w przypadku trójwymiarowym p^6. Operacja projekcji stopni swobody wykorzystywana w adaptacji może być identyczna z operacjami zawężenia lub rozszerzenia stosowanymi w solwerze wielosiatkowym, stąd wszelkie analizy złożoności obliczeniowej tych ostatnich przenoszą się na przypadek adaptacji.

2.9 Podsumowanie – analiza złożoności obliczeniowej algorytmów sekwencyjnych adaptacyjnej MES pod kątem zastosowania w obliczeniach wielkiej skali

Rozważania przedstawione w niniejszym rozdziale miały na celu zanalizowanie algorytmów stosowanych w dyskretyzacji adaptacyjną metodą elementów skończonych pod kątem doboru metod odpowiednich do obliczeń wielkiej skali. Podstawową cechą takich algorytmów jest optymalna złożoność obliczeniowa, złożoność będąca funkcją rozmiaru zadania o najniższym możliwym do uzy-

skania rzędzie. Drugą fundamentalną własnością jest możliwość zrównoleglenia bez utraty optymalnej złożoności obliczeniowej. Zagadnieniom realizacji równoległej poświęcony jest następny rozdział.

W przypadku algorytmów stosowanych w symulacjach adaptacyjną metodą elementów skończonych, optymalność złożoności obliczeniowej polega najczęściej na liniowej zależności rozmiaru wymaganej pamięci i liczby operacji od liczby stopni swobody w problemie. Analizy przeprowadzone w niniejszym rozdziale obejmowały szereg algorytmów dyskretyzacji czasowej, rozwiązywania układów równań nieliniowych, całkowania numerycznego i rozwiązywania układów równań liniowych. Wśród tych metod cechami optymalności dla szerokiej grupy zagadnień wykazują się:

- niejawne nieliniowe schematy całkowania czasowego, dla których zagęszczanie siatki i wzrost liczby stopni swobody nie prowadzi automatycznie do skrócenia kroku czasowego i zwiększenia całkowitej liczby kroków czasowych w symulacji,

- warianty metody Newtona rozwiązywania układów równań nieliniowych, w których wydajny schemat iteracyjny z odpowiednią aproksymacją macierzy jakobianowej łączy się z efektywną procedurą rozwiązywania układów równań liniowych,

- wielosiatkowe solwery rzadkich układów równań liniowych, które są najbliższe osiągnięcia optymalnej złożoności obliczeniowej względem rozmiaru układu równań,

- strategie i procedury adaptacji (oszacowania błędu oraz modyfikacji siatki i przypisania wartości stopni swobody) korzystające wyłącznie z obliczeń lokalnych.

Rozdział 3

Algorytmy równoległej implementacji metody elementów skończonych

Problem zrównoleglenia obliczeń dowolnego rodzaju polega na dekompozycji całości zadania obliczeniowego na szereg podzadań wykonywanych przez pojedyncze procesory. Zasady dekompozycji mogą być różne zależnie od specyfiki zadania. W klasycznych obliczeniach macierzowych dekompozycja jest często oparta na dekompozycji struktury danych lub dekompozycji pętli po wierszach i kolumnach macierzy [106]. Naturalnym sposobem zrównoleglenia obliczeń dla metody elementów skończonych jest wykorzystanie dekompozycji obszaru obliczeniowego. Pierwsze prace teoretyczne na ten temat powstały jeszcze w XIX wieku [159], choć oczywiście nie w kontekście obliczeń automatycznych. Idea metod Schwarza polega na iteracyjnym rozwiązywaniu równań różniczkowych cząstkowych. Obszar obliczeniowy dzieli się na dwa nakładające się podobszary i inicjuje rozwiązanie w każdym z nich. Zadanie rozwiązuje się kolejno w każdym podobszarze, biorąc za warunek na brzegu wewnętrznym pomiędzy podobszarami wartości aktualnego rozwiązania w podobszarze sąsiednim. Pętle po podobszarach powtarza się aż do uzyskania zbieżności rozwiązania.

Pierwotne koncepcje Schwarza zostały w latach osiemdziesiątych i dziewięćdziesiątych XX w. uogólnione na przypadki zastosowań numerycznych, z wieloma podobszarami, z możliwym nienakładaniem się podobszarów i innymi modyfikacjami [143]. Najpopularniejszym obecnie zastosowaniem metod Schwarza jest, omówiona w p. 2.7.1, metoda rozwiązywania układów równań liniowych, będąca uogólnieniem metod iteracji prostej [162].

Metoda dekompozycji obszaru stanowi podstawę opisywanej w niniejszym rozdziale metodologii zrównoleglenia algorytmów stosowanych w symulacjach z użyciem dyskretyzacji przestrzennej metodą elementów skończonych. Szczególny nacisk położony jest na uzyskanie *wydajnych* programów równoległych, w konstrukcji których uwzględnione są własności systemów komputerowych, na których dokonuje się obliczeń.

3.1 Docelowa architektura sprzętu i metodologia programowania

Istotnym czynnikiem, od którego zależy sposób zrównoleglenia algorytmów MES, jak i dowolnych innych obliczeń, jest założona docelowa architektura systemu komputerowego i związana z nią metodologia programowania. Obliczenia wykorzystujące dekompozycję obszaru w naturalny sposób odwzorowują się na systemy wieloprocesorowe z pamięcią rozproszoną. Każdy z podobszarów przyporządkowany jest pojedynczemu procesorowi. Z każdym z podobszarów związana jest odpowiednia struktura danych. Składniki tej struktury przechowywane są w lokalnej pamięci procesora. Tworzy się jednoznaczne przyporządkowanie: podobszar–procesor–lokalna struktura danych–lokalna pamięć.

Dla systemów z pamięcią rozproszoną naturalną metodologią programowania jest wykorzystanie przesyłania komunikatów. Istnienie *de facto* standardu przesyłania komunikatów w postaci specyfikacji MPI, pozwala na tworzenie przenośnych programów równoległych. Istniejące implementacje MPI [81], coraz częściej stanowiące integralne wyposażenie równoległych systemów komputerowych, potrafią efektywnie wykorzystać specyfikę sprzętu komputerowgo, w tym ewentualne występowanie pamięci wspólnej. Tym samym model programowania z wykorzystaniem przesyłania komunikatów staje się modelem uniwersalnym, przeznaczonym dla dowolnego typu maszyn.

W opisywanej w niniejszej pracy koncepcji zrównoleglenia programów MES wykorzystuje się pojęcie lokalnej pamięci procesora i wymiany komunikatów między procesorami. Wszystkie analizy teoretyczne i praktyczne realizacje dotyczą tego właśnie modelu programowania i wykonania równoległego.

Z wyborem docelowej architektury systemu komputerowego związana jest kolejna cecha obliczeń równoległych – ziarnistość. Ziarnistość odpowiada proporcji czasu przeznaczonego przez procesory na niezależne obliczenia do czasu potrzebnego na komunikację międzyprocesorową oraz częstości dokonywania wymiany komunikatów. Obliczenia drobnoziarniste, o małej liczbie operacji pomiędzy wymianą komunikatów, dobrze pasują do systemów z szybką sie-

cią połączeń międzyprocesorowych. W systemach o powolnej sieci, zwłaszcza systemach o dużej zwłoce przy inicjowaniu transferu (takich jak wszystkie odmiany sieci Ethernet), narzut komunikacyjny w programach drobnoziarnistych staje się istotnym obciążeniem. Dla systemów o dużej zwłoce narzut rośnie nie tyle na skutek dużej liczby przesyłanych danych, ile z powodu dużej liczby wymienianych komunikatów. Aby go zmniejszyć, najważniejsza staje się redukcja liczby komunikatów, nawet jeśli odbywa się to kosztem zwiększenia ich rozmiarów i wzrostu innych narzutów wykonania równoległego (np. związanych z powtarzaniem tych samych obliczeń na różnych procesorach lub ze spowolnieniem zbieżności algorytmów iteracyjnych).

Ze względu na fakt, że dominującą architekturą systemów wieloprocesorowych stają się w ostatnich latach sieci (klastry) komputerów osobistych, a najbardziej ekonomiczną wersją takich sieci jest Ethernet, w metodologii zrównoleglenia programów MES opisanej w niniejszej pracy przyjęto założenie maksymalizacji ziarnistości obliczeń. Oznacza to, że algorytmy modyfikowane są tak, aby maksymalizować czas pomiędzy kolejnymi transferami danych. Zakłada się, że tam gdzie tylko jest to możliwe, dane do przesłania są grupowane i transportowane w ramach pojedynczych komunikatów o dużych rozmiarach.

Zakładana ziarnistość obliczeń ma dodatkowo wpływ na sposób organizacji pracy systemu równoległego. W przypadku obliczeń drobnoziarnistych, dla których liczba pojedynczych niezależnych zadań przekracza znacznie liczbę procesorów, stosuje się często dynamiczne szeregowanie zadań [80], umożliwiające precyzyjną kontrolę obciążenia procesorów. Przy dążeniu do maksymalizacji ziarna obliczeniowego, liczba zadań równoległych nie powinna być większa niż liczba procesorów, a rozmiar zadań przydzielanych poszczególnym procesorom powinien być maksymalizowany. Sytuacja taka odpowiada, wspomnianemu już, naturalnemu odwzorowaniu obliczeń MES wykorzystujących dekompozycję obszaru na systemy komputerowe, o liczbie procesorów równej liczbie podobszarów. W takim przypadku poszczególne zadania (podobszary) są *statycznie* przyporządkowywane procesorom, a zrównoważenie obciążenia osiąga się poprzez odpowiedni dobór rozmiarów podobszarów, które to rozmiary mogą zmieniać się dynamicznie w trakcie realizacji obliczeń.

3.2 Wydajność obliczeń równoległych i cele zrównoleglenia

Dla nowoczesnych jednoprocesorowych systemów komputerowych złożoność obliczeniowa algorytmów nie jest jedynym czynnikiem określającym czas wy-

konania. W niniejszej pracy zwracana była uwaga na takie czynniki, jak maksymalizacja efektywności wykorzystania pamięci podręcznej czy organizacja obliczeń, usprawniająca działanie optymalizujących kompilatorów. W przypadku obliczeń równoległych przybliżenie czasu wykonania miarami złożoności obliczeniowej algorytmów staje się jednocześnie – i trudniejsze, i jeszcze mniej dokładne. Z jednej strony, w analizie pojawiają się dodatkowe zmienne, takie jak liczba procesorów N_p czy parametry określające działanie sieci komunikacyjnej. Z drugiej strony, narzut związany z komunikacją międzyprocesorową zależy nie tylko od dających się łatwo wyrazić teoretycznych parametrów sieci komunikacyjnych, ale także od trudno uchwytnych elementów, takich jak możliwe przepełnienia sieci czy wzajemne odziaływanie współbieżnie wykonywanych obliczeń i przesyłania komunikatów [12]. Dlatego przy określaniu wydajności obliczeń równoległych, prócz analiz teoretycznych, istotną rolę odgrywają eksperymenty numeryczne i związane z nimi pomiary efektywności zrównoleglenia programów.

Istotną i typową cechą obliczeń rozproszonych (wykorzystujących lokalne pamięci procesorów) jest zmniejszenie ograniczeń pamięciowych narzucanych programom. Wiąże się to z często występującą cechą algorytmów (widoczną także dla wielu algorytmów analizowanych w poprzednim rozdziale) polegającą na wyższej złożoności obliczeniowej czasowej nad pamięciową. W miarę wzrastania rozmiaru zadania wymagania czasowe rosną szybciej niż wymagania pamięciowe. Dla dużych zadań pomnażanie liczby procesorów z pamięciami lokalnymi szybciej nasyca zapotrzebowanie na pamięć niż na czas pracy procesora[1]. Z tej przyczyny, w niniejszym rozdziale, w analizach równoległych wersji algorytmów przeznaczonych do obliczeń wielkiej skali, a więc do zadań o dużym rozmiarze, zagadnienia złożoności czasowej algorytmów traktowane są jako istotniejsze od kwestii złożoności pamięciowej[2].

3.2.1 Przyspieszenie obliczeń i efektywność zrównoleglenia

Istnieją dwa podstawowe cele zrównoleglenia obliczeń. Pierwszym z nich jest uzyskanie minimalnego czasu obliczeń dla zadanego problemu i określonej

[1] Nie bez znaczenia jest tu także, charakterystyczny dla ostatnich lat, fakt niskich kosztów pamięci lokalnych i wyposażania pojedynczych procesorów w coraz pojemniejsze pamięci lokalne.

[2] Niezależnie od tego, pomnożenie zasobów pamięciowych systemów równoległych w stosunku do komputerów jednoprocesorowych jest częstokroć istotniejsze dla umożliwienia rozwiązywania dużych zadań niż zwiększanie mocy obliczeniowych. W miarę jednak dalszego zwiększania rozmiaru zadań oraz liczby procesorów efekt różnych rzędów złożoności pamięciowej i czasowej algorytmów zaczyna odgrywać decydującą rolę.

liczby procesorów w maszynie równoległej. Czas taki można próbować oszacować poprzez klasyczną analizę czasowej złożoności obliczeniowej, uwzględniającą dodatkowe czynniki związane z wykonaniem równoległym. Gdy analiza taka jest trudna, co zdarza się często dla złożonych symulacji, można ją zastąpić względną analizą porównującą czas wykonania równoległego z czasem wykonania sekwencyjnego. Taka analiza możliwa jest do wykonania teoretycznie, a jednocześnie można ją weryfikować eksperymentalnie. Typowymi miarami, którymi posługuje się w takich przypadkach, są przyspieszenie obliczeń i efektywność zrównoleglenia.

Teoretycznie, przyspieszenie S_p (*speed-up*) definiuje się jako stosunek czasu T_s wykonania najlepszego algorytmu sekwencyjnego dla danego problemu do czasu T_p równoległego rozwiązania problemu (zakłada się identyczność wszystkich wykorzystywanych procesorów):

$$S_\mathrm{p}(N_\mathrm{p}) = \frac{T_\mathrm{s}}{T_\mathrm{p}(N_\mathrm{p})} \tag{3.1}$$

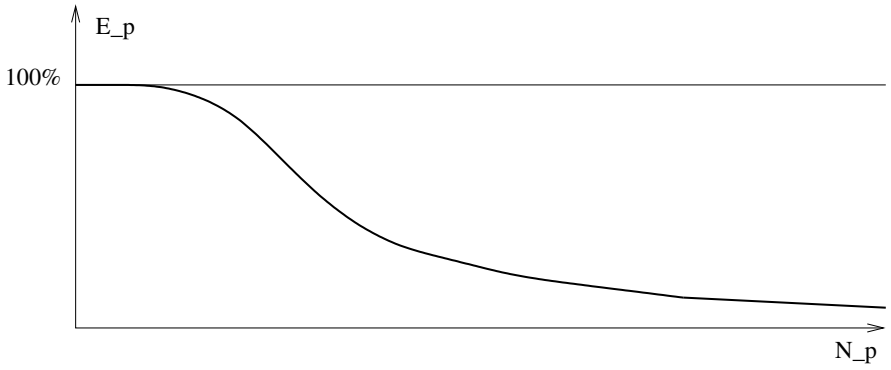
Ideałem jest uzyskanie przyspieszenia liniowego, dla którego czas obliczeń skraca się proporcjonalnie do wzrostu liczby procesorów i w efekcie $S_\mathrm{p} = N_\mathrm{p}$. Ideał ten zakłada pomijalny czas komunikacji międzyprocesorowej.

W praktyce, dla złożonych problemów i złożonych programów, porównuje się najczęściej nie czas dla najlepszego algorytmu sekwencyjnego i czasy realizacji równoległych, ale czasy wykonania tego samego programu na jednym $T_\mathrm{p}(1)$ i na wielu procesorach $T_\mathrm{p}(N_\mathrm{p})$. Niezależnie od uzyskanego przyspieszenia obliczeń, należy wtedy wykazać, że algorytm użyty w programie równoległym jest optymalny dla danego zadania. W przeciwnym wypadku zysk z zastosowania obliczeń równoległych nie jest oczywisty. Przykładowo, jest stosunkowo łatwo uzyskać dobre parametry przyspieszenia w przypadku rozwiązywania układów równań liniowych metodami iteracji prostej, ale metody te są dalekie od optymalnych. Optymalny, sekwencyjny solwer wielosiatkowy może uzyskiwać znacznie lepsze czasy obliczeń. Uzyskanie przyspieszenia obliczeń zbliżonego do liniowego w przypadku takiego solwera jest zadaniem znacznie trudniejszym.

Efektywność zrównoleglenia E_p określa się w procentach, jako stosunek przyspieszenia do liczby procesorów:

$$E_\mathrm{p} = \frac{S_\mathrm{p}}{N_\mathrm{p}} \cdot 100\% \tag{3.2}$$

Idealnemu przyspieszeniu liniowemu odpowiada efektywność zrównoleglenia 100%. W praktyce, przy rozwiązywaniu tego samego zadania z coraz większą



Rys. 3.1. Typowy przebieg funkcji efektywności zrównoleglenia obliczeń E_{p} w zależności od liczby procesorów N_{p}

liczbą procesorów, efektywność prawie zawsze dąży do zera [3]. Wyjaśnieniem tego zjawiska jest z jednej strony fakt istnienia w prawie wszystkich programach równoległych fragmentów sekwencyjnych nie poddających się zrównolegleniu, jak też niemniej ważny, naturalny efekt wzrostu liczby komunikatów międzyprocesorowych przy wzroście liczby procesorów. Typowy przebieg funkcji efektywności w zależności od liczby procesorów przedstawia rys. 3.1.

3.2.2 Skalowalność obliczeń równoległych

Drugim z podstawowych celów zrównoleglenia programów jest umożliwienie rozwiązywania coraz większych zadań, niedostępnych dla maszyn jednoprocesorowych. Postawienie takiego celu pozwala na szukanie równowagi pomiędzy rozmiarem zadania i liczbą procesorów użytych do jego rozwiązania. Nie chodzi już zatem o zwiększanie liczby procesorów dla każdego zadania. Zadania małe rozwiązuje się albo sekwencyjnie, albo stosując małą liczbę procesorów. Pełne możliwości systemów równoległych wykorzystuje się w wypadku zadań prawdziwie wielkiej skali. Nie zapomina się przy tym o efektywności obliczeń. W miejsce jednak bezwzględnego przyspieszenia obliczeń, dla pojedynczego zadania o określonym rozmiarze, wprowadza się nowe miary, uwzględniające zwiększanie rozmiaru zadania.

Jedną z takich miar jest przyspieszenie skalowane (*scaled speed-up*), obliczane przy założeniu jednoczesnego wzrostu i liczby procesorów, i rozmiaru zadania [83]. W celu zdefiniowania przyspieszenia skalowanego wygodne jest wprowadzenie charakterystyki wykonania programu przez komputer zwanej

pracą W (*workload*). Praca systemu komputerowego jest to liczba operacji koniecznych do sekwencyjnej realizacji algorytmu[3]. Chcąc określić wydajność programu równoległego dla zwiększającego się rozmiaru zadania, rozważa się czas realizacji obliczeń jako funkcję pracy systemu komputerowego. W definicji przyspieszenia skalowanego $S_{\mathrm{p}}^{\mathrm{s}}$ zakłada się, że praca systemu jest liniową funkcją liczby procesorów, $W = N_{\mathrm{p}}W_0$. Wtedy czasy wykonania i przyspieszenie obliczeń stają się złożonymi funkcjami liczby procesorów:

$$S_{\mathrm{p}}^{\mathrm{s}}(N_{\mathrm{p}}) = \frac{T_{\mathrm{s}}(N_{\mathrm{p}}W_0)}{T_{\mathrm{p}}(N_{\mathrm{p}}, N_{\mathrm{p}}W_0)} \tag{3.3}$$

Liniowe przyspieszenie skalowane (niekoniecznie o współczynniku 1) uzyskuje się, gdy dla rozmiaru zadania rosnącego proporcjonalnie wraz z liczbą procesorów narzut związany z wykonaniem równoległym także wzrasta proporcjonalnie. Sytuacja taka odpowiada stałemu narzutowi przypadającemu na pojedynczy procesor i stałemu czasowi wykonania równoległego, $T_{\mathrm{p}}(N_{\mathrm{p}}, N_{\mathrm{p}}W_0) = = \mathrm{const}$.

Odpowiadająca skalowanemu przyspieszeniu skalowana efektywność wyrażona jest wzorem:

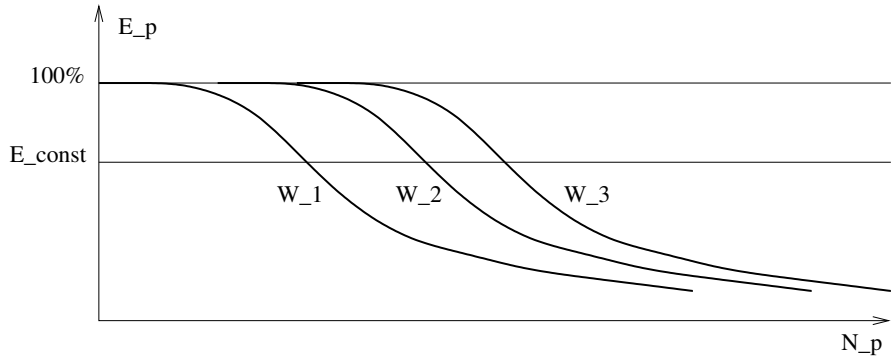
$$E_{\mathrm{p}}^{\mathrm{s}}(N_{\mathrm{p}}) = \frac{T_{\mathrm{s}}(N_{\mathrm{p}}W_0)}{N_{\mathrm{p}}T_{\mathrm{p}}(N_{\mathrm{p}}, N_{\mathrm{p}}W_0)} \cdot 100\% \tag{3.4}$$

Jeśli założy się, w naturalny sposób, że $T_{\mathrm{s}}(N_{\mathrm{p}}W_0) = N_{\mathrm{p}}T_{\mathrm{s}}(W_0)$ (choć często jest to realizowane tylko w przybliżeniu, np. w wyniku zmian w wydajności wykorzystania pamięci podręcznej), to definicja skalowanej efektywności upraszcza się do postaci:

$$E_{\mathrm{p}}^{\mathrm{s}}(N_{\mathrm{p}}) = \frac{T_{\mathrm{s}}(W_0)}{T_{\mathrm{p}}(N_{\mathrm{p}}, N_{\mathrm{p}}W_0)} \cdot 100\% \tag{3.5}$$

Inną miarą opracowaną w celu przełamania negatywnego wydźwięku faktu malejącej do zera efektywności zrównoleglenia E_{p} jest tzw. funkcja izoefektywności (*iso-efficiency function*) [106]. Punktem wyjścia definicji funkcji izoefektywności jest obserwacja typowego zachowania algorytmów dobrze poddających się zrównolegleniu, zobrazowanego na rys. 3.2. Pokazana jest na

[3]Jeżeli za pracę algorytmu przyjmie się liczbę operacji zmiennoprzecinkowych, to dzieląc ją przez czas wykonywania obliczeń (obojętne sekwencyjnych czy równoległych), otrzymuje się klasyczną miarę wydajności obliczeń w milionach operacji zmiennoprzecinkowych na sekundę [Mflop/s]. Miary przyspieszenia i efektywności obliczeń prezentują dla programów równoległych tę klasyczną wydajność zrelatywizowaną do wydajności programów sekwencyjnych. Ich użycie w miejsce klasycznej miary wiąże się z trudnością precyzyjnego obliczenia pracy dla złożonych algorytmów numerycznych.



Rys. 3.2. Typowy przebieg funkcji efektywności zrównoleglenia obliczeń $E_{\mathrm{p}}(N_{\mathrm{p}})$ dla trzech przypadków rozmiaru zadania: $W_1 < W_2 < W_3$

nim zależność efektywności E_{p} od liczby procesorów N_{p} dla trzech rozmiarów problemu: W_1, W_2 i W_3, przy czym $W_1 < W_2 < W_3$. Pozioma kreska $E_{\mathrm{p}}(N_{\mathrm{p}}) = E_{\mathrm{const}}$ wskazuje na możliwość takiego dobierania rozmiaru zadania do liczby procesorów, aby utrzymać stałą efektywność zrównoleglenia obliczeń. Formalnie funkcję izoefektywności definiuje się w sposób uwikłany jako funkcję $W_{\mathrm{izo}}(N_{\mathrm{p}})$, dla której efektywność zrównoleglenia pozostaje stała:

$$E_{\mathrm{p}} = \frac{T_{\mathrm{s}}(W_{\mathrm{izo}}(N_{\mathrm{p}}))}{N_{\mathrm{p}} T_{\mathrm{p}}(N_{\mathrm{p}}, W_{\mathrm{izo}}(N_{\mathrm{p}}))} \cdot 100\% = \mathrm{const} \tag{3.6}$$

Wykazanie istnienia funkcji izoefektywności niskiego rzędu dla danego algorytmu uważane jest za dowód jego skalowalności. Jeśli rząd zbliżony jest do liniowego, algorytm jest dobrze skalowalny. Jednak dla większości algorytmów rozwikłanie wzoru (3.6) napotyka poważne trudności. Dla celów praktycznych istotną wskazówką jest obserwacja, że na mocy wzorów (3.4) i (3.5) algorytmom posiadającym liniową funkcję izoefektywności odpowiada stały czas wykonania przy wzroście rozmiaru zadania proporcjonalnym do liczby procesorów. Dla danego algorytmu posiadanie liniowej funkcji izoefektywności jest równoważne uzyskiwaniu liniowego przyspieszenia skalowanego obliczeń równoległych. W dalszych rozważaniach za kryterium skalowalności programów przyjmowana będzie zdolność do utrzymania stałego lub niewiele wzrastającego czasu obliczeń przy wzroście rozmiaru zadania proporcjonalnym do liczby procesorów.

3.3 Ogólne zasady zrównoleglenia algorytmów MES – lokalność obliczeń i podział obszaru obliczeniowego na nakładające się podobszary

Przedstawiana w niniejszej pracy metodologia zrównoleglenia programów MES stara się realizować scharakteryzowane w punkcie poprzednim cele, polegające na minimalizacji czasu działania i zapewnieniu skalowalności programów poprzez, z jednej strony, dobór algorytmów – optymalnych dla obliczeń sekwencyjnych i poddających się efektywnemu zrównolegleniu, a z drugiej strony, uwzględnienie w procesie zrównoleglenia specyfiki docelowej architektury systemu komputerowego.

Sekwencyjnej optymalności algorytmów poświęcone były rozważania poprzedniego rozdziału. Przedstawione tam algorytmy dobierane i modyfikowane były tak, aby uzyskać optymalną liniową złożoność obliczeniową w zależności od rozmiaru zadania wyrażanego liczbą stopni swobody N (w przypadku algorytmów optymalnych liczba N może być uznawana za odpowiednik pracy systemu komputerowego W).

Nieformalną miarą możliwej do uzyskania efektywności równoległej wersji danego algorytmu jest stopień lokalności jego obliczeń. W dalszych analizach pojęcie lokalności obliczeń używane jest w dwóch znaczeniach. W przypadku realizacji obliczeń dla konkretnego odwzorowania algorytmu na architekturę maszyny wieloprocesorowej, operacje są lokalne, jeśli każdy uczestniczący w ich realizacji procesor odwołuje się tylko do danych zawartych w swojej pamięci lokalnej. Innymi słowy, obliczenia lokalne to obliczenia nie wymagające komunikacji międzyprocesorowej. Aby odróżnić ją od omawianej poniżej abstrakcyjnej lokalności obliczeń, ta lokalność nazywana jest lokalnością realizacji.

W kontekście abstrakcyjnych operacji związanych z aproksymacją równań różniczkowych cząstkowych lokalność ma także inną interpretację, niezależną od konkretnej realizacji komputerowej. Obliczenia są w pełni lokalne, jeśli dotycząc danego obiektu siatki (i związanych z nim stopni swobody) nie odwołują się do innych obiektów siatki (i ich stopni swobody). Takie obliczenia nigdy nie wymagają komunikacji międzyprocesorowej, są jednak rzadkością. Częstszym przypadkiem są obliczenia prawie lokalne, które dotycząc danego obiektu siatki odnoszą się tylko do obiektów będących jego najbliższymi sąsiadami. W przypadku realizacji równoległej powoduje to zazwyczaj odwołanie się do komunikacji międzyprocesorowej. W miarę odnoszenia się do coraz dalszych sąsiadów stopień lokalności obliczeń maleje. Z reguły oznacza to konieczność zwiększenia liczby danych przesyłanych między procesorami.

W przypadku obliczeń prawie lokalnych niejednokrotnie możliwe są takie modyfikacje algorytmów, w których powiela się w lokalnych pamięciach procesorów struktury danych dotyczące pewnych obiektów siatki, tak aby algorytmy w trakcie działania nie musiały odwoływać się do komunikacji międzyprocesorowej. Wtedy realizacja prawie lokalnych operacji abstrakcyjnych przeprowadzana jest lokalnie w sensie działania systemu komputerowego. Powoduje to jednak zwiększone zapotrzebowanie na pamięć w stosunku do wersji sekwencyjnej i niejednokrotnie wprowadzenie dodatkowych obliczeń.

Wybór pomiędzy zarysowanymi powyżej możliwościami wynika często z analizy szybkości wykonania równoległego dla konkretnych architektur systemów komputerowych. Spowolnienie obliczeń równoległych, w stosunku do idealnego modelu przyspieszenia liniowego, wywołane jest przez narzut obliczeń równoległych, na który składają się:

- wymiana komunikatów między procesorami,
- dodatkowe obliczenia nie występujące w algorytmach sekwencyjnych,
- niezrównoważenie obciążenia procesorów, kiedy jeden lub kilka procesorów w maszynie równoległej nie realizuje obliczeń, a ich czas pracy jest czasem jałowym.

Wszystkie wymienione powyżej elementy pojawiają się w bardziej złożonych programach równoległych. Dopasowanie metodologii zrównoleglenia do konkretnej architektury polega często na redukcji czynnika najbardziej spowalniającego obliczenia dla danej architektury.

Przy tworzeniu równoległych wersji algorytmów redukcja całkowitej liczby danych wymienianych między procesorami jest często sprzeczna z redukcją dodatkowych obliczeń wprowadzanych przez wykonanie równoległe. Dążenie do gruboziarnistości obliczeń oznacza wybór w powyższej alternatywie na rzecz redukcji komunikacji międzyprocesorowej kosztem możliwego wprowadzenia dodatkowych obliczeń.

Ze względu na postawienie za cel maksymalizacji ziarna obliczeniowego w ramach przyjętej w niniejszej pracy metodologii zrównoleglenia obliczeń MES, zrównoleglenie to odbywa sie na podstawie dekompozycji obszaru obliczeniowego na *nakładające się* podobszary. Dzięki występowaniu w rozważanych algorytmach wielu obliczeń prawie lokalnych, wprowadzenie strefy nakładania się podobszarów (zwanej dalej w skrócie nakładką) zwiększa liczbę operacji możliwych do wykonania lokalnie, bez odwoływania się do komunikacji międzyprocesorowej.

Obecność nakładki komplikuje proste przyporządkowanie podobszar––procesor–lokalna struktura danych–lokalna pamięć omawiane w p. 3.1. W celu zachowania ułatwień, jakie niesie jego jednoznaczność, wprowadza się następującą strategię uwzględniania istnienia nakładki. Zakłada się podział obszaru obliczeniowego na nienakładające się podobszary i związane z nim, choć nie do końca determinowane przez niego, jednoznaczne przyporządkowanie obiektów siatki procesorom (to ostatnie wprowadza dodatkowo przypisanie procesorom obiektów na brzegu między nienakładającymi się podobszarami). Do sprecyzowania dalszych rozważań służyć będą następujące pojęcia:

Definicja 3.1 Podstawowym podobszarem dekompozycji *nazywany jest zbiór obiektów siatki, które na podstawie podziału obszaru obliczeniowego na nienakładające się podobszary przypisane są pojedynczemu procesorowi.*

Definicja 3.2 Rozszerzonym podobszarem dekompozycji *nazywany jest zbiór obiektów siatki, dla których dane przechowywane są w lokalnej pamięci pojedynczego procesora.*

Definicja 3.3 Lokalną nakładką *nazywany jest, określony dla każdego procesora, zbiór obiektów siatki należących do rozszerzonego podobszaru dekompozycji i nie należących do podstawowego podobszaru dekompozycji.*

Definicja 3.4 Globalną nakładką *nazywana jest suma lokalnych nakładek.*

Jeśli w tekście użyte jest określenie *podobszar dekompozycji*, oznacza to, że odnosi się ono do podstawowych i rozszerzonych podobszarów. Jeśli używane jest określenie *nakładka*, oznacza ono globalną nakładkę.

W przypadku zrównoleglania obliczeń przy wykorzystaniu dekompozycji obszaru, cechy podziału obszaru obliczeniowego na podobszary mają wpływ na wszystkie wymienione uprzednio czynniki spowalniające obliczenia równoległe (patrz także p. 3.5.1). Jedną z najważniejszych cech jest uzyskany stosunek sumy objętości (dla obliczeń dwuwymiarowych (2W) – powierzchni) podobszarów do sumy powierzchni (2W – obwodu) ich brzegów. Dla wielu algorytmów stosunek ten determinuje stosunek czasu przeznaczonego przez procesory na niezależne obliczenia do czasu potrzebnego procesorom na komunikację międzyprocesorową. Algorytmy takie nazywane będą dalej algorytmami wykazującymi proporcję obliczeń do komunikacji [4] (*computation to communication*

[4]Stosunek obliczeń do komunikacji jest często uznawany za miarę ziarnistości obliczeń. Miara ta tylko pośrednio uwzględnia istotny czynnik sposobu organizacji obliczeń – częstości wymiany komunikatów, niejednokrotnie niezależnej od ich całkowitej wielkości.

ratio) równą proporcji objętości do powierzchni (*volume to surface ratio* – określenie to będzie stosowane także w przypadku obliczeń w dwóch wymiarach przestrzennych, gdzie w rzeczywistości oznacza stosunek powierzchni do obwodu).

Dla tak określonych algorytmów można oszacować czas obliczeń i czas komunikacji, otrzymując jednocześnie przybliżoną charakterystykę ziarnistości obliczeń. W przypadku optymalnych równomiernych podziałów obszaru obliczeniowego podobszary będą miały kształt zbliżony do foremnego i objętość rzędu trzeciej (2W – drugiej) potęgi odpowiednio określonego liniowego rozmiaru podobszarów. Powierzchnia podobszarów będzie wielkością rzędu kwadratu (2W – pierwszej potęgi) rozmiaru liniowego. Stosunek objętości do powierzchni, a w konsekwencji obliczeń do komunikacji, będzie dla algorytmów tego typu rzędu rozmiaru liniowego podobszarów. Dla podziałów nieoptymalnych proporcja obliczeń do komunikacji może być wielkością niższego rzędu.

Przy rozwiązywaniu tego samego zadania i zwiększaniu liczby podobszarów (procesorów) rozmiar liniowy podobszarów maleje, co prowadzi do zmniejszania proporcji obliczeń do komunikacji i niekorzystnego zmniejszania ziarnistości obliczeń równoległych. Dążenie do utrzymania gruboziarnistości obliczeń oznacza, że nie dopuszcza się do podziału obszaru obliczeniowego na zbyt małe podobszary (czyli dostosowuję liczbę procesorów realizujących obliczenia do rozmiaru zadania).

W przypadku równoległych aproksymacji MES wykorzystujących dekompozycję obszaru na nakładające się podobszary można, w pewnych przypadkach, oszacować liczbowo proporcję obliczeń do komunikacji, używając dwóch parametrów: liczby stopni swobody N i liczby procesorów N_p. Zakładając podział obszaru na podstawowe podobszary dekompozycji równej wielkości, otrzymuje się liczbę stopni swobody w każdym podobszarze równą N/N_p. Dla dalszych analiz wygodnie jest wyrażać rozmiar tworzonej nakładki przez liczbę związanych z nią stopni swobody[5]. Liczbę tę można wyrazić jako iloczyn powierzchni (2W – obwodu) podobszaru, wyrażonej w jednostkach geometrycznych, i grubości nakładki g_n, wyrażonej liczbą stopni swobody na jednostkę powierzchni (2W – obwodu). Jeśli za liniowy rozmiar podobszaru przyjmie się trzeci (2W – drugi) pierwiastek z liczby stopni swobody w podobszarze, to rozmiar nakładki jest rzędu $g_\mathrm{n} \cdot (N/N_\mathrm{p})^{2/3}$ (2W – $g_\mathrm{n} \cdot (N/N_\mathrm{p})^{1/2}$).

Dalsze analizy (patrz p. 3.4) pokazują, że dla wielu z rozważanych rów-

[5]Zagadnienie powiązania podobszarów składających się z obiektów siatki ze stopniami swobody odpowiadającymi funkcjom bazowym przypisanym obiektom siatki opisane jest w p. 2.5.3.

noległych algorytmów MES narzut związany z komunikacją jest proporcjonalny do wielkości globalnej nakładki, a więc przy stałej grubości nakładki równy $N_\mathrm{p} \cdot O\left((N/N_\mathrm{p})^{2/3}\right)$ (2W – $N_\mathrm{p} \cdot O\left((N/N_\mathrm{p})^{1/2}\right)$), podczas gdy liczba operacji algorytmów wzrasta proporcjonalnie do sumarycznej objętości rozszerzonych podobszarów dekompozycji równej $N + N_\mathrm{p} \cdot O\left((N/N_\mathrm{p})^{2/3}\right)$ (2W – $N + N_\mathrm{p} \cdot O\left((N/N_\mathrm{p})^{1/2}\right)$). Oznacza to proporcję obliczeń do komunikacji równą proporcji objętości do powierzchni w przypadku nakładki o *stałej* grubości. W przypadku gdy grubość nakładki (wyrażana liczbą stopni swobody) rośnie, proporcja obliczeń do komunikacji i ziarnistość obliczeń maleją.

Postawienie za cel zrównoleglenia programu MES minimalizacji jego czasu działania, mierzonej standardowymi: przyspieszeniem i efektywnością zrównoleglenia, oraz jego skalowalności, związanej z rozważaniem problemów o zmiennym rozmiarze, oznacza konieczność analizy dwóch ciągów realizacji obliczeń. W pierwszym rozważa się sekwencję rozwiązań tego samego problemu (stałe N) przy wzrastającej liczbie procesorów. W tym przypadku, dla algorytmów wykazujących proporcję obliczeń do komunikacji równą proporcji objętości do powierzchni, wielkość komunikacji rośnie proporcjonalnie do $N_\mathrm{p}^{1/3}$ (2W – $N_\mathrm{p}^{1/2}$). Niezależnie od wielkości stałych proporcjonalności oznacza to zwiększanie narzutu obliczeń równoległych i zmniejszanie efektywności zrównoleglenia.

Inaczej jest dla drugiego ciągu realizacji obliczeń, gdy dla oceny skalowalności programu rozwiązuje się zadania o rozmiarze rosnącym liniowo wraz z liczbą procesorów. W przypadku algorytmów wykazujących proporcję obliczeń do komunikacji równą proporcji objętości do powierzchni i praca, i wielkość komunikacji rosną proporcjonalnie do N_p. Oznacza to możliwość uzyskania liniowego przyspieszenia skalowanego, a więc i dobrej skalowalności całego programu.

3.4 Algorytmy równoległe dla poszczególnych etapów obliczeń MES

W niniejszym punkcie prezentowane są równoległe wersje algorytmów MES opisywanych w rozdz. 2, uzyskane z zastosowaniem metodologii dekompozycji obszaru obliczeniowego. Przeprowadzana jest dla nich analiza efektywności zrównoleglenia w stosunku do wersji sekwencyjnych. Dla algorytmów równoległych wykorzystujących dekompozycję na nakładające się podobszary, istotną dla efektywności zrównoleglenia jest proporcja obliczeń do komunikacji, związana z grubością nakładki i stopniem lokalności obliczeń. Te dwa ostatnie aspekty są podstawą rozważań przeprowadzanych w następnych punktach.

3.4.1 Dyskretyzacja czasowa i rozwiązywanie układów równań nieliniowych

Dyskretyzacja czasowa i rozwiązywanie układów równań nieliniowych są przykładami zadań, dla których często istnieje alternatywa, czy wybrać algorytmy bardziej wydajne numerycznie, ale trudniejsze do zrównoleglenia i mniej efektywne w wersji równoległej, czy algorytmy mniej optymalne numerycznie, za to łatwo i efektywnie dające się zrównoleglić. Pierwszą grupę algorytmów (w przypadku obliczeń adaptacyjną MES) stanowią metody niejawne, prowadzące do konieczności rozwiązywania układów równań liniowych. Drugą są metody jawne, dla których dyskretyzacja przestrzenna sprowadza się (z punktu widzenia obliczeniowego) do całkowania numerycznego. To ostatnie, jak dalej zostanie omówione, jest operacją immanentnie równoległą, o dużym stopniu lokalności.

W poprzednim rozdziale (pp. 2.2 i 2.3) zwracana była uwaga na lepsze dostosowanie metod niejawnych do rozwiązywania zagadnień wielkiej skali adaptacyjną MES. Ich zadaniem jest zamiana złożonego, zależnego od czasu i/lub nieliniowego problemu na sekwencje prostszych liniowych zagadnień przestrzennych. Z punktu widzenia wydajności obliczeń (maksymalnego wykorzystania zasobów systemów komputerowych) istotna jest liczba generowanych przez te metody problemów liniowych i efektywność dyskretyzacji przestrzennej tych ostatnich.

Rozstrzygnięcie wyboru między metodami jawnymi i niejawnymi dla algorytmów dyskretyzacji czasowej i rozwiązywania równań nieliniowych zależy od własności rozwiązywanego problemu i w związku z tym nie jest szczegółowo dyskutowane, podobnie jak dla przypadku obliczeń sekwencyjnych w rozdz. 2, także dla realizacji równoległej. Nie jest także rozważana ani szczegółowa postać tych algorytmów, ani ich implementacja. Obie są silnie zależne od rozwiązywanego problemu i jednocześnie w dużej mierze niezależne od późniejszej dyskretyzacji przestrzennej. W wielu przypadkach, przy realizacji równoległej algorytmy te stosowane są bez zmian.

3.4.2 Całkowanie numeryczne

Całkowanie numeryczne jest przykładem procedury tzw. konfudująco równoległej (*embarrassingly parallel*). W przypadku sekwencyjnym jest ono realizowane poprzez pętlę po wszystkich podobszarach całkowania. Naturalnym sposobem zrównoleglenia algorytmu jest rozbicie tej globalnej pętli na wiele pętli realizowanych przez poszczególne procesory. Każdy procesor wykonuje

obliczenia dla podobszarów całkowania zawartych w przypisanym mu podstawowym podobszarze dekompozycji. Wysoki stopień lokalności obliczeń umożliwia uzyskanie wysokiej efektywności zrównoleglenia.

Aby przeprowadzić całkowanie numeryczne wyrażeń sformułowania skończenie elementowego (2.30) lub (2.31), konieczna jest znajomość, dla każdego podobszaru całkowania, opisu geometrii elementów i wartości funkcji kształtu w podobszarze (patrz p. 2.6.1). Dla zagadnień zależnych od czasu i/lub nieliniowych konieczna jest także znajomość wartości stopni swobody z poprzedniej chwili i/lub iteracji. Przy braku nakładki może zdarzyć się (np. dla wspomnianego przypadku problemów niestacjonarnych i nieliniowych, a także dla dyskretyzacji nieciągłej i całkowania po bokach elementów), że przeprowadzenie całkowania wymaga przesyłania danych dotyczących obiektów siatki sąsiadujących z danym podobszarem całkowania i/lub związanych z nimi stopni swobody. Dzieje się tak w przypadku całkowania po obszarach obiektów leżących przy brzegu podstawowego podobszaru dekompozycji. Przesyłania tego można uniknąć, jeżeli uwzględni się lokalną nakładkę i zawrze potrzebne dane w strukturze danych nakładki, pod warunkiem jednak, że nie dokonuje się całkowania numerycznego po podobszarach całkowania należących do nakładki.

Ten ostatni problem, selekcji obiektów, po obszarach których dokonuje się całkowania, związany jest ze sposobem tworzenia i przechowywania macierzy skończenie elementowego układu równań liniowych. Sposób ten wynika z cech stosowanej aproksymacji, omówionych poniżej, i wymagań algorytmu rozwiązywania układu równań, które zostaną omówione w następnym punkcie.

Jednoznaczne przypisanie obiektów siatki, a co za tym idzie odpowiadających im stopni swobody, poszczególnym procesorom prowadzi do rozbicia globalnego wektora niewiadomych na podwektory odpowiadające poszczególnym podstawowym podobszarom dekompozycji. Prowadzi to także do rozbicia macierzy układu równań na poziome pasma (zbiory wierszy) odpowiadające podwektorom. Naturalnym sposobem przechowania macierzy układu w lokalnych pamięciach procesorów jest przechowywanie tak określonych poziomych pasm. Każdy niezerowy wyraz takiego pasma odpowiada parze stopni swobody (lub pojedynczemu stopniowi dla wyrazów diagonalnych). Tylko jeden ze stopni swobody każdej pary (odpowiadający funkcji testującej) musi należeć do podwektora odpowiadającego podstawowemu podobszarowi dekompozycji (procesorowi).

Obszar wspólnego niezerowania się funkcji bazowych związanych z pojedynczą parą stopni swobody, a więc obszar całkowania numerycznego prowadzącego do uzyskania wyrazu macierzy odpowiadającego tej parze, może wykra-

czać poza podstawowy podobszar dekompozycji, związany z pasmem macierzy zawierającym ten wyraz. Zdarza się to w przypadku, gdy jeden ze stopni swobody nie należy do podobszaru, ale może zachodzić także wtedy, gdy oba znajdują się na brzegu podobszaru. Dotyczy to wszystkich rozważanych w niniejszej pracy typów aproksymacji (ciągłej liniowej, ciągłej wyższego rzędu, nieciągłej), choć dla każdej na skutek działania innego mechanizmu.

W przypadku braku nakładki, sytuacja powyższa powoduje konieczność przesłania danych do procesora wykonującego całkowanie po określonym podobszarze dekompozycji od procesorów, którym przyporządkowane są sąsiednie podobszary. Jeśli rozważa się eliminację komunikacji przez wprowadzenie nakładki, jej rozmiar musi uwzględniać obiekty siatki będące dodatkowymi podobszarami całkowania oraz kolejne obiekty umożliwiające wykonanie tego dodatkowego całkowania bez komunikacji. Uwzględnienie dodatkowych podobszarów całkowania oznacza, że obliczenia dla nich są powielane na różnych procesorach, co z kolei zmniejsza efektywność zrównoleglenia.

Dalsze zwiększenie rozmiaru nakładki koniecznego do eliminacji komunikacji, może zostać spowodowane przez uwzględnienie schematu przechowania macierzy układu, w którym pewne wiersze są powielane na różnych procesorach. Chcąc zrealizować tworzenie wierszy lokalnie, należy powielić ich obliczanie w identyczny sposób jak dokonywane jest ono na procesorach, do których wiersze te są przypisane na mocy pierwotnego podziału na podstawowe podobszary dekompozycji. Takie schematy przechowywania pojawiają się w algorytmach rozwiązywania układów równań liniowych metodami wielosiatkowymi i metodami z dużym nakładaniem się podobszarów [163].

W dalszych analizach zakłada się, że rozmiar nakładki zawsze jest taki, aby możliwe było dokonanie całkowania numerycznego bez konieczności wymiany danych między procesorami. Z jednej strony oznacza to, że rozmiar ten wynika z wymagań implementacji solwera układów równań liniowych, które określają schemat rozproszonego przechowywania macierzy i w konsekwencji zbiór wierszy macierzy układu przechowywanych lokalnie. Z drugiej strony, ostateczny rozmiar nakładki determinowany jest przez typ aproksymacji, który precyzuje zbiór obiektów siatki (stanowiących podobszary całkowania i wspomagających całkowanie), dla których dane muszą być przechowywane lokalnie dla lokalnego przeprowadzenia całkowania numerycznego.

Z punktu widzenia skalowalności algorytmu całkowania istotnym parametrem pozostaje rozmiar nakładki, określany przez jej grubość. Grubość nakładki może być określana przez liczbę warstw elementów tworzących ją i stopień aproksymacji w tych elementach. Jeśli tak wyrażana grubość pozostaje

stała wraz ze wzrostem rozmiaru zadania, to algorytm całkowania numerycznego wykazuje cechę proporcji obliczeń do komunikacji równej proporcji objętości do powierzchni. Jest więc skalowalny. Jeśli grubość nakładki rośnie wraz ze wzrostem rozmiaru zadania (co często występuje przy adaptacji typu p i może zachodzić w pewnych wariantach metod wielosiatkowych), skalowalność równoległego całkowania numerycznego pogarsza się.

3.4.3 Rozwiązywanie układów równań liniowych

Analizowane w niniejszym punkcie zrównoleglenie solwera równań liniowych polega na zrównolegleniu algorytmu odpowiedniej metody Kryłowa – sprzężonych gradientów (s. 79) lub GMRES (s. 80) – oraz algorytmu poprawy uwarunkowania macierzy układu. W przypadku opisywanych uprzednio optymalnych wielosiatkowych metod poprawy uwarunkowania zrównoleglenie dotyczy algorytmu `MG` (s. 74). Ostatnim algorytmem podlegającym zrównolegleniu jest algorytm wygładzania błędu, w przypadku algorytmów z rozdz. 2 – za pomocą metod Schwarza lub przy zastosowaniu niekompletnego rozkładu ILU(0).

Metody podprzestrzeni Kryłowa

Zrównoleglenie algorytmów metod Kryłowa, przy wyłączeniu procedur poprawy uwarunkowania, sprowadza się do zrównoleglenia podstawowych operacji wektorowych, takich jak np. skalowanie, suma, iloczyn skalarny czy norma, dokonywanych na globalnych wektorach oraz zrównolegleniu wykonania iloczynu dowolnego globalnego wektora z macierzą układu równań (niekiedy ta ostatnia operacja realizowana jest łącznie z algorytmem poprawy uwarunkowania – patrz p. 2.7.5, s. 81). Pozostałe operacje algorytmów, w których nie uczestniczą globalne wektory (w tym np. rozwiązanie problemu minimalizacyjnego w algorytmie GMRES), wykonywane są przez wszystkie procesory lub przez jeden wybrany, który następnie rozgłasza wyniki do wszystkich pozostałych. Czas wykonania powyższych operacji jest jednak, przy założeniu gruboziarnistości obliczeń, pomijalnie mały w stosunku do operacji angażujących wektory globalne.

Każda pozycja w dowolnym globalnym wektorze biorącym udział w operacjach algorytmów metod Kryłowa jednoznacznie odpowiada pozycji określonego stopnia swobody w globalnym wektorze niewiadomych. Realizacja operacji z udziałem wektorów globalnych oparta jest na ich dekompozycji na podwektory – przechowywane w lokalnych pamięciach procesorów – implikowanej przez podziały globalnego wektora stopni swobody. Wykorzystywane są

dwa rodzaje podziałów oparte na klasyfikacji stopni swobody, wynikającej z dekompozycji obszaru obliczeniowego i istnienia nakładki.

Dla każdego podobszaru definiuje się następującą klasyfikację stopni swobody:

Definicja 3.5 Stopniami swobody wewnętrznymi *nazywane są stopnie swobody związane z podstawowym podobszarem dekompozycji.*

Definicja 3.6 Stopniami swobody nakładki *nazywane są stopnie swobody związane z lokalną nakładką.*

Definicja 3.7 Stopniami swobody lokalnymi *nazywane są, łącznie, stopnie swobody wewnętrzne i nakładki. Są one związane z rozszerzonym podobszarem dekompozycji i jednocześnie są to wszystkie stopnie swobody przechowywane w lokalnej pamięci odpowiadającego procesora.*

Definicja 3.8 Stopniami swobody zewnętrznymi *nazywane są stopnie swobody nie będące stopniami lokalnymi.*

Analogicznie do nazw stopni swobody wprowadza się nazwy podwektorów zawierających wartości stopni swobody z odpowiedniej grupy. Dla każdego podobszaru istnieje wewnętrzny podwektor stopni swobody, lokalny podwektor stopni swobody i podwektor stopni swobody nakładki. Wyróżnienie wewnętrznych podwektorów, lokalnych podwektorów i podwektorów nakładki stosuje się także do innych wektorów globalnych.

Pojęcia podwektorów wewnętrznych i lokalnych są podstawą definicji równoległych operacji wektorowych w algorytmach metod Kryłowa. W przypadku operacji redukcji (iloczyn skalarny, norma) każdy procesor realizuje odpowiednie działania na wewnętrznych podwektorach wektorów będących argumentami operacji, po czym następuje runda komunikacji międzyprocesorowej, w celu uzyskania ostatecznego wyniku. Dla pozostałych operacji (inicjowanie, skalowanie, suma) każdy procesor wykonuje operacje na lokalnych podwektorach wektorów będących argumentami operacji. Nie zachodzi potrzeba wymiany komunikatów, jednak część operacji jest powielana na różnych procesorach.

Zrównoleglenie iloczynu wektorów globalnych z macierzą układu równań liniowych zależy od sposobu rozproszonego przechowywania macierzy. Ten z kolei wynika ze sposobu realizacji poprawy uwarunkowania macierzy. Mechanizmy określania wymagań ze strony algorytmów poprawy uwarunkowania omawiane są w następnych punktach. Poniżej rozważone są ich konsekwencje.

Istnieją dwa przypadki wymagań dotyczących przechowywania macierzy układu równań. W obu zakłada się, że każdy procesor przechowuje pewną liczbę wierszy macierzy, tworzących, w sposób rzeczywisty lub umowny, poziome pasmo macierzy. Pierwszy przypadek ma miejsce, jeśli pasma nie zachodzą na siebie, drugi, kiedy występuje zachodzenie pasm.

W przypadku niezachodzenia pasm na siebie procesor przechowuje wiersze odpowiadające wewnętrznym stopniom swobody przyporządkowanego sobie podobszaru dekompozycji. Aby dokonać iloczynu skalarnego pojedynczego takiego wiersza z dowolnym globalnym wektorem, iloczynu stanowiącego elementarny składnik iloczynu macierzy z wektorem, znane muszą być wszystkie wartości wektora, dla których istnieją niezerowe wyrazy w rozważanym wierszu. Wartości te odpowiadają stopniom swobody sąsiadującym ze stopniem, dla którego utworzony jest wiersz macierzy. Aby dokonać iloczynu skalarnego lokalnie, należy wykorzystać taką dekompozycję, dla której lokalnie przechowywane są, oprócz wewnętrznych stopni swobody, także te spośród pozostałych składowych wektora globalnego, które odpowiadają stopniom swobody sąsiadującym z wewnętrznymi stopniami swobody. Z tymi dodatkowymi stopniami swobody związany jest dodatkowy zbiór obiektów siatki, sąsiadujących z podstawowym podobszarem dekompozycji. Ten zbiór dodatkowych obiektów określa wielkość nakładki, której lokalne przechowanie (ściślej: lokalne przechowanie danych związanych z nakładką) umożliwa lokalne przeprowadzenie iloczynu macierz–wektor.

Dla danego pasma macierzy przechowywanego w pamięci lokalnej procesora, zgodnie z mechanizmem tworzenia wyrazów macierzy układu równań liniowych przez całkowanie numeryczne, wielkość nakładki wymagana do lokalnego dokonania całkowania numerycznego jest identyczna z wielkością nakładki wymaganą do lokalnego przeprowadzenia iloczynu macierzy z dowolnym wektorem globalnym.

Po wykonaniu mnożenia macierz układu–globalny wektor prawidłowe wartości znajdują się tylko w *wewnętrznych* podwektorach wynikowego wektora globalnego. Aby uzyskać prawidłowe wartości w *lokalnych* podwektorach, konieczna jest runda komunikacji między procesorami. Każdy procesor otrzymuje od odpowiednich procesorów dane dotyczące podwektorów nakładki i wysyła potrzebne innym procesorom wybrane wartości z podwektorów wewnętrznych. Kolejność wykonywania tych operacji i ich przeplatanie z lokalnymi obliczeniami może mieć istotne znaczenie dla efektywności realizacji równoległej [179].

W drugim przypadku, kiedy pasma macierzy przechowywane lokalnie zachodzą na siebie, stopnie swobody odpowiadające wierszom pasm obejmują

nie tylko stopnie swobody wewnętrzne, ale także pewne stopnie swobody sąsiadujące z nimi. Aby zrealizować lokalnie iloczyny skalarne, składające się na mnożenie macierz–wektor, należy lokalnie przechowywać fragmenty globalnych wektorów odpowiadające przechowywanym lokalnie wierszom macierzy, a także wszystkie pozostałe składowe biorące udział w mnożeniu. Oznacza to utworzenie nakładki, odpowiednio większej niż w przypadku niezachodzenia pasm na siebie, przy czym dodatkowymi stopniami swobody, nie odpowiadającymi lokalnie przechowywanemu pasmu macierzy, są stopnie swobody związane tylko z fragmentem nakładki. Tylko dla odpowiadających im fragmentów globalnych wektorów konieczna jest wymiana między procesorami wartości wynikowych po wykonaniu mnożenia. Dla pozostałych fragmentów podwektorów nakładki obliczenia są powielane na różnych procesorach. To powielanie obliczeń stanowi dodatkowy narzut w stosunku do poprzedniej wersji mnożenia macierzy układu z globalnymi wektorami i musi być kompensowane przez odpowiednie zyski przy całościowej realizacji solwera (np. przyspieszenie zbieżności).

Wielosiatkowa poprawa uwarunkowania macierzy układu

Algorytm `MG` (s. 74) wielosiatkowej poprawy uwarunkowania macierzy układu równań liniowych, dla siatki każdego poziomu, traktowanej jako siatka gęsta, składa się z:

- wygładzenia błędu na siatce gęstej,
- sumowania wektorów, będących wektorami globalnymi dla siatki gęstej,
- mnożenia wektorów globalnych z macierzą $\mathbf{A}_{C_i}$, w celu uzyskania residuum na siatce gęstej,
- zawężenia wektora residuum do przestrzeni zgrubnej, związanej z siatką rzadką,
- rozszerzenia wektora poprawki rozwiązania z przestrzeni zgrubnej.

Wyjątkiem od powyższego schematu jest siatka najrzadsza, dla której jedyną operacją jest dokładne rozwiązanie układu równań liniowych z macierzą $\mathbf{A}_{C_0}$, odpowiadającą problemowi korekty (2.83). Można w tym celu użyć wielu iteracji wygładzania błędu i wtedy realizacja obliczeń przebiega zgodnie z opisem zamieszczonym w następnym punkcie. Można układ rozwiązać odpowiednią metodą Kryłowa, z jednopoziomową poprawą uwarunkowania macierzy algorytmami wygładzania błędu, i wtedy realizacja równoległa odbywa się

według zasad opisanych w punktach poprzednim i następnym. Można wreszcie dokonać rozwiązania metodą bezpośrednią i wtedy najczęściej cały układ rozwiązuje się na pojedynczym procesorze i rozsyła odpowiednie fragmenty wektora wynikowego pozostałym procesorom. Dla celów niniejszych analiz zakłada się, że w każdym z powyższych przypadków koszt obliczeń na siatce najrzadszej jest (podobnie jak to było zakładane dla obliczeń sekwencyjnych w p. 2.6.5) niższego rzędu niż koszt wygładzania błędu na siatce ostatecznej (najgęstszej).

Zrównoleglenie algorytmu `MG` odbywa się, podobnie jak wszystkich innych algorytmów omawianych w niniejszej pracy, na podstawie dekompozycji obszaru. Tym razem jednak dekompozycja dotyczy obliczeń na siatkach różnych poziomów (z możliwym wyłączeniem siatki najrzadszej). Dla każdego rozważanego poziomu można określić inną dekompozycję. Jak pokazują dalsze analizy, różne strategie doboru dekompozycji prowadzą do różnych proporcji obliczeń do komunikacji. Ze względu na ukierunkowanie analiz w niniejszej pracy na architektury systemów komputerowych o relatywnie wolnej sieci komunikacyjnej, szczegółowej analizie (i późniejszej realizacji) poddany jest przypadek minimalizacji liczby danych przesyłanych między procesorami. Przypadek ten odpowiada wykorzystaniu dla siatek wszystkich poziomów tej samej pierwotnej dekompozycji obszaru na nienakładające się podobszary. W przypadku niejednorodnie zagęszczanych siatek adaptacyjnych strategia ta może prowadzić do negatywnych konsekwencji, w postaci niezrównoważenia obciążenia procesorów przy obliczeniach na siatkach niższych poziomów [36].

Macierze $\mathbf{A}_{C_i}$, odpowiadające problemowi korekty (2.83) i określone dla siatek kolejnych poziomów, mogą być otrzymywane poprzez projekcje Galerkina (2.86) lub poprzez całkowanie sformułowania słabego (2.83) na siatce odpowiedniego poziomu. W algorytmach równoległych macierze przechowywane są w sposób rozproszony. Jeśli wykorzystuje się założenie o identyczności pierwotnej dekompozycji obszaru na podstawowe podobszary dekompozycji na wszystkich poziomach, to dekompozycja ta determinuje wygodny podział macierzy $\mathbf{A}_{C_i}$ na niezachodzące na siebie poziome pasma, przechowywane w pamięciach poszczególnych procesorów. W takim przypadku poszczególne macierze można łatwo uzyskać poprzez przeprowadzane w pełni lokalnie projekcje Galerkina. Ze względu na rzadką strukturę macierzy $\mathbf{A}_{C_i}$ złożoność tych projekcji jest rzędu liczby stopni swobody na danej siatce (zależy także silnie od stopnia aproksymacji – patrz pp. 2.7.8 i 2.7.9).

Wyrazy macierzy $\mathbf{A}_{C_i}$ można alternatywnie obliczyć przez całkowanie wyrażeń ze sformułowania skończenie elementowego dotyczącego problemu ory-

ginalnego, stosując zmianę podobszarów całkowania na związane z aktualnie rozważaną siatką i używając funkcji kształtu zdefiniowanych dla tej siatki (por. wzór (2.85)). Obliczenia na siatkach różnych poziomów można traktować zupełnie niezależnie od siebie. W związku z tym do przypadków siatek niższych poziomów odnoszą się wszelkie analizy przeprowadzane dla całkowania numerycznego na siatce najgęstszej w p. 3.4.2. Nakładka wynikająca z wymagań całkowania numerycznego ponownie definiowana jest tak, aby zagwarantować obliczenie wszystkich wyrazów lokalnie przechowywanych fragmentów macierzy $\mathbf{A}_{C_i}$ bez odwoływania się do komunikacji międzyprocesorowej.

Sposób przechowywania macierzy $\mathbf{A}_{C_i}$ związanych z siatkami kolejnych poziomów, a więc i wielkość lokalnie przechowywanych poziomych pasm $\mathbf{A}_{C_i}$, zależy od wymagań algorytmu wygładzania błędu. Znając sposób przechowania macierzy $\mathbf{A}_{C_i}$, operacje ich iloczynu z dowolnymi wektorami globalnymi przeprowadza się identycznie jak w przypadku algorytmów metod Kryłowa, wykorzystując odpowiednie podziały wektorów globalnych. Również w identyczny sposób realizuje się standardowe operacje wektorowe na wektorach globalnych. Ponownie wielkość nakładki wymagana do lokalnego przeprowadzenia iloczynu macierzy $\mathbf{A}_{C_i}$ z globalnymi wektorami jest taka sama jak wielkość nakładki potrzebna do lokalnego obliczenia, poprzez całkowanie numeryczne, wszystkich wyrazów lokalnie przechowywanego pasma $\mathbf{A}_{C_i}$.

Specyficznymi operacjami algorytmów metod wielosiatkowych są zawężenie do przestrzeni zgrubnej i rozszerzenie z przestrzeni zgrubnej, oba przeprowadzane dla wektorów globalnych danej siatki, traktowanej jako siatka gęsta. W celu dokonania analiz teoretycznych operacje te zapisywane są w postaci zawierającej operatory macierzowe $\mathbf{R}_{C_i}$ i $\mathbf{R}_{C_i}^T$. W praktyce realizacja polega na sekwencji operacji lokalnych. Każda pozycja w wektorze należącym do wektorowej przestrzeni zgrubnej związana jest, poprzez odpowiadający stopień swobody, z pewną funkcją bazową na siatce rzadkiej. Każda taka funkcja daje się przedstawić jako kombinacja liniowa funkcji bazowych na siatce gęstej. Operacje zawężenia i rozszerzenia (restrykcji i prolongacji) polegają na wykorzystaniu współczynników takich kombinacji liniowych (patrz p. 2.7.4).

Powyższe współczynniki można otrzymać albo poprzez interpolację, albo odpowiednią projekcję funkcji bazowych. Aby znaleźć w sposób lokalny współczynniki dla funkcji bazowych związanych z danym podobszarem, konieczne jest lokalne przechowywanie danych o obszarze, w którym funkcje te są różne od zera. Prowadzi to do stworzenia nakładki o takiej samej szerokości jak wymagana w przypadku całkowania numerycznego i iloczynu macierzy układu dla siatki na danym poziomie. Problem polega na tym, że dla danej pary siatka

gęsta–siatka rzadka rozmiar nakładki wyznaczany jest przez siatkę rzadką i jest z reguły wielokrotnością rozmiaru wymaganego dla siatki gęstej.

Jeśliby chcieć, w takim wypadku, dostosowywać nakładkę dla siatki gęstej do nakładki dla siatki rzadkiej, poczynając od siatki najrzadszej, prowadziłoby to do wzrostu rozmiaru nakładki na kolejnych poziomach w postępie geometrycznym. W przypadku zwiększania rozmiaru problemu poprzez adaptację typu h, rozwiązanie takie prowadzi do utraty skalowalności algorytmu poprzez eksponencjalny wzrost rozmiaru wymienianych komunikatów przy wygładzaniu na siatce najgęstszej. Możliwym wyjściem z tej sytuacji jest ograniczenie liczby poziomów siatki w algorytmie wielosiatkowym, co jednak wymaga dokładnego rozwiązania większego problemu na siatce najrzadszej. W takim wypadku daje się utrzymać skalowalność algorytmu, ale liczba wymienianych danych pozostaje nadal duża, co może powodować nieoptymalność algorytmu i prowadzić do przepełnienia sieci komunikacyjnej. Bardziej wyrafinowane strategie dekompozycji i równoważenia obciążenia są wymagane w celu uzyskania optymalności obliczeń [36].

Wygładzanie błędu blokową metodą Gaussa-Seidla

Algorytm wygładzania błędu blokową metodą Gaussa-Seidla składa się z globalnej pętli po blokach poprawy uwarunkowania, w trakcie której rozwiązuje się problemy lokalne (2.56). Dla tego algorytmu naturalną, równoległą realizacją jest rozbicie pętli po wszystkich blokach poprawy uwarunkowania na szereg pętli wykonywanych równolegle, z których każda przebiega po blokach poprawy uwarunkowania przydzielonych pojedynczemu podobszarowi dekompozycji (procesorowi) [30, 15]. Po zakończeniu pętli lokalnych odbywa się wymiana komunikatów między procesorami celem uaktualnienia odpowiednich wartości w lokalnych podwektorach.

Przy realizacji algorytmu wygładzania błędu blokową metodą Gaussa-Seidla (patrz p. 2.7.8) wykorzystuje się elementarne bloki macierzy układu $\mathbf{A}_{C_i}$ oraz specjalnie utworzone diagonalne bloki poprawy uwarunkowania. Bloki poprawy uwarunkowania przed wykonywaniem pętli wygładzania błedu są poddawane odwróceniu lub rozkładowi LU, w celu przyspieszenia wykonania późniejszych wielokrotnych iteracji algorytmu Gaussa-Seidla. Przy realizacji równoległej te wstępne operacje na blokach poprawy uwarunkowania są przeprowadzone lokalnie przez poszczególne procesory.

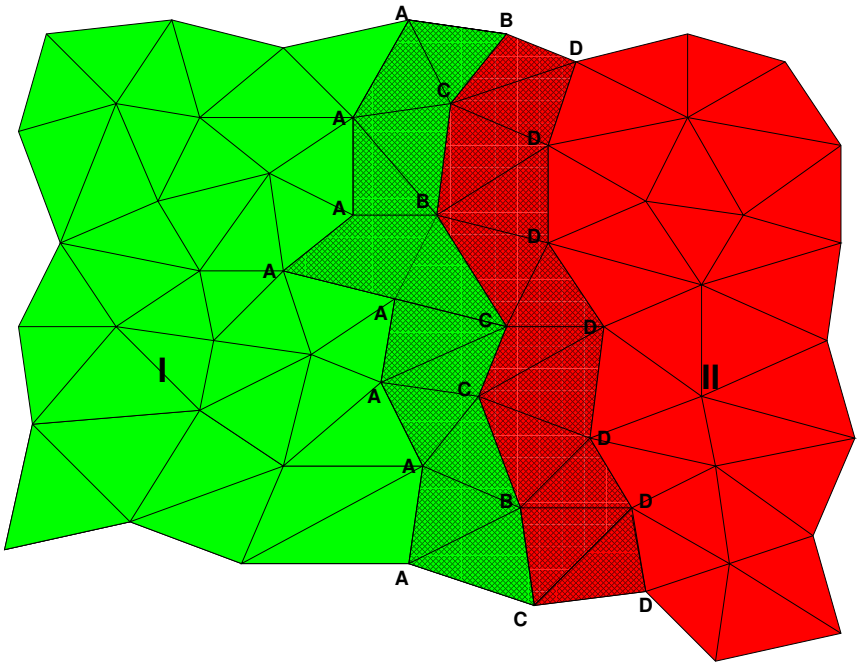
W celu efektywnej realizacji algorytmu blokowej metody Gaussa-Seidla wszystkie bloki macierzy występujące w lokalnych problemach rozwiązywanych na danym procesorze są przechowywane w jego pamięci lokalnej. Dotyczy to

bloków poprawy uwarunkowania, a także wszystkich, powiązanych z nimi poprzez zagadnienia lokalne, bloków elementarnych. Wyznacza to schemat przechowywania macierzy $\mathbf{A}_{C_i}$ w postaci poziomych pasm (zbiorów wierszy). Ten schemat determinuje także rozmiar nakładki.

Podwektory poprawy uwarunkowania są najczęściej, w celu zagwarantowania zbieżności iteracji Gaussa-Seidla, powiązane z małymi podobszarami obszaru obliczeniowego – łatami elementów (patrz pp. 2.5.5 i 2.7.9). Nawet jeśli podwektory poprawy uwarunkowania nie zachodzą na siebie, powiązane z nimi łaty mogą nakładać się na siebie. Powoduje to nakładanie się na siebie rozszerzonych podobszarów dekompozycji. Zachodzenie na siebie podwektorów poprawy uwarunkowania związane jest ze zwiększonym obszarem nakładania się łat elementów. Celem powiększania łat elementów jest zwiększenie efektywności wygładzania błędu [45]. Prowadzi to jednak, w przypadku realizacji równoległej, z jednej strony – do zachodzenia na siebie poziomych pasm macierzy przechowywanych w pamięciach lokalnych procesorów, a z drugiej strony – do zwiększonego nakładania się na siebie łat elementów, i w konsekwencji do zwiększania rozmiaru nakładki. Tak określona nakładka jest jeszcze dalej rozszerzana ze względu na wymagania dotyczące lokalnej realizacji tworzenia wyrazów macierzy i realizacji ilczynów macierzy z wektorami. Może to znacznie zwiększać koszty pamięciowe algorytmu w porównaniu z wersją sekwencyjną.

Poza rozbiciem pojedynczej pętli na pętle związane z poszczególnymi podobszarami, wersje równoległa i sekwencyjna algorytmu wygładzania błędu blokową metodą Gaussa-Seidla pozostają identyczne. W wyniku tego liczba operacji arytmetycznych w różnych wersjach jest taka sama i niezależna od liczby procesorów oraz rozmiaru nakładki. Inaczej dzieje się z wielkością komunikacji miedzyprocesorowej. Liczba danych przesyłanych między procesorami zależy od liczby stopni swobody w nakładce i mechanizmu wymiany wartości w wektorach globalnych w trakcie iteracji. Mechanizm ten opisany jest poniżej, w przypadku wymiany wartości stopni swobody przy rozwiązywaniu szczególnego równania liniowego, otrzymanego przez zastosowanie klasycznej liniowej dyskretyzacji MES. W takiej sytuacji stopnie swobody przypisane są tylko wierzchołkom elementów. Można ten przypadek łatwo uogólnić na wszelkie inne typy aproksymacji rozważane w niniejszej pracy.

Rysunek 3.3 przedstawia fragment siatki MES z podziałem na dwa podobszary – I i II – oraz nakładką o szerokości dwóch elementów (powstałą przez poszerzenie o warstwę grubości jednego elementu podstawowych podobszarów dekompozycji). Kolory odpowiadają obiektom wewnętrznym dla danego podobszaru, a obszar zacieniowany to nakładka. Obiekty (krawędzie i wierzchołki)



Rys. 3.3. Ilustracja klasyfikacji stopni swobody przeprowadzanej w celu równoległego rozwiązania układu równań liniowych (opis w tekście)

na brzegu między podstawowymi podobszarami dekompozycji mogą być przyporządkowane arbitralnie. W sytuacji przedstawionej na rysunku wierzchołki oznaczone literą B są wewnętrzne dla podobszaru I, a oznaczone literą C dla podobszaru II.

Zakłada się, że podwektory poprawy uwarunkowania są tożsame z elementarnymi podwektorami stopni swobody i odpowiadają stopniom swobody w pojedynczym wierzchołku elementu. Stopnie swobody w sąsiedztwie nakładki, z racji różnic w traktowaniu podczas iteracji solwera, dzielą się na cztery grupy:

- A – stopnie swobody wewnętrzne dla podobszaru I i leżące na zewnętrznym brzegu lokalnej nakładki dla podobszaru II; w trakcie iteracji są uaktualniane tylko w podobszarze I,
- B – stopnie swobody wewnętrzne dla podobszaru I i leżące na wewnętrznym brzegu lokalnej nakładki dla podobszaru II; w trakcie iteracji są uaktualniane w obu podobszarach,

- C – stopnie swobody leżące na wewnętrznym brzegu lokalnej nakładki dla podobszaru I i wewnętrzne dla podobszaru II; w trakcie iteracji są uaktualniane w obu podobszarach,

- D – stopnie swobody leżące na zewnętrznym brzegu lokalnej nakładki dla podobszaru I i wewnętrzne dla podobszaru II; w trakcie iteracji są uaktualniane tylko w podobszarze II.

Algorytm wymiany wartości stopni swobody, po każdej lokalnej pętli po podwektorach poprawy uwarunkowania, jest dla kolejnych podobszarów następujący. Podobszar I: wysyła do podobszaru II uaktualnione wartości stopni swobody A i C, otrzymuje uaktualnione wartości stopni swobody B i D, zgodnie z algorytmem nanosi poprawki w stopniach swobody B na wartości otrzymane lokalnie, wysyła poprawione wartości stopni swobody B, otrzymuje poprawione wartości stopni swobody C. Podobszar II: wysyła do podobszaru I uaktualnione wartości stopni swobody B i D, otrzymuje uaktualnione wartości stopni swobody A i C, zgodnie z algorytmem nanosi poprawki w stopniach swobody C na wartości otrzymane lokalnie, wysyła poprawione wartości stopni swobody C, otrzymuje poprawione wartości stopni swobody B. Wymiana znacznie się upraszcza, jeśli nakładka ma szerokość tylko jednego elementu i istnieją tylko stopnie swobody grupy A i B. Istnieją także warianty metod dekompozycji [49], w których pomija się niektóre z poprawek dokonywanych lokalnie. Celem tych pominięć jest uproszczenie wymiany wartości stopni swobody po każdej iteracji, co w wielu wypadkach nie prowadzi do znacznego spowolnienia zbieżności solwera.

Ze względu na dokonywanie wymiany wartości między procesorami po zakończeniu pętli po lokalnych podwektorach, opisywany równoległy algorytm blokowej metody Gaussa-Seidla staje się kombinacją blokowych metod Jacobiego i Gaussa-Seidla – cześć danych używanych w problemach lokalnych jest uaktualniana na bieżąco w trakcie obliczeń, część danych (dla podwektorów związanych z grupami obiektów siatki na brzegach podobszarów dekompozycji) pochodzi z początku iteracji. Zmienia się w porównaniu z algorytmem sekwencyjnym kolejność rozwiązywania problemów lokalnych. Powoduje to, trudne do ilościowego ujęcia, spowolnienie zbieżności metody[6]. Niemniej zwiększanie

[6]W teoretycznych analizach przedstawioną w niniejszym punkcie równoległą wersję wygładzania błędu blokową metodą Gaussa-Seidla traktuje się jako addytywną metodę Schwarza z podobszarami odpowiadającymi dekompozycji obszaru. W tym ujęciu lokalne problemy odpowiadają całym podobszarom dekompozycji i rozwiązywane są w sposób przybliżony, poprzez jedną iterację blokowej metody Gaussa-Seidla.

liczby procesorów (podobszarów) i zwiększanie rozmiaru nakładki na pewno wpływa ujemnie na zbieżność[7] [53].

Wygładzanie błędu przy zastosowaniu blokowego niekompletnego rozkładu ILU(0)

Algorytm wykorzystywany w przypadku poprawy uwarunkowania przy zastosowaniu blokowego niekompletnego rozkładu ILU(0), podobnie jak algorytm wygładzania błędu blokową metodą Gaussa-Seidla, realizowany jest w dwóch fazach. W pierwszej następuje przygotowanie bloków poprawy uwarunkowania algorytmem `ILU(0)` z s. 83, w drugiej właściwe wygładzanie błędu algorytmem `FR-BS` z s. 84.

Algorytm `ILU(0)`, podobnie jak wszelkie wersje eliminacji Gaussa, jest trudny do efektywnego zrównoleglenia. Udaje się ono tylko dla szczególnych konfiguracji obszarów obliczeniowych [177]. W przypadku ogólnym można z algorytmami `ILU(0)` i `FR-BS` postąpić podobnie jak z algorytmem wygładzania błędu blokową metodą Gaussa-Seidla. Każdą występującą w algorytmach globalną pętlę po blokach poprawy uwarunkowania rozbija się na odpowiednią liczbę (równą liczbie procesorów) pętli po lokalnych dla każdego procesora blokach poprawy uwarunkowania. Po zakończeniu pętli lokalnych następuje wymiana danych między procesorami.

Przy takim uproszczonym podejściu do zrównoleglenia wygładzania błędu z zastosowaniem blokowego niekompletnego rozkładu ILU(0), algorytm traci, z teoretycznego punktu widzenia, własności charakterystyczne dla wersji sekwencyjnej. Znika ścisłe powiązanie obliczeń dla wszystkich sąsiadujących z sobą bloków poprawy uwarunkowania. Obliczenia dla bloków należących do różnych podobszarów dekompozycji stają się luźno powiązane, wyłącznie poprzez wymianę danych po zakończeniu pętli wygładzania błędu. W przypadku analiz zbieżności właściwym ujęciem otrzymanej metody jest traktowanie jej jako addytywnej metody Schwarza z podobszarami odpowiadającymi rozszerzonym podobszarom dekompozycji i niedokładnym rozwiązaniem problemów lokalnych, przy zastosowaniu niekompletnego rozkładu ILU(0).

Realizacja uproszczonego zrównoleglenia wygładzania błędu przy zastosowaniu rozkładu ILU(0) wymaga niewielkich zmian w algorytmach sekwencyjnych. W celu efektywnego wykonania wersji równoległej należy tylko zagwarantować, aby wszystkie operacje wykonywane przez procesory po rozbiciu pętli

[7] Chodzi tu nie o bezwzględną zbieżność metody, ale o zbieżność wersji równoległej w porównaniu ze zbieżnością wersji sekwencyjnej.

globalnych na zbiory pętli lokalnych mogły być realizowane lokalnie. Oznacza to, że sposób przechowywania bloków macierzy (elementarnych i poprawy uwarunkowania) oraz mechanizm wymiany wartości w wektorach po wykonaniu iteracji wygładzania błędu jest podobny jak w równoległym algorytmie blokowej metody Gaussa-Seidla. Oznacza to także, trudne do analitycznego ujęcia, spowolnienie zbieżności solwera. Czy spowolnienie to jest skutecznie rekompensowane przez wysoką efektywność rozważanego uproszczonego zrównoleglenia, zależy od rozwiązywanego problemu i cech aproksymacji.

Z punktu widzenia złożoności obliczeniowej istotny jest fakt, że przy użyciu do wygładzania błędu niekompletnej eliminacji Gaussa, wielkość podwektorów poprawy uwarunkowania nie odgrywa tak istotnej roli dla zbieżności solwera równań liniowych jak w przypadku iteracji prostych Gaussa-Seidla. Wykorzystuje się więc najczęściej nienakładające się podwektory poprawy uwarunkowania, np. identyczne z podwektorami elementarnymi. Oznacza to, że przechowywane lokalnie pasma macierzy $\mathbf{A}_{C_i}$ nie zachodzą na siebie, co implikuje optymalną złożoność całkowania numerycznego i przeprowadzania iloczynów macierz–wektor. Nakładka, mimo minimalnego rozmiaru, wynikającego tylko z założenia lokalnego wykonywania powyższych operacji, ma wielkość zależną od typu i stopnia aproksymacji. Przy zdeterminowanym przez powyższe wymagania rozmiarze nakładki, zależność złożoności komunikacyjnej od rozmiaru nakładki pozostaje dla wygładzania z zastosowaniem niekompletnej eliminacji Gaussa taka sama jak dla iteracji prostych Gaussa-Seidla.

Podsumowanie – złożoność obliczeniowa równoległych wersji solwerów układów równań liniowych

Złożoność obliczeniowa zaprezentowanych równoległych wersji omawianych w niniejszej pracy iteracyjnych solwerów liniowych jest tego samego rzędu co złożoność wersji sekwencyjnych. Złożoność komunikacyjna jest liniową funkcją rozmiaru nakładki (mierzonego liczbą związanych z nią stopni swobody). Ten z kolei wynika z grubości nakładki, określanej przez wymagania algorytmów, i sumy powierzchni podobszarów dekompozycji. Ta ostatnia związana jest z cechami dekompozycji obszaru i jej minimalizacja jest jednym z celów algorytmów podziału siatki omawianych w p. 3.5.1. Niemniej jednak, przy wzrastającej liczbie podobszarów (procesorów) dla tego samego zadania suma powierzchni podobszarów zawsze rośnie.

Przy takich charakterystykach wykonania równoległego nie jest możliwe uzyskanie liniowego przyspieszenia wersji równoległej, można jednak dążyć do zapewnienia skalowalności algorytmu, a nawet uzyskania liniowego przyspie-

szenia skalowanego. Zależeć to będzie nie tylko od czasowej i komunikacyjnej złożoności pojedynczej iteracji algorytmu, ale także od zbieżności solwera, zależnej od problemu i typu aproksymacji [53].

3.4.4 Adaptacja

Adaptacja siatki przebiega w dwóch etapach. W pierwszym szacowany jest aktualny błąd aproksymacji i obliczane są elementowe wskaźniki adaptacji. W drugim, zależnie od wartości wskaźników i strategii adaptacji, elementy są dzielone, złączane, zmieniany jest lokalny stopień adaptacji, przesuwane są wierzchołki elementów itp.

Jak już wspomniano w p. 2.8, algorytmy pierwszego etapu adaptacji, a więc w szegółności ich wersje równoległe, nie są analizowane w niniejszej pracy. Analizowane są natomiast algorytmy drugiego etapu, a także wszelkie konsekwencje, jakie dla realizacji pozostałych faz obliczeń MES wynikają z zastosowania różnych typów adaptacji.

Równoległe wersje algorytmów drugiego etapu adaptacji analizowane są, zgodnie z przyjętą w pracy metodologią, pod kątem lokalności obliczeń i wynikających z niej charakterystyk komunikacji. Spośród tych algorytmów tylko podziały i złączanie elementów nie mają charakteru w pełni lokalnego i muszą być modyfikowane dla wykonania równoległego.

Modyfikacje w fazie określania elementów do podziału/złączania konieczne są dla siatek o ograniczonej nieregularności (w szczególności dla siatek regularnych). Nawet jeśli obliczanie wskaźników adaptacji jest dokonywane w pełni lokalnie, to niektóre z elementów przeznaczane są do podziału (lub niedopuszczane do złączenia) niezależnie od wartości ich wskaźnika, dla utrzymania założonej ograniczonej nieregularności siatki. Jeśli elementy wskazywane do podziału znajdują się na brzegu rozszerzonego podobszaru dekompozycji, do ustalenia ostatecznej listy dzielonych/złączanych elementów konieczna jest jedna runda lub nawet kilka rund wymiany danych między procesorami.

Założeniem równoległej realizacji procesu modyfikacji siatki jest przeprowadzanie podziałów i złączeń lokalnie dla wszystkich wskazanych lokalnych obiektów siatki. Trudności pojawiają się w przypadku podziału obiektów bezpośrednio na granicach rozszerzonych podobszarów dekompozycji. Dla siatek nieregularnych podziałowi elementu po jednej stronie granicy niekoniecznie towarzyszy podział elementu po drugiej stronie. Na brzegu pomiędzy tymi elementami pojawiają się obiekty występujące tylko po stronie elementu dzielonego. W celu zapewnienia spójności globalnej struktury danych konieczna jest najczęściej (zależnie od konkretnej implementacji) wymiana komunikatów

między procesorami.

Złączanie elementów może zostać przeprowadzone lokalnie tylko wtedy, jeśli wszystkie obiekty tworzące złączane elementy są dostępne lokalnie. Oznacza to, że albo obiekty te muszą przed złączaniem zostać przesłane do procesorów dokonujących złączenia, albo podziały obszaru na podobszary będą przyporządkowywały pojedynczym procesorom całe rodziny elementów. To ostatnie rozwiązanie jest korzystne dla wielosiatkowych solwerów równań liniowych, dla których upraszcza procesy rzutowania rozwiązania pomiędzy siatkami.

3.5 Algorytmy wspomagające równoległą realizację obliczeń MES

3.5.1 Podział na podobszary

Najważniejszymi algorytmami wspomagającymi równoległą realizację obliczeń MES są algorytmy podziału obszaru obliczeniowego na podobszary. Od cech podziału zależą, z jednej strony, wydajność numeryczna metod, a z drugiej, wydajność obliczeniowa. Wydajność numeryczna, w kontekście omawianych metod, sprowadza się głównie do zbieżności iteracyjnych solwerów układów równań liniowych, choć ma także wpływ, dla niektórych strategii rozwiązywania, na zbieżność solwerów równań nieliniowych, czy zbieżność rozwiązań chwilowych do stanu ustalonego. Przez wydajność obliczeniową rozumiana jest efektywność zrównoleglenia pojedynczej iteracji omawianych algorytmów. W przypadku omawianych metod niejawnych oznacza to ponownie koncentrację na procedurach tworzenia i rozwiązywania układów równań liniowych.

Podział obszaru obliczeniowego na podobszary ma dwa główne cele: pierwszym jest zapewnienie odpowiedniego rozmiaru podobszarów, a drugim uzyskanie właściwego kształtu podobszarów.

Realizacja pierwszego z celów ma doprowadzić do zrównoważenia obciążenia procesorów – minimalizacji czasu jałowego ich pracy. Zadanie zrównoważenia obciążenia procesorów może polegać na prostym zagwarantowaniu równej wielkości podobszarów dla dedykowanych systemów o jednakowych procesorach, może też być znacznie bardziej złożone, jeśli chce się uwzględnić procesory o różnych mocach obliczeniowych [31] lub nawet zmienne obciążenie procesorów w trakcie wykonania zadania. W przypadku metod adaptacyjnych p i hp istotne znaczenie ma także zróżnicowanie wymaganych nakładów obliczeniowych w różnych fragmentach obszaru obliczeniowego. Zróżnicowanie to można uwzględnić przyporządkowując wagi poszczególnym obiektom siatki i żądając

właściwego rozkładu sum wag obiektów przyporządkowanych poszczególnym procesorom.

Miarą stopnia realizacji drugiego z celów podziału obszaru jest uzyskany stosunek powierzchni (obwodu) brzegu podobszarów do objętości (powierzchni) ich wnętrzna, który (wraz z grubością nakładki) determinuje ziarnistość obliczeń i wielkość komunikacji międzyprocesorowej. Odpowiedni kształt podobszarów ma także wpływ na zbieżność iteracyjnych solwerów równań liniowych.

W przypadku algorytmów MES dekompozycja obszaru obliczeniowego przeprowadzana jest jako podział siatki MES. Z punktu widzenia złożoności obliczeniowej istotne są algorytmy dokonujące podziału na podstawowe (nienakładające się) podobszary. Podobszary nakładające się są najczęściej generowane dwustopniowo, najpierw generuje się podobszary nienakładające się, a następnie tworzy nakładkę o określonej grubości. To ostatnie zadanie ma najczęściej charakter techniczny, zależny dodatkowo od struktur danych używanych do przechowywania siatki MES, i dokonywane jest w czasie $O(N)$.

W przypadku rozważania podziału całego ciągu siatek przy realizacji równoległej metod wielosiatkowych konieczne jest uwzględnienie minimalizacji komunikacji dla fazy rzutowania wektorów pomiędzy przestrzeniami związanymi z różnymi siatkami. Można to zrobić przez narzucenie dodatkowego ograniczenia, z góry przypisując pewne elementy siatki danego poziomu określonym podobszarom uzyskanym przez podział siatki poziomu wyższego [36]. W skrajnym przypadku podział siatki na niższych poziomach jest w całości determinowany przez podział siatki najgęstszej, a w konsekwencji podstawowe podobszary dekompozycji dla obliczeń na różnych poziomach pokrywają się.

Zadanie podziału siatki na nienakładające się podobszary polega na jednoznacznym przyporządkowaniu obiektów siatki poszczególnym procesorom. Istnieją dwa podstawowe podejścia do rozwiązania tego zadania. W pierwszym, zwanym dalej bezpośrednim, wykorzystuje się strukturę danych siatki MES oraz cechy topologiczne (połączenia elementów) i geometryczne (położenie wierzchołków) siatki [74]. W drugim, zwanym dalej grafowym, dokonuje się reprezentacji siatki MES w postaci grafu i rozwiązuje zadanie podziału grafu na podgrafy [157].

W podejściu grafowym istotnym etapem jest reprezentacja siatki MES. W klasycznej MES, dla której stopnie swobody związane są wyłącznie z wierzchołkami elementów, standardowym podejściem jest przypisanie wierzchołkom grafu wierzchołków elementów, a krawędziom grafu elementów lub krawędzi. Obliczenia w algorytmach związane są wtedy z wierzchołkami grafu, a krawędzie grafu oznaczają uzależnienie pomiędzy obliczeniami dla różnych wierzchoł-

ków. Wierzchołkom i krawędziom grafu przypisuje się wagi odzwierciedlające: w przypadku wierzchołków grafu zróżnicowanie liczby operacji związanych z poszczególnymi wierzchołkami siatki, a w przypadku krawędzi grafu zróżnicowanie stopnia powiązania między wierzchołkami tworzącymi parę przyporządkowaną pojedynczej krawędzi. Zbiór krawędzi grafu, które po jego podziale łączą wierzchołki należące do różnych podgrafów (podobszarów) nazywa się separatorem krawędziowym (*edge separator*). Zadanie podziału grafu definiuje się jako problem optymalizacji z więzami, gdzie celem jest minimalizacja sumy wag związanych z separatorem krawędziowym, a ograniczeniem jest równy podział sum wag związanych z wierzchołkami podgrafów. To ostatnie wymaganie łagodzi się przez dopuszczenie drobnego (rzędu kilku procent) niezrównoważenia obciążenia.

Minimalizacja wag związanych z separatorem krawędziowym służy jako przybliżenie minimalizacji komunikacji międzyprocesorowej. Przybliżenie to prawidłowo odzwierciedla stan faktyczny dla rozważanej powyżej klasycznej liniowej aproksymacji MES. Jednak przy bardziej wyrafinowanych aproksymacjach konieczna jest modyfikacja reprezentacji. Na przykład, dla aproksymacji nieciągłej obliczenia związane są z elementami, a dla rozmiaru komunikacji międzyprocesorowej istotne znaczenie ma liczba boków elementów na brzegu między podobszarami. Rozwiązaniem tego problemu może być zastosowanie innej reprezentacji siatki, dla której wierzchołkami grafu są elementy, a jego krawędziami boki elementów. W przypadku aproksymacji wyższego stopnia i adaptacji typu p lub hp konieczne staje się uwzględnienie lokalnego stopnia aproksymacji. Można to osiągnąć poprzez połączenie odpowiedniej grafowej reprezentacji siatki ze złożonym przypisywaniem wag wierzchołkom i krawędziom grafu.

Problem dekompozycji grafu, zdefiniowany jako problem optymalizacji, jest zagadnieniem NP-zupełnym. Istnieje wiele heurystycznych metod aproksymacji tego problemu, różniących się między sobą jakością uzyskanych podziałów i złożonością obliczeniową. Najpopularniejsze obecnie są metody wielopoziomowej rekurencyjnej bisekcji (*multilevel recursive bisection*), produkujące wysokiej jakości podziały i działające w czasie liniowym względem rozmiaru grafu. Przegląd tych i innych metod grafowych wraz z odniesieniami do literatury zawiera praca [157].

Wśród metod bezpośrednich podziału siatki można wyróżnić algorytmy podziałów geometrycznych i algorytmy zachłanne. Pierwsze wykorzystują geometryczne cechy obiektów siatki i polegają najczęściej na odpowiednich, rekurencyjnych podziałach zbioru obiektów siatki na części o określonej proporcji

wag. Podziały mogą być dokonywane np. na podstawie drzewiastej, rekurencyjnej dekompozycji przestrzeni (metody *octree*) lub uszeregowania odpowiednich obiektów (najczęściej wierzchołków elementów) wzdłuż pewnych linii (prostych, jak np. w metodzie inercyjnych bisekcji (*recursive inertial bisection*) lub łamanych, jak w metodzie krzywych wypełniających przestrzeń (*space filling curves*)). Algorytmy geometryczne charakteryzują się, podobnie jak metody wielopoziomowej rekurencyjnej bisekcji, liniową złożonością czasową względem liczby elementów (a więc i rozmiaru zadania N), dają jednak w wyniku podziały mniej optymalne [96].

Algorytmy zachłanne konstruują podobszary poprzez dodawanie do aktualnie rozważanego podobszaru kolejnych elementów i innych obiektów siatki. Istnieją różnorodne wersje tych algorytmów oparte na różnych strategiach doboru kolejnych elementów [74]. Wszystkie one powiązane są z klasycznym algorytmem przeszukiwania grafu wszerz [57]. Zazwyczaj jakość podziałów generowanych przez metody zachłanne nie jest najwyższa, mają one jednak nie tylko liniową złożoność czasową, ale często także niskie stałe w funkcji złożoności.

Przykładem zachłannego algorytmu podziału siatki MES, wykorzystującego geometryczne cechy triangulacji, jest przedstawiony poniżej algorytm `PS` [140], znajdowania podobszarów z użyciem frontu (kolejki) wierzchołków elementów. W jego definicji użyte są następujące oznaczenia:

- S – podobszar jako zbiór elementów i wierzchołków (krawędzie i boki dołączane są na późniejszym etapie),
- F – front, grupa wierzchołków sąsiadujących z wierzchołkami w S i będących kandydatami do dołączenia do S (front F powinien być zorganizowany w sposób umożliwiający efektywną realizację wyboru wierzchołka, np. jako kolejka priorytetowa),
- N_s^S – aktualna liczba stopni swobody w podobszarze,
- $N_{\max}^S$ – zadana maksymalna liczba stopni swobody w podobszarze,
- w – aktualnie rozważany wierzchołek frontu,
- e – aktualnie rozważany element,
- w_e – wierzchołek aktualnie rozważanego elementu.

Algorytm 3.1 (PS)

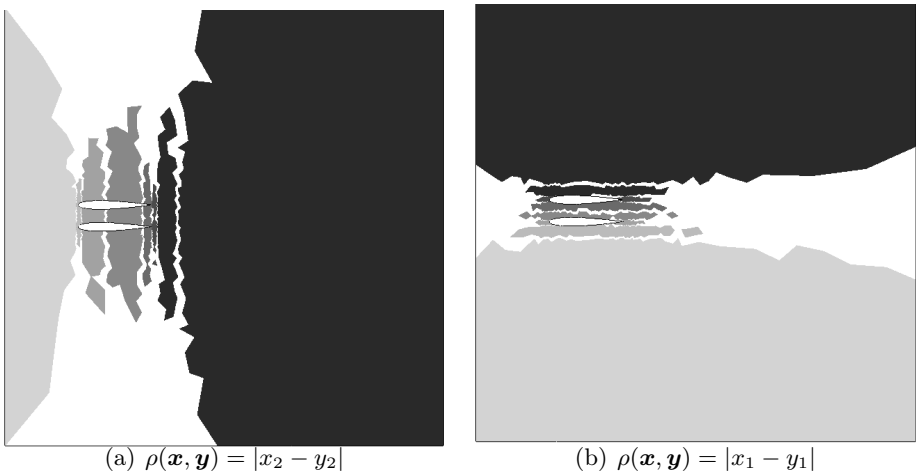
```
PS()
{
  inicjuj front F
  dla każdego podobszaru S {
    dopoki ( N_s^S < N_max^S ) {
      wybierz z frontu F wierzchołek w o ekstremalnej wadze
      jeżeli ( w ∉ S ) dodaj w do S
      dla każdego elementu e, ktorego wierzchołkiem jest w {
        jeżeli ( e ∉ S ) {
          dodaj e do S
          dla każdego wierzchołka w_e elementu e, w_e ≠ W {
            jeżeli ( w_e ∉ F i w_e ∉ S ) dodaj w_e do F
            jeżeli ( w_e ∉ S ) dodaj w_e do S
} } } } } }
```

Inicjowanie frontu polega na wyborze pojedynczego wierzchołka, najczęściej o ekstremalnych wartościach współrzędnych.

Specyfiką powyższego algorytmu jest stosowanie kryteriów geometrycznych przy wyborze kolejnych dołączanych węzłów. Celem jest uzyskanie podobszarów o określonych kształtach. Kształty te zależą od sposobu obliczenia wag związanych z wierzchołkami. Jeśli za wagę wierzchołka przyjmiemy jego odległość, w sensie pewnej normy, od początkowego wierzchołka podobszaru, to możemy otrzymać różne typy podziałów siatki, zależnie od przyjętej normy. Rysunek 3.4 przedstawia podziały przykładowej siatki MES, użytej w symulacjach przepływów wokół profili lotniczych, związane z różnymi normami geometrycznymi. Dla zilustrowanego dwuwymiarowego przykładu normy określają odległość $\rho(\boldsymbol{x}, \boldsymbol{y})$ miedzy punktami: $\boldsymbol{x}$ o współrzędnych (x_1, x_2) i $\boldsymbol{y}$ o współrzędnych (y_1, y_2). Białe fragmenty na rysunkach odpowiadają obszarom nakładek, które w tym przypadku mają szerokość jednego elementu.

Istotną cechą algorytmu PS jest możliwość operowania na elementach i wierzchołkach siatki o zadanej generacji. Wtedy dodanie elementu pociąga za sobą dodanie wszystkich jego potomków wraz z ich wierzchołkami. Oznacza to, że podziału siatki można dokonywać np. zawsze opierając się na siatce początkowej, co przynosi zmniejszenie czasu działania algorytmu i przydaje się w realizacji algorytmów wielosiatkowych oraz algorytmów powrotnego rozrzedzania siatki[8]. Inną zaletą algorytmu jest możliwość ustalania maksymalnej

[8]W bardziej rozbudowanej wersji, nie prezentowanej w niniejszej pracy, algorytm potrafi jednocześnie dokonywać podziału siatki opierając się na elementach i wierzchołkach danej



(a) $\rho(\boldsymbol{x}, \boldsymbol{y}) = |x_2 - y_2|$ (b) $\rho(\boldsymbol{x}, \boldsymbol{y}) = |x_1 - y_1|$

(c) $\rho(\boldsymbol{x}, \boldsymbol{y}) = \max(|x_1 - y_1|, |x_2 - y_2|)$ (d) $\rho(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$

Rys. 3.4. Różne podziały obszaru obliczeniowego w symulacji przepływu wokół profilu lotniczego dla czterech strategii doboru wierzchołków z frontu w algorytmie PS z s. 154, opartych na różnych definicjach odległości $\rho(\boldsymbol{x}, \boldsymbol{y})$

liczby stopni swobody odrębnie dla każdego podobszaru. Zmienny rozmiar podobszarów może być wykorzystywany w algorytmie równoważenia obciążenia procesorów [28].

Złożoność czasowa i pamięciowa algorytmu zależą od szczegółów implementacji. W przypadku zastosowania do przechowania frontu kolejki priorytetowej o logarytmicznym czasie wstawiania i pobierania elementów oraz tablic z adresowaniem bezpośrednim do przechowania informacji o przynależności do podobszarów, złożoność czasowa jest rzędu $N \log_2 N$, a pamięciowa rzędu N.

3.5.2 Równoważenie obciążenia i transfer obiektów siatki

Lokalne adaptacje siatki lub dynamiczne zmiany warunków pracy procesorów prowadzą do nierównowagi obciążenia procesorów. Aby ją przywrócić, konieczne są dwa działania: dokonanie nowego przypisania obiektów siatki i związanych z nimi struktur danych procesorom oraz efektywny transfer tych struktur między procesorami.

Pierwsze z zadań realizowane jest przez algorytmy ponownego podziału siatki (*mesh repartition*). Do sformułowania problemu dodaje się, prócz celów określonych dla pierwotnego podziału – minimalizacji powierzchni (obwodów) podobszarów i zachowania zrównoważenia obciążenia procesorów, dodatkowe wymaganie minimalizacji wielkości transferu obiektów siatki, koniecznego przy przejściu od starego do nowego podziału.

Zadanie to, podobnie jak zadanie dokonania pierwotnego podziału, jest problemem NP-zupełnym. Spośród różnorodnych heurystyk stosowanych do rozwiązania tego problemu [157, 88], najkorzystniejszymi są metody dyfuzji (*diffusion methods*), w których uwzględnia się przenikanie elementów przez granice między podobszarami. Najefektywniejsze wersje tych metod, podobnie jak w przypadku podziałów pierwotnych, wykorzystują mechanizm działania wielopoziomowego i osiągają liniową złożoność obliczeniową.

Mając dane nowe przyporządkowanie obiektów siatki podobszarom, dokonywany jest między procesorami transfer struktur danych związanych z obiektami siatki, w tym struktur odpowiadających stopniom swobody. Proces ten ma w dużym stopniu charakter techniczny i zależy od implementacji programu MES. Dla maksymalizacji ziarnistości obliczeń równoległych należy dokonywać transferu danych za pomocą komunikatów o możliwie dużych rozmiarach. Prowadzi to do konieczności grupowania danych przed właściwym przesyłaniem informacji. W przypadku dekompozycji obszaru na nakładające się podobszary,

generacji oraz tworzyć nakładkę na podstawie elementów i wierzchołków dowolnej innej generacji.

grupowanie i transfer muszą uwzględniać istnienie naładki. Konkretna postać procedur zależy od implementacji kodu (przykładowe rozwiązanie omówione jest w p. 4.3).

Jak analizowane jest to w p. 4.2.1, nie istnieje jedna optymalna struktura danych dla wszystkich rozwiązywanych zagadnień i typów aproksymacji MES. Pewne dane muszą być przechowywane w pamięci, wiele może być odtwarzanych w trakcie realizacji algorytmów. Przy wyborze struktury danych dla równoległej realizacji MES należy uwzględnić częstość i rozmiar spodziewanych transferów danych dotyczących obiektów siatki i stopni swobody. Aby zminimalizować czas poświęcony na transfer, można, dla potrzeb przesyłania danych, dobierać reprezentacje siatki o jak najmniejszej liczbie danych. Prowadzi to jednak do konieczności odtworzenia pozostałych wykorzystywanych w obliczeniach informacji. Do znalezienia optymalnej reprezentacji siatki i związanej z nią strategii transferu konieczne jest uwzględnienie tak cech problemu i aproksymacji, jak i parametrów systemu komputerowego, na którym dokonuje się obliczeń (mocy obliczeniowej procesorów oraz własności sieci).

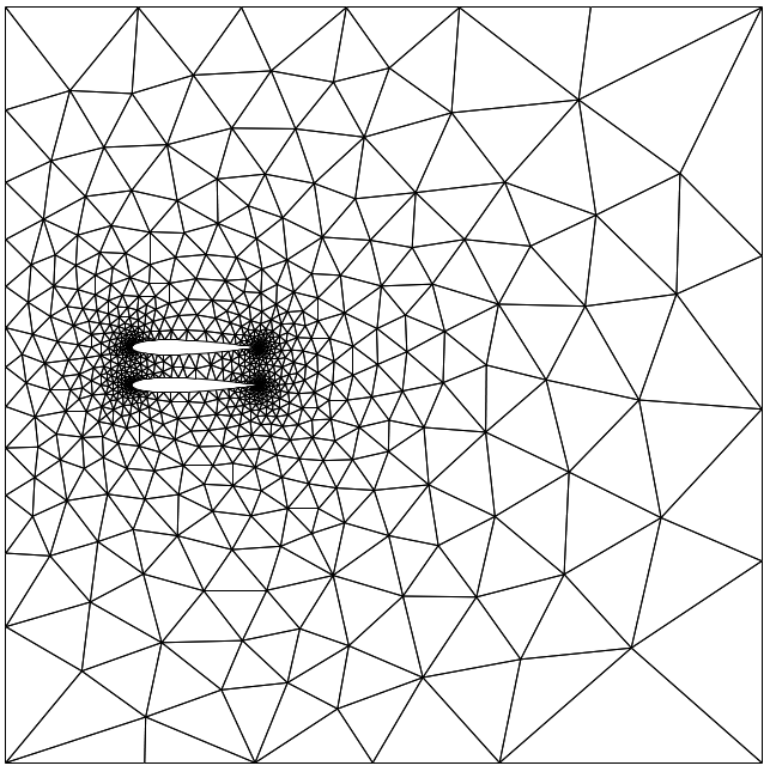
Proces transferu danych dotyczących obiektów siatki i stopni swobody zawsze ma złożoność czasową $O(N)$.

3.6 Przykłady obliczeń

3.6.1 Symulacja przepływu naddźwiękowego wokół profilu lotniczego

Pierwszym przykładem ilustrującym zagadnienia wydajności obliczeń równoległych jest symulacja nielepkiego przepływu o liczbie Macha 3 wokół profilu bi-NACA0012 [64], dla którego początkową siatkę elementów skończonych przdstawia rys. 3.5. Symulacja polega na przeprowadzeniu sekwencji obliczeń dla kolejnych siatek. Dla każdej siatki obliczenia składają się z uzyskania rozwiązania stacjonarnego i następnie adaptacji siatki na podstawie oszacowania błędu [17]. Rysunek 3.6 przedstawia fragment rozwiązania uzyskanego na piątej zaadaptowanej siatce.

Wykorzystaną strategią symulacji jest zastosowanie opisywanych już metod: niejawnej nieliniowej dyskretyzacji czasowej schematem Eulera, niedokładnej metody Newtona do rozwiązania problemu nieliniowego na każdym kroku czasowym i metody GMRES do rozwiązania powstałego układu równań. W metodzie GMRES zastosowana jest jednosiatkowa poprawa uwarunkowania macierzy algorytmem uogólnionej metody Gaussa-Seidla, ze względu na dobrą zbieżność solwera dla rozważanego przypadku.

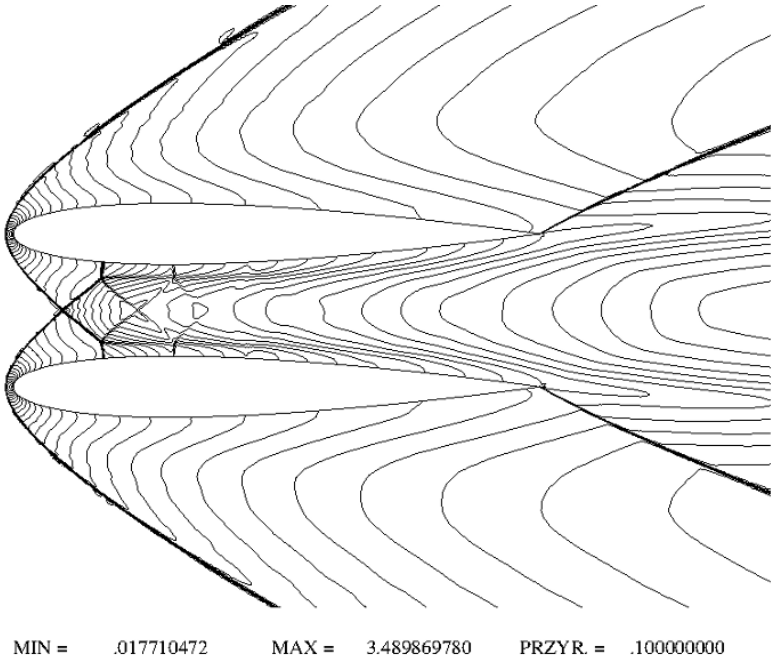


Rys. 3.5. Symulacja przepływu naddźwiękowego wokół profilu bi-NACA0012: siatka początkowa i geometria obszaru obliczeniowego

W celu realizacji równoległej użyto algorytm podziału siatki `PS` z s. 154. Za każdym razem stosowana była wersja oparta na normie euklidesowej. Jeden z podziałów siatki wykorzystywany w obliczeniach (dla 8 podobszarów), przedstawiony jest na rys. 3.4(d).

Ilustracją efektywności zrównoleglenia obliczeń są wyniki przedstawione w tabl. 3.1 i na rys. 3.7. Dotyczą one rozwiązania pojedynczego układu równań liniowych (wraz z utworzeniem macierzy układu) w wybranym, reprezentatywnym dla całej symulacji, kroku czasowym. Pokazują przyspieszenie obliczeń i efektywność zrównoleglenia dla liczby procesorów od 1 do 16. Liczba stopni swobody w rozwiązywanym problemie wynosi 86 060.

Maszyną, na której dokonano obliczeń, był komputer wieloprocesorowy HP



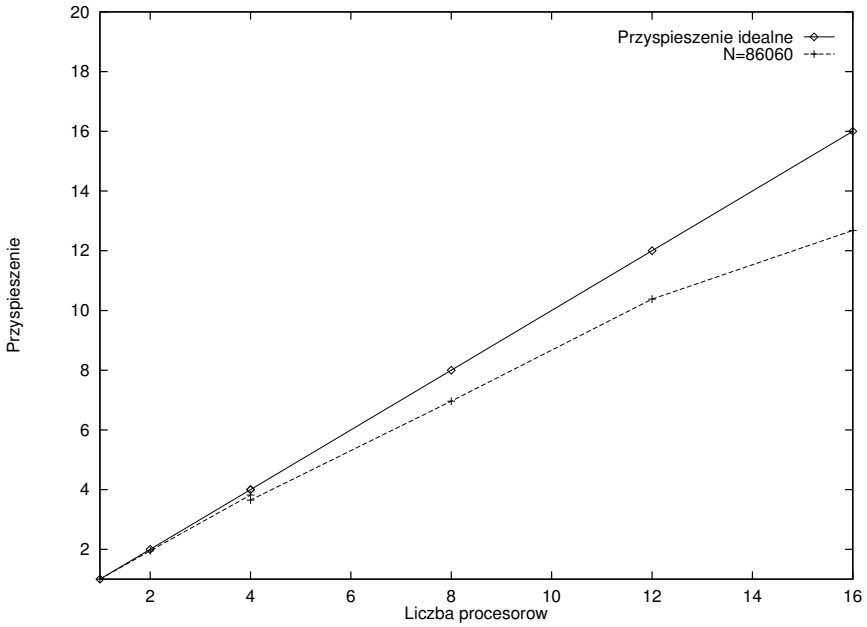
Rys. 3.6. Symulacja przepływu naddźwiękowego wokół profilu bi-NACA0012: rozwiązanie na piątej zaadaptowanej siatce w postaci izolinii liczby Macha

Exemplar SPP1600. Komputer ten wyposażony był (był – ponieważ został już wycofany z użytkowania) w węzły obliczeniowe, po cztery procesory w każdym z węzłów. Procesory pojedynczego węzła miały dostęp do szybszej lokalnej pamięci węzła. Jeżeli w obliczeniach wykorzystywano więcej niż jeden węzeł obliczeniowy, efektywność zrównoleglenia malała z powodu konieczności odwoływania się do globalnych zasobów pamięci.

Parametry wykonania przedstawione w tabl. 3.1 i na rys. 3.7 pokazują charakterystyczne zjawisko utraty liniowości przyspieszenia obliczeń i spadku efektywności zrównoleglenia przy wzroście liczby procesorów, nawet w przypadku algorytmu tak dobrze poddającego się zrównolegleniu, jak użyty algorytm GMRES z jednosiatkową poprawą uwarunkowania macierzy metodami Schwarza.

Tablica 3.1. Charakterystyki równoległego rozwiązania układu równań liniowych o 86 060 stopniach swobody w trakcie symulacji przepływu naddźwiękowego wokół profilu bi-NACA0012 na komputerze HP Exemplar SPP1600

Liczba procesorów	Czas obliczeń	Przyspieszenie	Efektywność
1 (1 węzeł)	112.97	—	—
2 (1 węzeł)	57.98	1.95	97.5%
4 (1 węzeł)	29.56	3.82	95.5%
4 (2 węzły)	30.91	3.65	91.5%
8 (2 węzły)	16.23	6.96	87%
12 (3 węzły)	10.88	10.38	86.5%
16 (4 węzły)	8.91	12.68	79%



Rys. 3.7. Wykres przyspieszenia obliczeń uzyskanego przy równoległym rozwiązywaniu układu równań liniowych o 86 060 stopniach swobody w trakcie symulacji przepływu naddźwiękowego wokół profilu bi-NACA0012 na komputerze HP Exemplar SPP1600

Tablica 3.2. Charakterystyki wykonania równoległego 10 iteracji metody GMRES z jednosiatkową poprawą uwarunkowania macierzy, przy rozwiązywaniu zagadnienia Laplace'a za pomocą nieciągłej aproksymacji Galerkina na sieci komputerów osobistych

N	N_p	Redukcja błędu $\cdot 10^3$	Szybkość zbieżności	Czas obliczeń	Przyspieszenie	Efektywność
48 896	1	0.288	0.443	2.26	1.00	100%
	2	0.328	0.448	1.16	1.94	97%
	4	0.340	0.450	0.61	3.70	92%
	8	0.361	0.452	0.36	6.28	78%
391 168	1	9.313	0.626	17.85	1.00	100%
	2	10.173	0.632	8.93	1.99	100%
	4	10.252	0.633	4.53	3.94	98%
	8	11.183	0.638	2.34	7.63	95%
3 129 344	2	48.041	0.738	70.76	1.00	100%
	4	47.950	0.738	35.63	1.98	99%
	8	48.748	0.739	17.71	3.99	100%

3.6.2 Nieciągła aproksymacja równania Laplace'a

Drugim przykładem obliczeniowym jest równoległa realizacja obliczeń dla omawianego już wcześniej (patrz pp. 2.6.6 i 2.7.11) zastosowania nieciągłej aproksymacji Galerkina wraz z solwerem GMRES wykorzystującym jedno- i wielopoziomową poprawę uwarunkowania macierzy układu do rozwiązania modelowego problemu dyfuzji w sześcianie jednostkowym.

W tym przypadku maszyną równoległą była sieć Linuxowych komputerów osobistych z procesorem Pentium 4, 1.6 GHz i pamięcią 1 GBajt każdy, połączonych w technologii szybkiego Ethernetu 100 Mbit/s.

Tablice 3.2 i 3.3 prezentują wyniki dla dwóch wersji poprawy uwarunkowania macierzy, jednosiatkowej i trzysiatkowej, oraz różnych rozmiarów zadania, odpowiadających kolejnym siatkom uzyskanym przez jednorodne zagęszczanie. Metodą poprawy uwarunkowania jest w każdym przypadku równoległa uogólniona metoda Gaussa-Seidla ze zmodyfikowanymi łatami elementów (patrz p. 2.7.9). N, jak zwykle, oznacza całkowitą liczbę stopni swobody będącą miarą wielkości problemu, a N_p liczbę użytych komputerów (procesorów). Dla każdej kombinacji metody poprawy uwarunkowania i rozmiaru zadania odno-

Tablica 3.3. Charakterystyki wykonania równoległego 10 iteracji metody GMRES z trzysiatkową poprawą uwarunkowania macierzy, przy rozwiązywaniu zagadnienia Laplace'a za pomocą nieciągłej aproksymacji Galerkina na sieci komputerów osobistych

N	N_{p}	Redukcja błędu $\cdot 10^3$	Szybkość zbieżności	Czas obliczeń	Przyspieszenie	Efektywność
391 168	1	0.018	0.335	26.18	1.00	100%
	2	0.017	0.334	14.18	1.85	92%
	4	0.018	0.335	9.08	2.88	72%
	8	0.024	0.346	7.60	3.44	43%
3 129 344	2	0.027	0.350	111.16	1.00	100%
	4	0.027	0.350	57.76	1.92	96%
	8	0.027	0.348	33.15	3.35	84%

towane są wyniki obliczeń dla 1, 2, 4 i 8 komputerów w sieci. Dla największego zadania nie uzyskano wyników przy użyciu jednego komputera, gdyż zadanie nie mieściło się w pamięci operacyjnej.

Aby otrzymać wyniki odzwierciedlające czysto obliczeniową efektywność zrównoleglenia, rezultaty w tabl. 3.2 i 3.3 odpowiadają zawsze 10 iteracjom metody GMRES. Dodatkową ilustracją zbieżności są zamieszczone w tablicach: szybkość zbieżności GMRES (obliczona wg wzoru (2.93)) i wielkość względnej redukcji residuum układu równań po 10 iteracjach (podzielona przez 10^{-3}). Ze względu na równoważność zastosowanej poprawy uwarunkowania kombinacji metod Schwarza szybkość zbieżności metody GMRES z jednosiatkową poprawą uwarunkowania maleje wraz ze wzrostem liczby procesorów. W przypadku metod wielosiatkowych, dzięki zastosowaniu szerokiej strefy nakładania się podobszarów, określonej dla wszystkich poziomów przez wymagania lokalności obliczeń na siatce najrzadszej, udaje się uzyskać praktycznie stałą szybkość zbieżności. Okupione jest to zmniejszeniem efektywności zrównoleglenia obliczeń dla wzrastającej liczby procesorów.

Wyniki w tablicach dobrze ilustrują skalowalność różnych wersji algorytmu. Dla jednosiatkowej poprawy uwarunkowania skalowane przyspieszenie obliczeń jest zbliżone do liniowego. Czas rozwiązania zadania o 48 896 stopniach swobody na pojedynczym procesorze praktycznie nie odbiega od czasu rozwiązania zadania ośmiokrotnie większego na ośmiu procesorach. Podobnie ma się czas rozwiązania zadania o 391 168 stopniach swobody na jednym procesorze do

czasu rozwiązania zadania o 3 129 344 stopniach swobody (osiem razy więcej niż poprzednio) na ośmiu procesorach. Gorzej prezentuje się skalowalność wersji trzysiatkowej, ale i tutaj czasy wykonania dla ostatniej wymienianej pary problemów są zbliżone.

Rozdział 4

Modularna architektura rdzenia obliczeniowego MES i realizacja obliczeń równoległych

Analizy przeprowadzone w poprzednich rozdziałach wskazują na złożony charakter współczesnych wyrafinowanych aproksymacji metodą elementów skończonych. Umieszczenie w pojedynczym programie aproksymacji wyższego stopnia, adaptacji *hp* i solwerów liniowych z wielopoziomową poprawą uwarunkowania macierzy stanowi wyzwanie dla strategii implementacji. Złożony charakter algorytmów skłania do badań nad architekturą kodu. Celem jest osiągnięcie programu, który pomimo złożoności, byłby łatwo modyfikowalny, rozszerzalny o nowe metody i techniki, wreszcie łatwy w pielęgnacji. Klasycznym rozwiązaniem tego problemu jest modularna struktura kodu, z wyodrębnieniem modułów (komponentów) stanowiących możliwie niezależne części [133]. Dodatkowym wymaganiem, związanym z przeznaczeniem programu do obliczeń wielkiej skali, jest utrzymanie wysokiej wydajności obliczeń, wymaganie często sprzeczne z założeniem modularności. Przeznaczenie do obliczeń wielkiej skali oznacza jednocześnie, że program musi być przystosowany do realizacji równoległej.

Podstawą rozważań niniejszego rozdziału jest próba stworzenia modularnej architektury systemów MES, dla której ostateczne programy składane są z odpowiednich komponentów w taki sposób, że niezależnie można otrzymywać wersje sekwencyjne i równoległe. Wersje równoległe wykorzystują pewne pod-

stawowe moduły, które w niezmienionej postaci mogą także tworzyć programy sekwencyjne, i dodają do tego dodatkowe moduły, które w pełni kontrolują przebieg realizacji równoległej.

Podstawowym przedmiotem analiz niniejszego rozdziału jest implementacja rdzenia obliczeniowego MES. Rdzeniem obliczeniowym nazywane są te algorytmy, które występują we wszystkich typowych obliczeniach MES i decydują o charakterze aproksymacji i jej numerycznej efektywności. Są to algorytmy modyfikacji siatki MES, całkowania numerycznego wyrazów ze sformułowania słabego, tworzenia odpowiadającego sformułowaniu słabemu układu równań liniowych i rozwiązania tego układu. Jeśli rozwiązanie układu równań liniowych realizowane jest przez zewnętrzny solwer, to składnikiem rdzenia obliczeniowego MES pozostaje interfejs z solwerem, realizujący ważne funkcje organizacji procesu obliczeniowego [54].

Poza rdzeniem pozostają zależne od problemu procedury realizujące dyskretyzację czasową, rozwiązywanie równań nieliniowych czy specjalne sprzężenia problemów MES. Jako zewnętrzne w stosunku do rdzenia traktowane są także moduły lub oddzielne programy wspomagające obliczenia MES, odpowiedzialne m.in. za *preprocessing* (modelowanie geometryczne, generację siatki), *postprocessing* (wizualizację), dekompozycję obszaru obliczeniowego na podobszary, czy, ważne w rozbudowanych środowiskach rozwiązywania problemów, zadania monitorowania i sterowania.

Niniejszy rozdział przedstawia ogólną koncepcję modularnej architektury programów MES oraz podstawy pewnej szczególnej jej implementacji. Do konstruowania tej szczególnej implementacji wykorzystane są rozważania i analizy poprzednich rozdziałów pracy. Detale implementacji, mające charakter techniczny i mniej interesujące z naukowego punktu widzenia, są pominięte w rozważaniach. Rozdział zawiera ponadto opis typowego przebiegu obliczeń równoległych w ramach prezentowanej implementacji. Implementacja ta wykorzystywana jest we wszystkich przykładach obliczeniowych zaprezentowanych w pracy.

4.1 Modularna architektura rdzenia obliczeniowego MES

Architektura programów MES proponowana w niniejszej pracy składa się z szeregu modułów, które można ułożyć w odpowiednie warstwy implementacji. Rysunki 4.1 i 4.2 przedstawiają w postaci diagramów UML [153] proponowaną architekturę w przypadkach, odpowiednio, kodu sekwencyjnego i kodu równo-

ległego. Linie ciągłe łączą istotne dla omawianej architektury interfejsy, reprezentowane przez okręgi, z implementującymi je modułami. Linie przerywane zakończone strzałkami oznaczają wywołania procedur wyspecyfikowanych w interfejsach lub zawartych w zewnętrznych bibliotekach i programach.

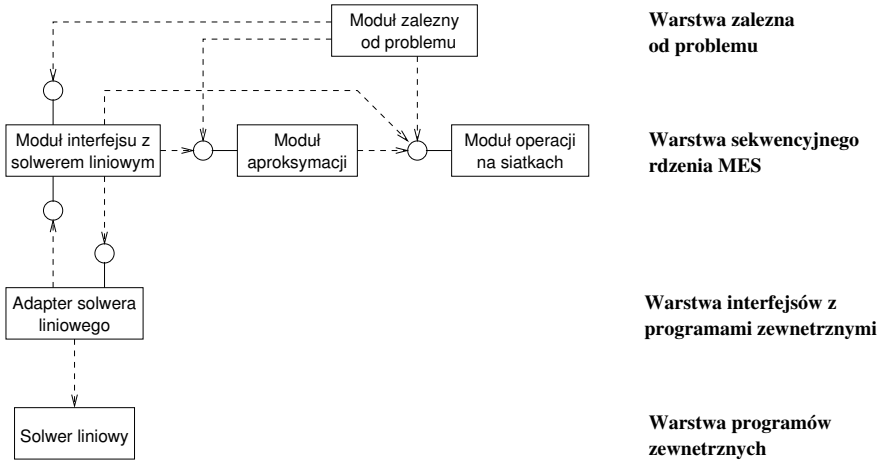
Na szczycie warstw implementacji znajduje się warstwa zależna od problemu. W tworzącym ją module zgromadzone są te procedury, które programista ostatecznej wersji kodu, przystosowanej do rozwiązywania konkretnej klasy problemów, musi najczęściej napisać sam. Celem niniejszej, jak i wielu innych (por. np. [128]), propozycji architektury kodów MES jest zminimalizowanie tej warstwy i dostarczenie całej pozostałej infrastruktury obliczeniowej, w tym infrastruktury do obliczeń równoległych, w postaci procedur gotowych do wykorzystania bez żadnych modyfikacji.

Ta pozostała infrastruktura obliczeniowa składa się z procedur specyficznych dla MES, tworzących jej rdzeń obliczeniowy, oraz zewnętrznych programów i bibliotek, których procedury realizują zadania wspomagające. Zewnętrzne programy i biblioteki stanowią najniższą warstwę w omawianej architekturze. Typowymi przykładami są tutaj biblioteki algebry liniowej, a dla wersji równoległych biblioteki przesyłania komunikatów czy programy dzielenia siatki MES.

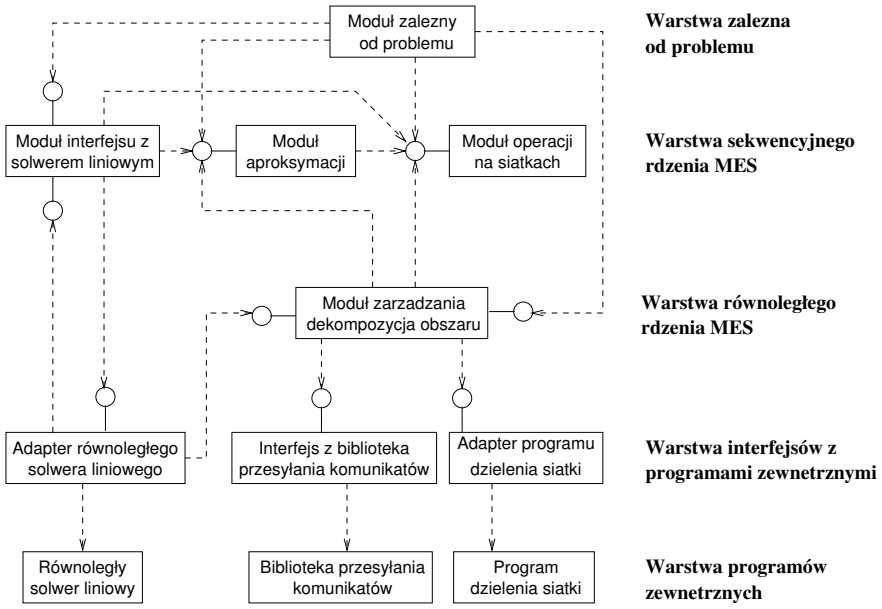
Interakcje procedur rdzenia i procedur zewnętrznych mogą mieć mniej lub bardziej złożony charakter. W przypadku prostych, jednokierunkowych interakcji w procedurach rdzenia umieszcza się wywołania funkcji realizowanych przez procedury zewnętrzne. Najczęściej nazwy tych funkcji są specyficzne dla implementacji procedur rdzenia i wymagają pośredniej warstwy interfejsu dla konkretnych implementacji procedur zewnętrznych. Taka warstwa pośrednia – adapter modułu zewnętrznego – pozwala uniezależnić rdzeń obliczeniowy MES od konkretnych realizacji funkcji zewnętrznych.

Bardziej złożony charakter mają interakcje rdzenia obliczeniowego MES z programami zewnętrznymi, np. w przypadku użycia wielosiatkowych solwerów równań liniowych. Zależnie od realizacji solwera, konieczne może stać się wtedy zastosowanie wywołań zwrotnych procedur rdzenia przez procedury solwera. Wywołania te może realizować warstwa adaptera solwera [22], tak jak jest to przedstawione na rys. 4.1 i 4.2.

Środkowe warstwy w proponowanej architekturze kodów MES są tworzone przez procedury rdzenia obliczeniowego. Nowością tej architektury w porównaniu z innymi koncepcjami jest rozbicie rdzenia obliczeniowego na dwie niezależne warstwy – sekwencyjną i równoległą. Warstwę sekwencyjną tworzą moduły, które bez zmian umieszczane mogą być w kodach równoległych i se-



Rys. 4.1. Proponowana architektura dla sekwencyjnych kodów MES



Rys. 4.2. Proponowana architektura dla równoległych kodów MES

kwencyjnych, a w trakcie wykonania nie mają świadomości czy pracują w kodzie sekwencyjnym, czy równoległym. Warstwę równoległą tworzą moduły w pełni odpowiedzialne za efektywną realizację obliczeń w środowiskach równoległych.

Drugą nowością proponowanej architektury jest rozbicie warstwy sekwencyjnej na odrębne moduły, takie jak: moduł operacji na siatkach, moduł aproksymacji i moduł interfejsu z solwerem liniowym. Taka modularyzacja oznacza alternatywę w stosunku do modelu implementacji MES, w którym podstawowym pojęciem jest element, łączący cechy topologiczne, geometryczne i aproksymacyjne (por. np. model architektury MES w ujęciu obiektowym w [103]).

Dla omawianej w niniejszej pracy metodologii zrównoleglenia obliczeń MES prezentowana architektura kodu przewiduje, jako podstawowy moduł organizujący przebieg obliczeń równoległych, moduł zarządzania dekompozycją obszaru.

W następnych punktach scharakteryzowane są pokrótce podstawowe moduły rdzenia obliczeniowego MES. Prezentacja ma charakter abstrakcyjny, koncentrując się na interfejsach modułów[1]. Jest założeniem proponowanej architektury, że poszczególne moduły zdefiniowane są wyłącznie poprzez ich interfejsy, co gwarantuje ogólność definicji i pozostawia swobodę realizacji przy użyciu różnych technik i języków programowania. Uwagi dotyczące implementacji modułów zawarte są w p. 4.2.

4.1.1 Moduł operacji na siatkach

Procedury modułu operacji na siatkach odpowiedzialne są za wczytywanie, przechowywanie i modyfikowanie informacji dotyczących siatek elementów skończonych oraz za udostępnianie tych informacji wszystkim pozostałym modułom kodu. Modyfikacje siatki obejmują różnorakie typy zagęszczania i rozrzedzania siatki, takie jak: podziały elementów (zagęszczanie siatki, adaptacja typu h), przesunięcia wierzchołków elementów (adaptacja typu r), łączenie elementów powstałych z uprzednich podziałów (rozrzedzanie siatki, także należące do adaptacji typu h).

Moduł operacji na siatkach (nazywany dalej także po prostu modułem siatki) jest modułem najłatwiejszym do wydzielenia z całości programu MES. Jego procedury nie pobierają żadnych danych z pozostałych modułów rdzenia obliczeniowego. W przypadku złożonej geometrii obszarów obliczeniowych,

[1] W niniejszej pracy jedynym wykorzystywanym elementem interfejsów modułów są nagłówki procedur wywoływanych przez funkcje zewnętrzne. Bardziej szczegółowy opis interfejsów znajduje się w pracy [22].

procedury tworzenia nowych wierzchołków elementów mogą potrzebować interakcji z traktowanym jako zewnętrzny w stosunku do rdzenia obliczeniowego modułem modelowania geometrycznego. Podobnie w przypadku adaptacji poprzez *remeshing* [185, 104] wymagane jest współdziałanie modułu operacji na siatkach i generatora siatek. Nie ma natomiast przepływu informacji do modułu siatki np. o używanej aproksymacji czy rozwiązywanym problemie.

Zakłada się, że moduł może dokonywać operacji na kilku siatkach w ramach pojedynczej symulacji (w przypadku kilku pól aproksymacji w tym samym obszarze lub w przypadku sprzęgnięcia aproksymacji w różnych obszarach). Sprzężenie aproksymacji na różnych siatkach jest dokonywane przez moduły zewnętrzne w stosunku do modułu siatki. Siatki mogą być siatkami hybrydowymi [97, 105], złożonymi z elementów różnych typów. Oddzielenie modułu aproksymacji od modułu operacji na siatkach pozwala na przeprowadzanie adaptacji w przypadku wielu pól aproksymacji. Wprowadza jednocześnie komplikacje przy przeprowadzaniu modyfikacji siatki, która musi być realizowana w dwóch fazach [22].

W przypadku rozbudowanej struktury danych liczba procedur interfejsu modułu siatki sięga kilkudziesięciu. Poniżej przedstawione jest kompletne zestawienie grup procedur występujących w interfejsie modułu operacji na siatkach:

- procedury inicjacji i operacji wejścia/wyjścia importujące dane z plików lub innych modułów oraz eksportujące dane, w różnych możliwych formatach,
- procedury zwracające globalne informacje o siatce MES, takie jak: opis siatki, liczba elementów, liczby innych obiektów siatki,
- procedury informujące o atrybutach konkretnych obiektów siatki (patrz p. 4.2.1, s. 176),
- procedury umożliwiające wykonanie pętli po obiektach danego rodzaju,
- procedury podające informacje o przynależności obiektów niższego wymiaru do obiektów wyższego wymiaru (np. dla krawędzi – do których elementów przynależy) oraz o posiadaniu obiektów niższego wymiaru przez obiekty wyższego wymiaru (np. dla boku – które wierzchołki posiada),
- procedury zwracające dla obiektów siatki informacje o sąsiadach (przede wszystkim dla elementów),

- procedury informujące o „rodzinie” (rodzicu i dzieciach) danego obiektu siatki,
- procedury zwracające dodatkowe informacje związane z obiektami siatki, najczęściej przekazywane przez generator siatki, takie jak np. typ warunku brzegowego dla boków czy typ materiału dla elementów,
- procedury informujące o względnym położeniu jednych obiektów siatki względem drugich (procedury te mogą np. zwracać współczynniki transformacji pomiędzy lokalnymi współrzędnymi dla dwóch częściowo pokrywających się obiektów siatki),
- dla szczególnego przypadku obiektów geometrycznie liniowych, procedury podające informacje geometryczne związane z obiektem, np. rozmiar obiektu,
- procedury modyfikacji siatki: jej inicjacji i finalizacji oraz podziałów i złączeń obiektów siatki,
- procedury wpisujące dane o obiektach siatki do struktury danych modułu oraz usuwające je z niej.

4.1.2 Moduł aproksymacji

W proponowanej architekturze na moduł aproksymacji składają się wszelkie procedury dokonujące operacji związanych ze zbiorami stopni swobody i funkcjami kształtu. Elementarnymi procedurami modułu aproksymacji są procedury zapisujące, odczytujące i przepisujące wartości stopni swobody związane z pojedynczym obiektem siatki. Z punktu widzenia obliczeń MES, podstawowym zadaniem modułu aproksymacji jest dokonanie całkowania numerycznego wyrażeń ze sformułowania skończenie elementowego. Istotną grupą procedur w ramach modułu aproksymacji są także procedury dokonujące różnorodnych typów rzutowania i interpolacji. Interpolacje wykorzystują zazwyczaj jawne wzory, projekcje często związane są z rozwiązaniem lokalnych problemów (dla pojedynczych elementów lub innych obiektów siatki). Interpolacje i projekcje stosowane są np. do przypisywania wartości geometrycznym stopniom swobody obiektów siatki w przypadku elementów krzywoliniowych, do nadawania wartości początkowej stopniom swobody, do przypisywania wartości stopniom swobody obiektów tworzonych w trakcie adaptacji, do przeprowadzania prolongacji i restrykcji w trakcie wielosiatkowego rozwiązywania równań liniowych,

do nadawania wartości z warunków brzegowych stopniom swobody obiektów na brzegu obszaru obliczeniowego.

Ścisłe powiązanie procedur aproksymacyjnych z operacjami na siatkach utrudnia wydzielenie modułu aproksymacji z całości kodu MES. Funkcje kształtu definiowane są odrębnie dla każdego typu elementów. Podobnie kwadratury całkowania numerycznego zależą od rodzaju obszaru całkowania. Idącym najdalej rozwiązaniem kwestii powyższych zależności jest tworzenie odrębnych modułów aproksymacji dla poszczególnych typów siatek. Nie jest to jednak rozwiązanie optymalne. Spośród procedur modułu aproksymacji można wyróżnić podzbiór funkcji zależnych od typu siatki i podzbiór funkcji niezależnych, takich jak np. ogólna procedura całkowania numerycznego. W ramach rozważanej architektury rozwiązaniem optymalnym wydaje się tworzenie różnych modułów dla istotnie różnych typów siatek (np. siatek w obszarach dwu- i trójwymiarowych) i wyróżnianie w ich ramach podmodułów dla siatek zbliżonych (np. czworościennych i sześciennych).

Podobnie jak w przypadku modułu siatki, moduł aproksymacji może operować na kilku swoich podstawowych obiektach, w tym wypadku polach aproksymacji. Koordynacja działań związanych z różnymi polami odbywa sie poza modułem aproksymacji.

4.1.3 Moduł interfejsu z solwerem równań liniowych

W proponowanej architekturze programów MES solwer układów równań liniowych realizuje ogólny interfejs, dostosowany tak do metod bezpośrednich, prostych metod iteracyjnych, jak i do złożonych iteracyjnych solwerów wielosiatkowych. Zamysłem jest istnienie kilku tylko procedur interfejsu po stronie solwera. Procedury te realizowane są przez adapter solwera, stanowiący warstwę pośrednią pomiędzy kodem MES i niezależnym zewnętrznym solwerem. Podstawowe interakcje pomiędzy solwerem i resztą kodu odbywają się poprzez wywołania procedur programu MES, tworzących moduł interfejsu z solwerem równań liniowych, przez procedury adaptera solwera liniowego. Procedury modułu interfejsu z solwerem realizują interfejs dostosowany do współpracy z rozmaitymi solwerami równań liniowych. Rozwiązanie takie daje dużą elastyczność w doborze solwera liniowego dla różnych zagadnień, choć związane jest z koniecznością wyposażenia kodu solwera w procedury adaptera, wywołujące procedury kodu MES.

Interfejs realizowany przez moduł interfejsu z solwerem zawiera procedury umożliwiające tworzenie siatek różnych poziomów, przekazywanie bloków macierzy składających się na globalną macierz układu równań liniowych oraz prze-

kazywanie rozwiązania początkowego do solwera i końcowego z solwera. Posiada także procedury wspomagające rzutowanie rozwiązań pomiędzy siatkami różnych poziomów w przypadku obliczeń wielosiatkowych.

W programie może istnieć wiele solwerów, np. odrębne solwery związane z odrębnymi podproblemami problemu sprzężonego. Koordynacją ich działania zajmują się procedury modułu zależnego od problemu.

Procedura rozwiązania układu równań liniowych, podstawowa procedura zawarta w interfejsie adaptera solwera, posiada dwa tryby pracy. W trybie rozwiązania całości układu, wypełnia strukturę danych solwera odpowiednimi wartościami i dokonuje pełnego rozwiązania układu równań. W trybie rozwiązania dla nowej prawej strony, uzyskuje jedynie nowy wektor prawej strony i dokonuje rozwiązania wykorzystując istniejące struktury związane z macierzą układu. Z analiz złożoności obliczeniowej algorytmów rozwiązywania równań liniowych wynika, że, zarówno dla metod bezpośrednich, jak i optymalnych metod iteracyjnych, rozwiązanie dla nowej prawej strony ma niższy rząd złożoności czasowej niż rozwiązanie całości układu. Do efektywnej realizacji obliczeń wielkiej skali istnienie dwóch wymienionych trybów pracy solwera jest więc konieczne.

4.1.4 Moduł zarządzania dekompozycją obszaru

Ze względu na przyjęcie dekompozycji obszaru za podstawę zrównoleglenia kodów MES moduł odpowiedzialny za realizację obliczeń równoległych nazywany jest modułem zarządzania dekompozycją obszaru (w skrócie modułem dekompozycji). Procedury tego modułu są jedynymi procedurami rdzenia obliczeniowego MES świadomymi faktu wykonania równoległego w trakcie realizacji obliczeń. Procedury modułów sekwencyjnych działają w sposób identyczny, niezależnie od tego czy program jest uruchamiany na jednym, czy na wielu procesorach.

Procedury modułu dekompozycji wywoływane są w celu przeprowadzenia takich podstawowych dla wykonania równoległego operacji, jak: podział obszaru obliczeniowego na podobszary, tworzenie nakładki czy równoważenie obciążenia procesorów. Pełnią one także rolę pomocniczą w adaptacji, przy realizacji której uzgadniają ostateczną listę modyfikowanych obiektów uwzględniając założoną nieregularność siatki oraz utrzymują spójność struktur danych na różnych procesorach, po dokonaniu lokalnych modyfikacji siatki.

W trakcie realizacji obliczeń solwera liniowego procedury modułu dekompozycji pośredniczą przy wykonywaniu globalnych operacji wektorowych redukcji (iloczyn skalarny, norma) i przeprowadzają wymianę wartości w rozproszonych

fragmentach globalnych wektorów przy obliczaniu ich iloczynów z macierzą układu równań liniowych oraz przy wygładzaniu błędu (poprawie uwarunkowania macierzy). Poza tym umożliwiają uzyskanie podstawowych informacji o parametrach wykonania równoległego, takich jak liczba wykorzystywanych procesorów (podobszarów) czy identyfikator bieżącego procesora, a także dostarczają funkcje realizujące podstawowe operacje grupowego przesyłania komunikatów, takie jak rozgłaszanie (*broadcast*), rozpraszanie (*scatter*) i zbieranie (*gather*).

4.2 Implementacja modułów

W punkcie niniejszym omówiona jest przykładowa implementacja scharakteryzowanych powyżej modułów. Nadal jednak prezentacja nie wchodzi w szczegóły realizacji – konkretne struktury danych, języki programowania – zwracając uwagę tylko na pewne aspekty istotne z punktu widzenia obliczeń wielkiej skali i związane z dyskutowanymi w pracy algorytmami i metodologią zrównoleglenia. Ze względu na ograniczenie niniejszych rozważań do elementów specyficznych dla kodów MES nie jest dyskutowana implementacja wielosiatkowych solwerów równań liniowych, mimo że daje ona pole do szczególnie wyrafinowanych technik realizacji[2].

Definiowanie modułów poprzez określenie ich interfejsów nie przesądza o ostatecznej implementacji. Jednak jedną z podstaw proponowanej modularyzacji kodów MES jest powiązanie każdego z modułów z pewną podstawową strukturą danych stosowaną w implementacjach. Strukturami tymi są:

- struktura danych siatek MES dla modułu operacji na siatkach,
- struktura danych pól aproksymacji dla modułu aproksymacji,
- struktura danych przenumerowania stopni swobody dla modułu interfejsu z solwerem liniowym,

[2]Użycie odrębnego typu danych dla całości solwera pozwala na łatwe posługiwanie się wieloma egzemplarzami solwera w ramach jednego modułu. Użycie odrębnego typu danych dla pojedynczych poziomów siatki umożliwia zróżnicowanie operacji na poszczególnych poziomach, np. przypisanie każdemu poziomowi innej metody przybliżonego lub dokładnego (dla siatki najrzadszej) rozwiązania układu. Format przechowywania macierzy układu może być różny na każdym poziomie. Zróżnicowanie algorytmów i struktur danych na różnych poziomach może mieć istotne znaczenie przy realizacji równoległej, jeśli zróżnicowany rozmiar zadania na różnych poziomach sprawi, że optymalnymi są odmienne dla różnych poziomów dystrybucje danych między procesorami.

- struktura danych utrzymująca spójność rozproszonych struktur danych siatek i pól aproksymacji dla modułu zarządzającego dekompozycją obszaru.

Powiązanie określonych interfejsów definiujących moduły z podstawowymi strukturami danych zbliża proponowaną architekturę do rozwiązań obiektowych. Możliwa jest jednak implementacja tak określonej architektury w językach nieobiektowych, takich jak C czy Fortran95. Możliwe, i do pewnego stopnia naturalne, hierarchie klas odpowiadające modułom nie stanowią wymaganego elementu architektury.

4.2.1 Moduł operacji na siatkach

W przedstawianych w niniejszej pracy opisach założono, że siatka MES złożona jest z następujących obiektów składowych (obiektów siatki):

- elementy,
- boki,
- krawędzie,
- wierzchołki.

Opis ten najlepiej pasuje do siatek w obszarach trójwymiarowych. Dla zagadnień w obszarach dwuwymiarowych znika pojęcie krawędzi, natomiast w obszarach jednowymiarowych dodatkowo znika pojęcie boku i pozostają wyłącznie elementy i ich wierzchołki. Nie wszystkie wymienione powyżej obiekty siatki muszą być reprezentowane w strukturze danych.

Spośród obiektów siatki najważniejszym jest element. Z teoretycznego punktu widzenia, element określa się jako złożony z trzech składowych: obszaru w przestrzeni, zbioru funkcji kształtu (definiujących lokalną przestrzeń funkcyjną) i zbioru stopni swobody (jako funkcjonałów określonych na funkcjach z lokalnej przestrzeni) [52]. Dwie ostatnie składowe w całości zwiazane są z polem aproksymacji. Pierwsza składowa definiowana jest przez położenie obiektów tworzących element: boków, krawędzi, wierzchołków. Także w tym przypadku dane geometryczne mogą być związane z odpowiednią aproksymacją. Tym, co zawsze pozostaje w gestii modułu siatki są zależności topologiczne – wzajemne przynależności, zawierania i sąsiedztwo obiektów siatki.

Nie istnieje w ramach modułu siatki pojęcie węzła, które w klasycznej MES obejmuje położenie geometryczne i zbiór stopni swobody. Wszelkie odniesienia

do stopni swobody przejęte są przez moduł aproksymacji. Procedury modułu aproksymacji mogą realizować model klasyczny lub inne sposoby określania pól aproksymacji dla danej siatki.

Obiekty siatki charakteryzowane są w strukturze danych poprzez odpowiednie atrybuty. Poniżej przedstawiony jest typowy zestaw takich atrybutów (nie wszystkie atrybuty stosuje się do wszystkich obiektów – w nawiasach podane jest dla jakich obiektów dany atrybut posiada sens):

- typ (elementy, boki) – dla elementu np. trójkąt, sześcian, czworościan, dla boku np. trójkąt, czworokąt;
- stan (elementy, boki, krawędzie) – *aktywny* lub *nieaktywny*. Aktywne są te obiekty, które nie zostały podzielone w trakcie adaptacji i biorą udział w obliczeniach na siatce ostatecznej, najrzadszej. Nieaktywne są te obiekty, które zostały podzielone w trakcie adaptacji. Obiekty te mogą nadal istnieć w strukturze danych siatki MES i być uwzględniane np. przy obliczeniach na siatkach zgrubnych czy przy powrotnym rozrzedzaniu uprzednio zagęszczonej siatki;
- rodzic (elementy, boki, krawędzie) – wszystkie obiekty, które mogą być dzielone w trakcie adaptacji, mogą także powstawać w wyniku podziału określonego obiektu. Obiekt ten jest dla nich obiektem-rodzicem. W przypadku elementów brak rodzica oznacza przynależność do siatki początkowej;
- dzieci[3] (elementy, boki, krawędzie) – posiadanie dzieci jest jednoznaczne z nieaktywnością obiektu.

Jednoczesne występowanie w strukturze danych siatki MES informacji o obiektach aktywnych i nieaktywnych, tworzących różne generacje, może zostać wykorzystane przy tworzeniu siatek różnych poziomów na potrzeby wielosiatkowych solwerów równań liniowych (patrz p. 2.7.4). Jednak informacja o kolejnych poziomach siatki nie musi, a wręcz nie powinna być w jawny sposób zawarta w module siatki. Siatki różnych poziomów dla obliczeń wielosiatkowych powinny być tworzone przez procedury zewnętrzne w stosunku do modułu siatki, posługujące się jedynie informacją o rodzinach (rodzic–dzieci) obiektów siatki.

Istotnym rodzajem informacji o obiektach siatki, niezbędnym do przeprowadzenia adaptacji, jest informacja o sąsiedztwie. Minimalny zestaw danych

[3]Określenie *potomkowie* jest terminem ogólniejszym, oznaczającym także dzieci dzieci, itd.

tego typu zawiera informacje o aktywnych sąsiadach poprzez bok dla każdego aktywnego elementu (sąsiadem poprzez dany bok jest element, który zawiera albo ten bok, albo jego dowolnego przodka lub potomka).

Do zaprojektowania procedury zwracającej listę sąsiadów elementu, konieczne jest rozróżnienie rodzajów siatek, związanych z możliwymi strategiami dzielenia elementów. Jeśli przy podziałach nie dopuszcza się do powstania siatek nieregularnych (patrz p. 2.4.3), np. przez dokonywanie dodatkowych, często niejednorodnych, tymczasowych modyfikacji siatki[4], tworzenie list sąsiadów jest procedurą prostą.

Sytuacja znacznie komplikuje się w przypadku siatek nieregularnych. Nieregularność siatek może mieć różny stopień. Dla siatek jednonieregularnych każdy bok i każda krawędź dowolnego aktywnego elementu może posiadać co najwyżej jedno pokolenie potomków (czyli może mieć dzieci, ale nie może mieć wnucząt). Dzięki temu liczba aktywnych sąsiadów pojedynczego aktywnego elementu ograniczona jest do kilkunastu i może zostać zaprojektowana pojedyncza procedura zwracająca ich listę. W sytuacji ogólnej, kiedy siatki mogą być nieregularne w dowolnym stopniu, zwrócenie pełnej listy sąsiadów jest niewykonalne. Można wtedy zastosować konwencję, w której uwzględnia się jako sąsiadów tylko aktywne elementy zawierające boki rozważanego elementu lub przodków tych boków.

Powyższy przykład ilustruje istnienie w strukturze danych siatki elementów skończonych dwóch typów informacji. Informacje pierwszego typu mają charakter ustrukturalizowany (także dla siatek niestrukturalnych). Należą tu wszelkie dane o określonej liczebności składowych, np. lista boków elementu (trzy składowe dla trójkąta, pięć dla pryzmy), lista wierzchołków krawędzi (dwie składowe), lista dzieci elementu (zależnie od rodzaju elementu i typu podziału od dwóch do ośmiu składowych), lista równej wielkości sąsiadów elementu. Informacje drugiego typu odzwierciedlają dla siatek niestrukturalnych ich lokalnie nieprzewidywalny charakter, a także nieprzewidywalność przebiegu lokalnej adaptacji. Nie można z góry określić liczby elementów współdzielących pewną wspólną krawędź ani też liczby elementów (aktywnych i nieaktywnych) stykających się z określonym bokiem (aktywnym lub nieaktywnym).

Naturalnym rozwiązaniem przy projektowaniu struktur danych dla kodów MES jest wykorzystanie w maksymalnym stopniu danych ustrukturalizowanych, co prowadzi do prostszej postaci ostatecznego kodu. Zaletą przechowywania wyłącznie danych ustrukturalizowanych jest szybkość działania i prostota procedur dostępu do danych.

[4]Modyfikacje takie noszą zwyczajową nazwę „zagęszczania zielonego" (*green refinement*).

Wybór optymalnej struktury danych dla siatki MES nigdy jednak nie jest jednoznaczny i związany jest z zawsze istniejącym w przypadku projektowania dowolnych struktur danych (oraz związanych z nimi procedur) dylematem: przechowywać czy rekonstruować? Można teoretycznie wykazać [39], że do uzyskania dowolnej informacji o siatce, konieczne jest przechowywanie informacji tylko jednego rodzaju – identyfikatorów wierzchołków dla każdego elementu. Wszystkie pozostałe dane o siatce można otrzymać poprzez odpowiednie obliczenia. Dzięki temu dane o niektórych typach obiektów siatki (boki i krawędzie elementów) nie muszą być (i często nie są) przechowywane w strukturze danych. Bezpośrednie przechowywanie większej liczby danych usprawnia ich pozyskiwanie, prowadzi jednak do zwiększenia zajmowanej pamięci i zwiększenia złożoności procedur modyfikujących dane. W przypadku realizacji równoległej oznacza dodatkowo albo konieczność zwiększonego transferu przy redystrybucji struktury danych, albo skomplikowanie procedur redystrybucji, które muszą także uwzględniać rekonstrukcję danych.

Trudność zaprojektowania optymalnej struktury danych dla siatki MES jest potęgowana dodatkowo przez fakt, że w przypadku różnych typów dyskretyzacji i technik obliczeniowych różne są wymagania w stosunku do modułu siatki. Inne są dla aproksymacji liniowej, inne dla aproksymacji wyższych rzędów, inne dla adaptacji typu *hp*. Różne są wymagania dla solwerów z jednopoziomową poprawą uwarunkowania i dla wielopoziomowej poprawy. Inne są wymagania dla strategii adaptacji z rzadkimi zagęszczeniami, inne dla częstych zagęszczeń i rozgęszczeń. Optymalizacja struktury danych i procedur dostępu dla jednej grupy algorytmów nie musi oznaczać optymalności dla innej grupy [150].

W proponowanym modelu trudności te przesunięte są do fazy konkretnej realizacji modułu siatki. Ogólnie, ideą przewodnią prezentowanej koncepcji modułów MES jest istnienie wielu implementacji standardowego interfejsu, przeznaczonych i zoptymalizowanych dla różnych typów obliczeń. Dla każdej z nich dylemat: przechowywać czy rekonstruować może zostać rozstrzygnięty inaczej.

4.2.2 Moduł aproksymacji

Podstawowym obiektem modułu aproksymacji, stanowiącym podstawę do wyodrębnienia tego ostatniego z całości kodu MES, jest pole aproksymacji. W kontekście MES, w przeciwieństwie do klasycznych podejść różnicowych, dyskretny zbiór wartości określający pole nie zawsze związany jest z punktami w obszarze obliczeniowym. Stopnie swobody są współczynnikami kombinacji liniowej funkcji bazowych. Ponieważ funkcje bazowe zawsze można wiązać z

obiektami siatki, stopnie swobody naturalnie grupują się w zbiory przypisane poszczególnym obiektom siatki. Zbiór stopni swobody przyporządkowany pojedynczemu obiektowi ma określoną strukturę. Z pojedynczą funkcją bazową związany może być pojedynczy stopień swobody lub mały wektor stopni swobody w przypadku równań wektorowych. W trakcie symulacji potrzebnych może być kilka egzemplarzy zbioru stopni swobody, związanych z poszczególnymi chwilami czasowymi lub krokami iteracji solwera równań nieliniowych. Całkowita liczba stopni swobody związanych z pojedynczym obiektem siatki określana jest ostatecznie przez trzy parametry:

- liczbę funkcji bazowych związanych z obiektem,
- liczbę stopni swobody przypadających na pojedynczą funkcję bazową,
- liczbę egzemplarzy zbioru stopni swobody.

W przypadku rodzajów aproksymacji rozważanych w niniejszej pracy, zwłaszcza przy zastosowaniu adaptacji typu p i hp, korzystne jest oparcie schematu przechowywania wartości stopni swobody na pojęciu struktury stopni swobody.

Definicja 4.1 Strukturą stopni swobody *nazywana jest struktura danych związana z pojedynczym obiektem siatki, zawierająca zakodowane parametry określające liczbę stopni swobody przyporządkowanych obiektowi oraz odpowiadający całkowity zbiór stopni swobody.*

Kodowanie stopni aproksymacji związane jest z faktem, że dla niektórych typów aproksymacji stopień nie jest pojedynczą liczbą, ale może być np. określony odrębnie dla każdego kierunku przestrzennego.

Struktury stopni swobody określone są dla obiektów siatki o różnych rodzajach. Są one w sposób ścisły powiązane z funkcjami bazowymi, które tworzone są jako odpowiednie złożenia funkcji kształtu (patrz p. 2.5.2). Struktury stopni swobody mogą być podstawą konstrukcji uniwersalnego interfejsu dla modułu aproksymacji, możliwego do zastosowania w przypadku różnych typów problemów i dyskretyzacji [22].

4.2.3 Moduł interfejsu z solwerem równań liniowych

Tak jak dla interfejsu modułu aproksymacji podstawowym pojęciem może być struktura stopni swobody, tak dla interfejsu solwerów równań liniowych, opisywanych w niniejszej pracy, naturalnym pojęciem podstawowym jest elementarny podwektor wektora niewiadomych (patrz p. 2.7). Możliwe jest skonstruowanie interfejsu pomiędzy kodem MES i solwerem równań liniowych

(w proponowanej architekturze reprezentowanym przez adapter konkretnego solwera) wykorzystującego wyłącznie podwektory elementarne globalnych wektorów (niewiadomych i prawej strony układu równań liniowych) oraz odpowiadające im elementarne bloki macierzy układu równań liniowych.

Moduł interfejsu z solwerem równań liniowych dokonuje „tłumaczenia" danych z interfejsu z modułem aproksymacji, wyrażonych w terminach struktur stopni swobody, na dane w kategoriach podwektorów stopni swobody, na których oparty jest interfejs z solwerem równań liniowych. Aby umożliwić dowolne uporządkowanie równań i niewiadomych w solwerze liniowym, położenie dowolnego elementarnego podwektora w wektorze globalnym jest niezależne od numeracji struktur stopni swobody.

Zakłada się, że elementarne podwektory stopni swobody zajmują zwarte fragmenty globalnego wektora niewiadomych. Dzięki temu elementarny podwektor stopni swobody charakteryzowany jest tylko przez dwa parametry:

- liczbę pojedynczych stopni swobody,
- pozycję pierwszego stopnia swobody w globalnym wektorze niewiadomych.

Pozycja elementarnego podwektora stopni swobody w globalnym wektorze niewiadomych określa jednocześnie pozycję odpowiadającego mu podwektora w globalnym wektorze prawej strony. Pozycje pary elementarnych podwektorów stopni swobody określają miejsce odpowiadającego im elementarnego bloku w globalnej macierzy układu. Dlatego też przekazywanie do solwera macierzy elementowych i elementowych wektorów prawej strony może odbywać się wyłącznie przez wskazanie, jakie elementarne podwektory stopni swobody odpowiadają kolejnym blokom elementarnym tworzącym elementową macierz i kolejnym podwektorom tworzącym elementowy wektor prawej strony. Umożliwia to dokonywanie agregacji globalnej macierzy układu przez solwer równań liniowych (lub jego adapter) bez jakiejkolwiek wiedzy o rozwiązywanym problemie, użytej siatce i aproksymacji. Dzięki temu pojedynczy moduł solwera liniowego może być stosowany bez modyfikacji do rozwiązywania rozmaitych zagadnień.

W proponowanej architekturze kodów MES, w której moduł siatki nie zawiera informacji o siatkach różnych poziomów wykorzystywanych w solwerach wielosiatkowych, niezwykle ważnym zadaniem modułu interfejsu z solwerem liniowym jest dostarczenie solwerowi wielosiatkowemu wszelkich narzędzi koniecznych do realizacji obliczeń wielosiatkowych. Oznacza to, że moduł interfejsu z solwerem liniowym, na podstawie danych topologicznych i geometrycz-

nych o siatce oraz znajomości pól aproksymacji, musi dostarczyć solwerowi liniowemu macierze reprezentujące rozmaite operatory wykorzystywane w obliczeniach wielosiatkowych. Ponownie konstrukcja tych macierzy i ich przekazanie solwerowi mogą być w całości zrealizowane przy użyciu pojęć podwektorów elementarnych i elementarnych bloków macierzy [22].

4.2.4 Moduł zarządzania dekompozycją obszaru

Implementacja modułu dekompozycji obszaru polega na utworzeniu procedur realizujących: podział struktury danych całości programu na struktury przydzielone poszczególnym procesorom, klasyfikację obiektów występujących w tych strukturach i odpowiednie zarządzanie rozproszonymi danymi.

Podziału struktury danych dokonuje się na podstawie dekompozycji obszaru, przeprowadzanej w ramach proponowanej architektury przez programy zewnętrzne. Efektem podziału jest przypisanie numeru podobszaru (procesora) każdemu obiektowi siatki MES. Implikuje to przypisanie tego samego numeru związanej z obiektem strukturze stopni swobody. Struktura stopni swobody odpowiada podwektorowi stopni swobody, na którym operuje solwer równań liniowych. Podwektorowi odpowiada pasmo macierzy układu i podwektor wektora prawej strony. Wszystkie te obiekty dziedziczą numer podobszaru od początkowego obiektu siatki. Tym samym podział siatki MES implikuje dekompozycję całej struktury danych MES.

Do zapewnienia spójności struktur danych przechowywanych w lokalnych pamięciach procesorów musi zostać stworzony mechanizm stanowiący substytut globalnej przestrzeni adresowej w komputerach jednoprocesorowych. Możliwą realizacją tego mechanizmu (wykorzystaną w dalszych opisach implementacji oraz użytą w przykładach obliczeniowych) jest wyposażenie modułu dekompozycji w dodatkową strukturę danych. W ramach tej struktury każdemu obiektowi przypisuje się globalny międzyprocesorowy identyfikator MPID. Identyfikator ten składa się z zakodowanych: lokalnego identyfikatora (w ramach lokalnej struktury danych) i numeru podobszaru. MPID pozwala jednoznacznie zidentyfikować i zlokalizować każdy obiekt globalnej struktury danych programu MES.

4.2.5 Moduł interfejsu z bibliotekami przesyłania komunikatów

Procedury modułu dekompozycji nie wywołują procedur żadnej z konkretnych bibliotek przesyłania komunikatów, a tylko funkcje interfejsu, opatrzone na-

zwami według przyjętej w kodzie konwencji, realizujące konieczne operacje komunikacyjne. Mogą istnieć różne implementacje modułu interfejsu, wywołujące procedury różnych bibliotek przesyłania komunikatów. Tak więc można tworzyć implementacje modułu interfejsu komunikacyjnego nie tylko dla standardu MPI, ale także dla innych bibliotek, jak na przykład PVM.

Wszystkie przesłania komunikatów dzielą się na dwie grupy: przesłania dwupunktowe i operacje grupowe. W celu zmaksymalizowania ziarnistości obliczeń zakłada się, że przesłania dwupunktowe realizowane są z wykorzystaniem bufora pośredniego. Dane do przesłania gromadzone są w buforze, a następnie przesyłane, możliwie jak najrzadziej. Operacje grupowe wykorzystują standardowe typy wymiany komunikatów między wieloma procesorami (rozgłaszanie, rozpraszanie, zbieranie, itp.), w tym także redukcje (np. sumowania służące do obliczania norm, iloczynów skalarnych, itp.).

4.3 Równoległa realizacja obliczeń MES

Poniżej opisany jest przebieg obliczeń w ramach symulacji MES obejmujący etapy rozwiązania pojedynczego liniowego problemu dyskretyzacji przestrzennej oraz adaptacji siatki. Uwzględnione są także, konieczne do realizacji równoległej, etapy podziału obszaru obliczeniowego (siatki MES) i równoważenia obciążenia procesorów. Wymienione wyżej główne etapy opisane są w rozbiciu na drobniejsze fazy, umieszczone w porządku chronologicznym wykonania. Etapy i ich poszczególne fazy mogą być składane w bardziej złożone strategie rozwiązywania w przypadku problemów niestacjonarnych, nieliniowych, sprzężonych lub rozbudowanych strategii adaptacji.

Organizacja wszystkich faz obliczeń równoległych podlega temu samemu schematowi. Obliczenia dzieli się na dwie kategorie: obliczenia lokalne, wykonywane z użyciem wyłącznie danych lokalnych, oraz obliczenia globalne, w których biorą udział dane związane z całym obszarem obliczeniowym. Moduł dekompozycji zarządza przebiegiem obliczeń w taki sposób, aby moduły sekwencyjne realizowały obliczenia tak jak w programach sekwencyjnych, na przydzielonych sobie danych lokalnych. Koordynacją tych obliczeń w całości zajmuje się moduł dekompozycji. Jeżeli w obliczeniach pojawia się konieczność wykonania operacji globalnej, moduł sekwencyjny wykonuje fragment obliczeń związany z danymi odpowiadającymi obiektom wewnętrznym lub lokalnym i przekazuje sterowanie do modułu dekompozycji. Cała przedstawiona strategia realizacji obliczeń opiera się na takim doborze algorytmów i ich implementacji, aby procedury modułów sekwencyjnych realizując swe obliczenia nie

natrafiały nigdy na odwołania do obiektów zewnętrznych.

Podział obszaru obliczeniowego

Odpowiednia procedura modułu dekompozycji, wykorzystując zewnętrzny program dzielenia siatki, tworzy strukturę danych, w ramach której lokalnie przechowywane fragmenty struktur danych siatki i pól aproksymacji tworzą jedną globalną strukturę danych MES. Zgodnie z dokonanym podziałem na nienakładające się podobszary, każdemu obiektowi siatki i każdej strukturze stopni swobody przyporządkowany jest jednoznacznie procesor. W przypadku użycia identyfikatorów MPID, każdy z obiektów siatki i każda ze struktur stopni swobody otrzymuje swój jednoznaczny, globalny identyfikator.

Tworzenie nakładki

Tworzenie nakładki najczęściej w całości realizowane jest przez procedury modułu dekompozycji. Po ustaleniu rozmiaru strefy nakładania się podobszarów, odpowiednie fragmenty struktury danych przydzielane są poszczególnym podobszarom i przeprowadzana jest klasyfikacja obiektów siatki i struktur stopni swobody na wewnętrzne i nakładki. W przypadku korzystania z globalnych identyfikatorów MPID, struktura danych zapewnia tylko znajomość numeru procesora jednoznacznie przyporządkowanego obiektom siatki i strukturom stopni swobody. Konieczna dla wielu obliczeń informacja o lokalnie przechowywanych przez inne procesory kopiach danych (dotyczących nieprzyporządkowanych im obiektów siatki i struktur stopni swobody) musi być uzyskiwana poprzez rundę komunikacji międzyprocesorowej.

Moduł dekompozycji ustala rozmiar nakładki zgodnie z przyjętą strategią zrównoleglenia obliczeń. Może to być, korzystny w przypadku obliczeń z powolną komunikacją międzyprocesorową, rozmiar gwarantujący lokalne wykonywanie, istotnych z punktu widzenia wydajności, procedur całkowania numerycznego, mnożenia macierzy układu równań liniowych przez globalne wektory oraz wygładzania błędu. Taki rozmiar jest przyjęty jako podstawa dalszego opisu realizacji obliczeń.

Całkowanie numeryczne

W przyjętym modelu obliczeń całkowanie numeryczne nie poprzedza, ale jest fragmentem procesu rozwiązywania układu równań liniowych. Przy realizacji równoległej moduł dekompozycji pośredniczy w interakcjach między solwerem równań liniowych i resztą kodu MES. W trakcie tworzenia lokalnej, dla danego

podobszaru, struktury danych poziomów siatki moduł dekompozycji określa, dla których struktur stopni swobody tworzone będą pasma macierzy układu i po obszarach których obiektów siatki dokonywane będzie całkowanie. Następnie w trakcie tworzenia macierzy układu wywoływane są procedury modułów sekwencyjnych (modułu zależnego od problemu i modułu aproksymacji) dostarczające macierze elementowe. Procedury te działają identycznie jak w programach sekwencyjnych, dzięki lokalnej dostępności wszystkich potrzebnych danych.

Rozwiązanie układu równań liniowych

Globalne operacje na wektorach są realizowane przez odpowiednie procedury modułu dekompozycji, wykorzystujące standardowe procedury bibliotek przesyłania komunikatów, dostosowane do operacji wektorowych.

Mnożenie macierzy układu równań liniowych przez wektory globalne i rozwiązywanie problemów lokalnych w algorytmach wygładzania błędu realizowane jest lokalnie, dzięki odpowiedniemu rozmiarowi nakładki. Konieczna jest jednak, po obu tych operacjach, wymiana wartości, które nie zostały zmodyfikowane, między odpowiednimi procesorami.

Realizacja wymiany wartości w globalnych wektorach może być efektywnie dokonywana w module dekompozycji przy użyciu odpowiednich tablic przesłań. Tablice tworzy sie odrębnie dla każdego poziomu siatki uczestniczącego w obliczeniach. Dla każdego podobszaru tworzy się zestawy tablic odpowiadające sąsiadującym podobszarom. W pojedynczym zestawie znajdują się cztery (w uproszczonych przypadkach dwie) tablice przesłań. Każda z nich odpowiada jednej z grup stopni swobody opisanych w p.3.4.3, na s. 145, i zawiera identyfikatory podwektorów elementarnych należących do danej grupy.

Procedury rzutowania wektorów globalnych pomiędzy poziomami siatki są w pełni lokalne. Moduł dekompozycji wywołuje je w identyczny sposób jak przy realizacji sekwencyjnej, musi tylko zagwarantować aktualność danych używanych w projekcji. Podobnie jak w innych fazach obliczeń można to uzyskać albo przez odpowiednią wymianę danych między procesorami, albo przez lokalne wykonanie uaktualniających operacji.

Adaptacja

Operacje modyfikacji siatki przeprowadzane są przez procedury sekwencyjnego modułu siatki. Jedyną zmianą wymaganą przy realizacji równoległej jest zwrócenie do modułu dekompozycji listy wszystkich obiektów uczestniczących w po-

działach i złączeniach, razem z listami identyfikatorów obiektów utworzonych w trakcie podziału. Listy te pozwalają modułowi dekompozycji na przeprowadzenie rundy komunikacji, której celem jest uaktualnienie identyfikatorów MPID dla wszystkich obiektów zawartych w lokalnych strukturach danych.

Dodatkowe komplikacje pojawiają się w przypadku podziałów obiektów bezpośrednio na brzegach rozszerzonych podobszarów dekompozycji (por. p. 3.4.4). Na skutek lokalnych podziałów obiekty na takim brzegu mogą być różne dla zawierających je podobszarów. Moduł dekompozycji musi być wyposażony w mechanizmy utrzymywania spójności struktury danych w takich przypadkach.

Równoważenie obciążenia i transfer obiektów siatki

Pierwsze z tych zadań ponownie, tak jak pierwotny podział siatki, wykonywane jest przez moduł lub program, o którym zakłada się, że jest zewnętrzny w stosunku do rdzenia obliczeniowego MES. Na podstawie nowego podziału siatki moduł dekompozycji określa zakres koniecznych transferów pomiędzy podobszarami.

Transfer określany jest przez zbiór obiektów siatki (i związanych z nimi struktur stopni swobody), dla których ma nastąpić zmiana podobszaru, względem którego pozostają jako wewnętrzne. Dla uproszczenia dalszy opis przeprowadzony jest na przykładzie elementów. Transfer innych obiektów siatki i struktur stopni swobody dokonywany jest w analogiczny sposób.

Elementy, które mają zostać przydzielone innemu procesorowi tworzą zbiór zwany dalej łatą elementów. Ze względu na istnienie nakładki w podziale na podobszary, wzbogaca się pierwotną łatę elementów o warstwy elementów tworzące odpowiednią nakładkę łaty. Pierwotne elementy łaty stają się jej elementami wewnętrznymi.

Zgodnie z usytuowaniem elementów względem podobszaru wysyłającego i podobszaru przyjmującego dokonuje się dalszej klasyfikacji elementów. Działania podjęte w stosunku do konkretnych elementów zależą od ich ostatecznej klasyfikacji. Tablica 4.1 przedstawia kompletną klasyfikację elementów (odnosi się ona także do pozostałych obiektów siatki i struktur stopni swobody) łącznie z wyszczególnieniem operacji wykonywanych dla każdej z grup w trakcie transferu danych. „Powiadomienie” oznacza przesłanie komunikatu do procesora, względem którego obiekty siatki są wewnętrzne, o konieczności dokonania transferu.

Pełny transfer obiektów obejmuje wymazanie ich ze struktury danych nadawcy, wpisanie do struktury danych odbiorcy (wraz ze wszystkimi, zre-

Tablica 4.1. Klasyfikacja obiektów siatki przeprowadzana w celu dokonania transferu danych między procesorami

Położenie względem:			Podejmowane
łaty	nadawcy	odbiorcy	działania
wewnętrzne	wewnętrzne	zewnętrzne	transfer danych i zmiana MPID
wewnętrzne	wewnętrzne	w nakładce	zmiana MPID
w nakładce	wewnętrzne	zewnętrzne	transfer bez zmiany MPID
w nakładce	wewnętrzne	w nakładce	brak
w nakładce	w nakładce	wewnętrzne	brak
w nakładce	w nakładce	zewnętrzne	powiadomienie
w nakładce	w nakładce	w nakładce	powiadomienie

konstruowanymi spójnie i poprawnie, zależnościami rodzinnymi i sąsiedzkimi) oraz ustalenie prawidłowych identyfikatorów MPID. Dla niektórych z obiektów, zgodnie z tabl. 4.1, przeprowadza się tylko niektóre operacje. Jeśli przekazywany obiekt nie ulega wymazaniu po stronie nadawcy, konieczne jest przesłanie zwrotne jego nowego identyfikatora MPID. Podobnie jak przy równoległej adaptacji siatki, w przypadku obiektów na granicach między podobszarami i siatek nieregularnych pojawiają się dodatkowe kłopoty natury technicznej, które muszą zostać rozwiązane przez procedury modułu dekompozycji w celu zapewnienia poprawności i spójności globalnej struktury danych [21, 131].

4.4 Przykłady obliczeń

4.4.1 Symulacja zagadnienia konwekcji

Pierwszym przykładem numerycznym jest symulacja zagadnienia konwekcji. Zadanie to, proste z matematycznego punktu widzenia, zawiera jednak wszystkie trudności techniczne związane z równoległymi symulacjami problemów o dynamicznej adaptacji siatki.

Obliczenia w tym przykładzie dokonane zostały na sieci komputerów osobistych z procesorem Pentium 4, 1.6 GHz i pamięcią 1 GBajt każdy. Systemem operacyjnym był Linux, a siecią połączeń szybki Ethernet 100 Mbit/s.

Problem polega na unoszeniu wzdłuż osi y płaskiej fali o długości 200, w trójwymiarowym obszarze $[0, 38] \times [0, 1000] \times [0, 18]$. Użyta niestrukturalna siatka elementów pryzmatycznych jest adaptowana po każdym kroku oblicze-

niowym. W części obszaru, gdzie pojawia się fala, elementy są dzielone z ograniczeniem maksymalnej liczby generacji do trzech i założeniem jednonieregularności siatki. W momencie kiedy fala opuszcza fragment obszaru elementy są na powrót złączane.

W zagadnieniu tym jedynym interesującym procesem są kolejne, przeprowadzane równolegle, modyfikacje siatki i utrzymywanie zrównoważonego obciążenia procesorów. Lokalne zagęszczania i rozgęszczania siatki powodują, że w niektórych podobszarach elementów przybywa, a w innych ubywa. Zmiany te nie są gwałtowne ze względu na dokonywanie adaptacji po każdym kroku symulacji. Aby uniknąć zbyt częstych transferów obiektów siatki i realizować transfery odpowiednio większej liczby obiektów naraz, wybrana została strategia przywracania równowagi obciążenia wtedy, gdy ekstremalna liczba stopni swobody w pojedynczym podobszarze odbiega od średniej liczby stopni swobody we wszystkich podobszarach o pewien założony procent tej ostatniej. Przywracanie polega na wywołaniu procedur ponownego podziału siatki i dokonaniu, na podstawie uzyskanego nowego przypisania obiektów siatki podobszarom, odpowiedniego transferu danych.

Tablica 4.2 przedstawia charakterystyki modyfikacji siatki dla pięciu kolejnych kroków czasowych (od 100 do 105 – całość symulacji obejmowała 120 kroków czasowych) przykładowej symulacji z użyciem sieci czterech komputerów osobistych. Przyjęta wartość graniczna nierównowagi obciążenia wynosiła 10%. Dla umożliwienia lokalnych złączeń elementów transfer obiektów siatki dotyczył zawsze całych rodzin (przodków i potomków).

Wyniki w tabl. 4.2 pokazują, że całkowita liczba stopni swobody w trakcie symulacji pozostaje stała. Tyle samo obiektów powstaje, ile jest wymazywanych. Podziały dokonywane są jednak w innych podobszarach niż złączenia. Z każdym krokiem wzrasta nierównowaga liczby obiektów siatki i stopni swobody w poszczególnych podobszarach. Przeciętnie co trzy kroki czasowe dochodzi do przekroczenia wartości granicznej nierównowagi i transferu obiektów siatki oraz struktur stopni swobody. Liczby podane w tablicy dotyczą tylko obiektów siatki, dla których dokonywany był transfer całości struktury danych, nie zawierają liczb obiektów, dla których dokonywano jedynie uaktualnienia identyfikatorów MPID.

Przyspieszenie obliczeń uzyskane dla czterech komputerów (o stosunkowo powolnej sieci połączeń) i całości symulacji wyniosło 2.67. Daje to efektywność równą 67%. Zważywszy, że przyspieszenie to uwzględnia, poza zwykłym narzutem procedur całkowania numerycznego i solwera równań liniowych, także częste odwołania do procedur podziału siatki oraz transfery obiektów siatki,

Tablica 4.2. Charakterystyki przebiegu adaptacji i transferu obiektów siatki w trakcie wybranych kroków czasowych symulacji unoszenia fali płaskiej w prostopadłościennym obszarze trójwymiarowym

	Numer kroku czasowego				
	100	101	102	103	104
Średnia liczba stopni swobody	5086	5086	5086	5086	5086
Maksymalna liczba stopni swobody	5636	5120	5372	5596	5120
Minimalna liczba stopni swobody	4468	5012	4732	4508	4996
Liczba przesłanych wierzchołków	300	0	0	390	0
Liczba przesłanych krawędzi	1212	0	0	1671	0
Liczba przesłanych boków	1284	0	0	1863	0
Liczba przesłanych elementów	438	0	0	657	0

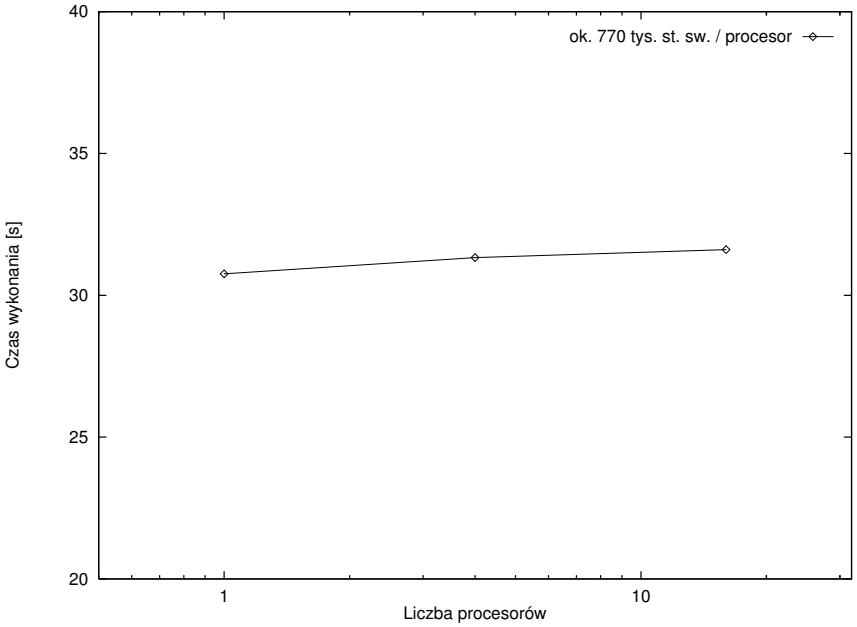
można uzyskany wynik uznać za zadowalający.

4.4.2 Aproksymacja równań Eulera – po raz ostatni

Kolejnym przykładem obliczeniowym jest (ponownie) symulacja nielepkiego przepływu o liczbie Macha 3 wokół profilu bi-NACA0012, omawianego w p. 3.6.1. Tym razem konkretne zadania dobrane są tak, aby zilustrować skalowalność obliczeń równoległych dla problemów wielkiej skali. Rozważana jest sekwencja kroków czasowych nieliniowej niejawnej metody Eulera. Parametry sterujące dobrane są tak, aby obliczenia dla każdego kroku zawierały dwie iteracje niedokładnej metody Newtona, a układ równań liniowych powstały w ramach każdej z iteracji (dla każdej z iteracji różny) rozwiązywany był za pomocą jednego restartu metody GMRES z 10 wektorami Kryłowa. Przytaczane czasy obliczeń obejmują więc łączny czas utworzenia i rozwiązania (niedokładnego) dwóch układów równań liniowych. Do porównania wybierany jest minimalny czas wykonania jednego z tak realizowanych kroków czasowych w przeciągu całej symulacji.

Maszyną równoległą dla tego i następnego przykładu był klaster komputerów osobistych z siecią typu Gigabit Ethernet i 12 węzłami obliczeniowymi. Każdy z węzłów wyposażony był w 2 procesory Pentium 4, 2.4 GHz, o pamięci 1 GBajt każdy.

Porównywane są trzy przypadki obliczeń. W pierwszym dokonywane były one na jednym procesorze. Liczba elementów wynosiła 384 256, liczba węzłów



Rys. 4.3. Skalowalność obliczeń dla symulacji przepływu naddźwiękowego wokół profilu bi-NACA0012: czas wykonania jednego kroku czasowego przy wzrastającej liczbie procesorów i wzrastającym rozmiarze zadania oraz założeniu stałego rozmiaru zadania na pojedynczym procesorze (czas obliczeń dla 16 procesorów uzyskany na podstawie czasu obliczeń dla 12 procesorów)

193 271, a liczba stopni swobody 773 084. Minimalny czas realizacji obliczeń dla pojedynczego kroku czasowego wyniósł 30.76 s. W drugim przypadku zadanie czterokrotnie większe (siatka uzyskana została przez jednorodne zagęszczanie siatki poprzedniej) rozwiązywane było na czterech procesorach. Czas obliczeń wyniósł 31.33 s. Dla przeprowadzenia obliczeń ostatniego przypadku ponownie dokonane zostało jednorodne zagęszczenie siatki. Ostateczna siatka miała 6 148 096 elementów, 3 078 623 węzły i 12 314 492 stopnie swobody. Ze względu na specyfikę klastra obliczenia wykonywane były na 12 procesorach. Najkrótszy czas realizacji obliczeń dla jednego kroku czasowego wyniósł 42.15 s.

W dwóch pierwszych przypadkach liczba stopni swobody w pojedynczym podobszarze waha się w okolicach 770 000, w ostatnim jest równa ok. 1 030 000. W celu uzyskania rezultatów ilustrujących skalowalność programu

wyniki ostatniego z przypadków obliczeń są modyfikowane. Właściwą do pokazania skalowalności liczbą procesorów jest 16. W przypadku zastosowania 12 procesorów liczba stopni swobody w pojedynczym podobszarze wzrasta w proporcji zbliżonej do 16/12 (1 030 000 / 770 000 $\approx$ 1.338) W podobnej proporcji (dokładnie w proporcji $(\frac{16}{12})^{1/2}$, jako że obliczenia są dwuwymiarowe) maleje wielkość komunikacji międzyprocesorowej. Dla przybliżonej ilustracji skalowalności obliczeń, przedstawionej na rys. 4.3, przyjęto liczbę procesorów równą 16 i czas obliczeń z wykorzystaniem 12 procesorów pomnożony przez 12/16 (dający w efekcie 31.61 s). Na podstawie powyższej analizy można uznać, że obliczenia wykazują dobrą, zbliżoną do liniowej skalowalność.

4.4.3 Nieciągła aproksymacja równania Laplace'a – po raz ostatni

Ostatnim przykładem obliczeń jest (ponownie) równoległe rozwiązanie zadania nieciągłej aproksymacji dla zagadnienia Laplace'a w sześcianie jednostkowym (patrz pp. 2.1.2, 2.6.6, 2.7.11 i 3.6.2). Tym razem celem jest rozwiązanie dużego zadania dla przetestowania skalowalności algorytmów i ich implementacji, a w szczególności braku „wąskich gardeł" (*bottlenecks*) w procesie rozwiązywania zagadnień wielkiej skali. Jest to test, zarówno użytych algorytmów, jak i architektury oraz szczegółów realizacji kodu.

Zastosowana siatka MES, o 6 258 688 elementach i 25 034 752 stopniach swobody, powstała przez kolejną adaptację siatki z przykładu w p. 3.6.2. Tak dużą liczbę elementów uzyskano dzięki równoległej adaptacji. Adaptacja sekwencyjna nie była możliwa, gdyż struktura danych MES zajmowała około 4.5 GBajta pamięci operacyjnej. Rozwiązanie zostało otrzymane po 15 iteracjach metody GMRES, z szybkością zbieżności równą 0.393, umożliwioną przez wielosiatkową poprawę uwarunkowania macierzy układu równań liniowych. Czas rozwiązania wyniósł 158 sekund.

Dodatek A

Oznaczenia

$[\,]$	–	operator skoku na brzegu międzyelementowym, s. 35,
$<>$	–	operator uśrednienia na brzegu międzyelementowym, s. 35,
α, β	–	parametry charakteryzujące różne warianty metody dyskretyzacji czasowej równań Eulera, s. 16,
$\mathbf{A}$	–	macierz układu równań liniowych MES, s. 43,
$\mathbf{A}_C$ $(\mathbf{A}_{C_i})$	–	macierz układu równań liniowych dla problemu zgrubnej korekty błędu (na siatce poziomu i), s. 73 (s. 77),
$\mathbf{A}^{\mathrm{E}}_{jk}$	–	blok elementarny macierzy układu równań liniowych MES, s. 45,
$\mathbf{A}^{\mathrm{P}}_{KM}$	–	blok poprawy uwarunkowania macierzy układu równań liniowych MES, s. 99,
$\boldsymbol{A}^{ij}$	–	macierzowe współczynniki przy pochodnych drugiego rzędu w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,
$a(.,.)$	–	forma dwuliniowa sformułowania MES, s. 70,
$a_{\mathrm{MES}}(.,.)$	–	forma dwuliniowa sformułowania ciągłego MES, s. 37,
$a_{\mathrm{NG}}(.,.)$	–	forma dwuliniowa sformułowania nieciągłego MES, s. 37,
$a_l(.,.)$	–	forma dwuliniowa odpowiadająca lokalnym problemom w trakcie iteracyjnego rozwiązywania układów równań liniowych, s. 71,

$\boldsymbol{B}^i$ – macierzowe współczynniki przy pochodnych pierwszego rzędu w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,
$\mathbf{b}$ – wektor prawej strony układu równań liniowych MES, s. 43,
$\boldsymbol{C}$ – macierzowy współczynnik przy wyrazie zerowego rzędu w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,
CFL – liczba *CFL*, s. 17,
CFL – parametr CFL, s. 19,
Δ – operator laplasjanu, $\Delta u = u_{,ii}$, s. 12,
δ_{ij} – delta Kroneckera, s. 12,
E_{p} – efektywność zrównoleglenia, s. 125,
$E_{\mathrm{p}}^{\mathrm{s}}$ – skalowana efektywność zrównoleglenia, s. 127,
$\tilde{\mathbf{e}}$ – błąd aktualnego rozwiązania w trakcie iteracyjnego rozwiązywania układów równań liniowych, s. 70,
$\tilde{\mathbf{e}}_C$ ($\tilde{\mathbf{e}}_{C_i}$) – błąd aktualnego rozwiązania zawężony do wektorowej przestrzeni zgrubnej w trakcie realizacji metody dwusiatkowej (cyklu V metody wielosiatkowej), s. 73 (s. 76),
$\tilde{e}$ – funkcja z przestrzeni skończenie elementowej V_h odpowiadająca błędowi aktualnego rozwiązania w trakcie iteracyjnego rozwiązywania układów równań liniowych, s. 71,
$\tilde{e}_l$ – rzut błędu $\tilde{e}$ na przestrzeń lokalną V_l, s. 71,
$\tilde{e}_C$ ($\tilde{e}_{C_i}$) – rzut błędu $\tilde{e}$ na przestrzeń zgrubną V_C (V_{C_i}), s. 72 (s. 76),
ϕ_l – funkcje bazowe MES, s. 41,
$\phi_{C_i}^l$ – funkcje bazowe MES na siatce zgrubnej poziomu i, s. 76,
$\hat{\phi}_l$ – funkcje kształtu, s. 39,
$\mathbf{F}$ – operator nieliniowy wynikły z zastosowania dyskretyzacji przestrzennej do stacjonarnego (lub zdyskretyzowanego w czasie) problemu nieliniowego, s. 23,
$\boldsymbol{f}_i^{\mathrm{E}}$ – wektor strumieni eulerowskich dla i-tego kierunku przestrzennego, s. 12,
$\boldsymbol{f}^{\mathrm{D}}$ – funkcja wektorowa w warunku brzegowym Dirichleta, s. 30,
$\boldsymbol{f}^{\mathrm{N}}$ – funkcja wektorowa w warunku brzegowym Neumanna, s. 30,
$\boldsymbol{f}^{\mathrm{R}}$ – funkcja wektorowa w warunku brzegowym Robina, s. 30,
Γ – brzeg obszaru obliczeniowego, s. 15, **30**,
Γ_{D} – część brzegu obszaru obliczeniowego z zadanym warunkiem brzegowym Dirichleta, s. 30,
Γ_{N} – część brzegu obszaru obliczeniowego z zadanym warunkiem brzegowym Neumanna, s. 30,

Γ_{R} – część brzegu obszaru obliczeniowego z zadanym warunkiem brzegowym Robina, s. 30,
Γ^{+} – część brzegu obszaru obliczeniowego, na której następuje wypływ, s. 30,
Γ^{-} – część brzegu obszaru obliczeniowego, na której następuje wpływ, s. 30,
Γ_e – brzeg elementu e, s. 32,
Γ_{ef} – wspólna część brzegu elementów e i f, s. 35,
Γ_{int} – brzeg międzyelementowy, suma brzegów Γ_{ef}, s. 35,
h – liniowy rozmiar elementu, s. 17,
$\mathbf{J}$ – macierz układu równań w metodzie Newtona, s. 23,
$\tilde{\mathbf{J}}$ – przybliżenie macierzy jakobianowej w metodzie Newtona dla równań Eulera, s. 26,
$\boldsymbol{J}_{\boldsymbol{T}_e}$ – macierz jakobianowa transformacji elementu e do przestrzeni odniesienia, s. 49,
$\boldsymbol{K}_{ij}$ – macierze sztucznej lepkości, s. 15,
$\boldsymbol{K}^{\mathrm{R}}$ – funkcja macierzowa w warunku brzegowym Robina, s. 30,
k^{ij} – współczynniki skalarnego równania dyfuzji, s. 57,
$l(.)$ – forma liniowa sformułowania MES, s. 70,
$l_{\mathrm{MES}}(.)$ – forma liniowa sformułowania ciągłego MES, s. 37,
$l_{\mathrm{NG}}(.)$ – forma liniowa sformułowania nieciągłego MES, s. 37,
$\boldsymbol{M}$ – macierzowy współczynnik przy pochodnej czasowej w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,
$\mathbf{M}^{-1}$ – liniowy operator implikowany przez algorytm poprawy uwarunkowania macierzy układu równań liniowych, s. 78,
N – liczba stopni swobody rozwiązania MES, liczba niewiadomych układu równań liniowych, s. 43,
N_{d} – szerokość diagonalnego pasma macierzy układu równań liniowych, s. 92,
N_u – liczba składowych wektorowej funkcji niewiadomej w zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,
N_{E} – liczba elementów w siatce MES, s. 57,
N_{I} – liczba punktów całkowania numerycznego po obszarze elementu odniesienia, s. 49,
N_{G} – liczba geometrycznych funkcji kształtu określających geometrię elementu, s. 50,
N_{K} – liczba funkcji kształtu użytych do aproksymacji rozwiązania w elemencie, s. 52,

N_{W} – średnia liczba operacji potrzebnych do obliczenia współczynnika w pojedynczym punkcie całkowania numerycznego, s. 54,

N_{C} – liczba całek składających się na pojedynczy wyraz macierzy elementowej, s. 54,

N_{bl} – liczba podwektorów wektora niewiadomych używanych w algorytmach metod Schwarza, s. 68,

$N_{\mathrm{bl}}^{\mathrm{E}}$ – liczba podwektorów elementarnych wektora niewiadomych, s. 44,

$N_{\mathrm{bl}}^{\mathrm{P}}$ – liczba podwektorów poprawy uwarunkowania używanych w algorytmach wygładzania błędu, s. 99,

N_{bl}^{K} – liczba podwektorów elementarnych składających się na K-ty podwektor poprawy uwarunkowania, s. 99,

N_{ssb}^{i} – liczba stopni swobody w i-tym podwektorze elementarnym, s. 99,

N_{ssB}^{K} – liczba stopni swobody w K-tym podwektorze poprawy uwarunkowania, s. 99,

$N_{\mathrm{sbl,off}}^{i}$ – liczba podwektorów elementarnych sąsiadujących z i-tym podwektorem elementarnym i nie należących wraz z nim do wspólnego podwektora poprawy uwarunkowania, s. 100,

$N_{\mathrm{sbl},<i}^{i}$ – liczba podwektorów elementarnych sąsiadujących z i-tym podwektorem elementarnym i mających indeksy mniejsze niż i, s. 103,

N_{k} – maksymalna liczba iteracji w pojedynczym restarcie metody GMRES, s. 80,

N_{p} – liczba procesorów realizujących obliczenia równoległe, s. 124,

$\boldsymbol{n}$ – wektor jednostkowy, normalny zewnętrznie względem brzegu obszaru obliczeniowego, s. 15, 35;
także – wektor jednostkowy, normalny do brzegu międzyelementowego, s. 35,

$\boldsymbol{n}_e$ – wektor jednostkowy, normalny zewnętrznie względem brzegu elementu e, s. 34,

Ω – obszar obliczeniowy, $\Omega \in \mathbb{R}^2$ lub $\Omega \in \mathbb{R}^2$, s. 15, **30**,

Ω_e – obszar elementu e, s. 32,

$\hat{\Omega}_e$ – obszar elementu odniesienia dla elementu e, s. 32,

Ω_l – podobszar obszaru obliczeniowego, s. 68,

$\mathbf{P}_l$ – operator rzutowania na lokalną przestrzeń wektorową związaną z podobszarem Ω_l, s. 70,

p – stopień wielomianu aproksymującego, s. 39,

p_e – indeks określający stopień aproksymacji w elemencie e, s. 39,

$\boldsymbol{q}^i$ – wektory strumieni w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,

$\mathbf{R}_l$ – operator zawężenia do przestrzeni wektorowej związanej z przestrzenią lokalną V_l, s. 69,

$\mathbf{R}_l^T$ – operator rozszerzenia z przestrzeni wektorowej związanej z przestrzenią lokalną V_l, s. 69,

$\mathbf{R}_C$ ($\mathbf{R}_{C_i}$) – operator zawężenia do przestrzeni wektorowej związanej z przestrzenią zgrubną V_C (V_{C_i}), s. 73 (s. 74, 76),

$\mathbf{R}_C^T$ ($\mathbf{R}_{C_i}^T$) – operator rozszerzenia z przestrzeni wektorowej związanej z przestrzenią zgrubną V_C (V_{C_i}), s. 73 (s. 74, 76),

$\mathbf{r}_k$ – residuum dla k-tej iteracji metody sprzężonych gradientów, s. 79,

$\hat{\mathbf{r}}_k$, $\bar{\mathbf{r}}_k$ – k-ty wektor bazowy podprzestrzeni Kryłowa w metodzie GMRES (przed normalizacją i po normalizacji), s. 80,

S_{p_e} – przestrzeń funkcji wielomianowych dla elementu odniesienia, s. 30,

S_p – przyspieszenie obliczeń, s. 125,

S_p^s – skalowane przyspieszenie obliczeń, s. 127,

$\boldsymbol{s}$ – wektor źródła w wektorowym zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,

$\boldsymbol{T}_e$ – transformacja elementu e do przestrzeni odniesienia, s. 32,

T_s – czas wykonania najlepszego algorytmu sekwencyjnego dla danego problemu, s. 125,

T_p – czas wykonania obliczeń programem równoległym dla danego problemu, s. 125,

t – chwila czasu, s. 12,

t_n – n-ta dyskretna chwila czasu, s. 16,

Δt – długość n-tego kroku czasowego, $\Delta t = t_{n+1} - t_n$, s. 16,

Δt_loc – lokalny krok czasowy, s. 19,

u – skalarna funkcja niewiadoma, s. 12, 57,

$\boldsymbol{u}$ – wektor zmiennych zachowawczych dla równań Eulera, s. 12; także – wektorowa funkcja niewiadoma w zagadnieniu konwekcji–dyfuzji–reakcji, s. 30,

$\boldsymbol{u}^n$ – rozwiązanie problemu zależnego od czasu w chwili t_n, s. 16,

$\bar{\boldsymbol{u}}$ – funkcja dyskretyzowana „pod prąd” na brzegu międzyelementowym, s. 36,

$\mathbf{u}$ – globalny wektor niewiadomych dyskretyzacji MES (wektor stopni swobody), s. 23, **43**,

$\mathbf{u}^*$ – rozwiązanie dokładne układu równań liniowych, s. 70,
$\mathbf{u}^m$ – rozwiązanie po m-tej iteracji algorytmu rozwiązywania układów równań liniowych, s. 68,
$\tilde{\mathbf{u}}$ – aktualne rozwiązanie w trakcie iteracyjnego rozwiązywania układów równań liniowych, s. 68,
$\tilde{\mathbf{u}}_{C_i}$ – aktualne rozwiązanie na siatce poziomu i w trakcie realizacji cyklu V metody wielosiatkowej, s. 76,
$\mathbf{u}_k^{\mathrm{E}}$ – podwektor elementarny wektora niewiadomych $\mathbf{u}$, s. 44,
V_h – przestrzeń skończenie elementowa funkcji skalarnych, s. 39,
$\boldsymbol{V}_h$ – przestrzeń skończenie elementowa funkcji wektorowych, s. 15, **31**,
V_l – lokalna przestrzeń skończenie elementowa funkcji określonych w podobszarze Ω_l obszaru obliczeniowego, s. 70,
V_C (V_{C_i}) – (zgrubna) przestrzeń skończenie elementowa związana z siatką zgrubną (poziomu i), s. 72 (s. 73, 73, 75),
v^I – waga I-tego punktu kwadratury całkowania numerycznego, s. 49,
W – praca systemu komputerowego, s. 126,
$W_{\mathrm{izo}}(N_{\mathrm{p}})$ – funkcja izoefektywności obliczeń równoległych, s. 128,
w – skalarna funkcja testująca, s. 57,
$\boldsymbol{w}$ – wektorowa funkcja testująca, s. 15,
$\boldsymbol{x}$ – wektor współrzędnych punktu w przestrzeni fizycznej, s. 12,
$\boldsymbol{\xi}$ – wektor współrzędnych punktu w przestrzeni odniesienia, s. 32,
$\boldsymbol{\xi}^I$ – współrzędne I-tego punktu kwadratury całkowania numerycznego, s. 49.

Literatura

[1] M. Ainsworth, J.T. Oden: A unified approach to a posteriori error estimation using element residual methods. *Numerische Mathematik*, 65, s. 23–50, 1993.

[2] S.K. Aliabadi, S.E. Ray, T.E. Tezduyar: SUPG finite element computation of viscous compressible flows based on the conservation and entropy variables formulations. *Computational Mechanics*, 11, s. 300–312, 1993.

[3] G.M. Amdahl: Validity of the single processor approach to achieving large scale computing capabilities. W: *AFIPS Conference Proceedings*, s. 483–485, 1967.

[4] R. Amstrong, D. Gannon, A. Geist, K. Kahey, S. Kohn, L. McInnes, S. Parker, B. Smolinski: Toward a common component architecture for high-performance scientific computing. Preprint P759-0699, Argonne National Laboratory, 1999. Strona internetowa: `http://z.ca.sandia.gov/cca-forum`.

[5] E. Arge, A.M. Bruaset, P.B. Calvin, J.F. Kanney, H.P. Langtangen, C.T. Miller: On the numerical efficiency of C++ in scientific computing. W: M. Daehlen, A. Tveito, red., *Numerical Methods and Software Tools in Industrial Mathematics*. Birkhauser Press, 1997.

[6] E. Arge, A.M. Bruaset, H.P. Langtangen, red.: *Modern software tools for scientific computing*. Birkhauser Press, 1997.

[7] Accelerated Strategic Computing Initiative, a program of the U. S. Department of Energy. Strona internetowa: `http://www.sandia.gov/ASCI/`.

[8] I. Babuŝka: Courant element: before and after. Technical Note BN-1154, Institute for Physical Science and Technology, University of Maryland, 1993.

[9] I. Babuŝka, W.C. Rheinbolt: Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15, s. 736–754, 1978.

[10] I. Babuŝka, W.C. Rheinbolt: A posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12, s. 1597–1615, 1978.

[11] I. Babuŝka, M. Suri: The optimal convergence rate of the *hp*-version of the finite element method. *SIAM Journal on Numerical Analysis*, 24, s. 750–776, 1987.

[12] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith: Efficient management of parallelism in object-oriented numerical software libraries. W: E. Arge, A.M. Bruaset, H.P. Langtangen, red., *Modern software tools for scientific computing*, s. 163–201, Birkhauser Press, 1997.

[13] K. Banaś: Algorytmy niejawnego całkowania po czasie dla metody stabilizowanych elementów skończonych. W: *Materiały XIII Krajowej Konferencji Mechaniki Płynów*, Vol. II, s. 225–230, 1998.

[14] K. Banaś: Parallel Newton-Krylov-Schwarz solver for nonlinear finite element simulations. W: W. Łakota et al., red., *Proceedings of the XIVth Polish Conference on Computer Methods in Mechanics*, s. 31–32, 1999.

[15] K. Banaś: Parallel PDE simulations based on domain decomposition. W: R. Wyrzykowski et al., red., *Proceedings of the Third International Conference on Parallel Processing and Applied Mathematics*, s. 445–454, 1999.

[16] K. Banaś: 3D *h*-adaptive finite element simulations of inviscid and viscous flows. *Mechanika Teoretyczna i Stosowana*, 35, s. 527–540, 1997.

[17] K. Banaś: A parallel adaptive code for compressible Navier-Stokes simulations. *TASK Quarterly*, 3, s. 17–37, 1999.

[18] K. Banaś: Convergence to steady-state solutions for stabilized finite element simulations of compressible flows. *Computers and Mathematics with Applications*, 40, s. 625–643, 2000.

[19] K. Banaś: A Newton-Krylov solver with multiplicative Schwarz preconditioning for finite element compressible flow simulations. *Communications in Numerical Methods in Engineering*, 18, s. 269–275, 2002.

[20] K. Banaś: A model for parallel adaptive finite element software. W: R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O.Widlund, J. Xu, red., *Domain Decomposition Methods in Science and Engineering*, *Lecture Notes in Computational Science and Engineering*. Vol. 40, s. 159––166, Springer, 2004.

[21] K. Banaś: A modular design for parallel adaptive finite element computational kernels. W: M. Bubak, G.D. van Albada, P.M.A. Sloot, J.J. Dongarra, red., *Computational Science – ICCS 2004, 4th International Conference, Proceedings, Part II*, *Lecture Notes in Computer Science*, Vol. 3037, s. 155–162, Springer, 2004.

[22] K. Banaś: On a modular architecture for finite element systems. I. Sequential codes. *Computing and Visualization in Science*, 2004.

[23] K. Banaś: Parallelization of large scale adaptive finite element computations. W: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Waśniewski, red., *Parallel Processing and Applied Mathematics, Proceedings of Vth International Conference, PPAM 2003*, *Lecture Notes in Computer Science*, Vol. 3019, s. 431–438. Springer, 2004.

[24] K. Banaś, L. Demkowicz: 3D *h-p* adaptive package. Report 4, Section of Applied Mathematics UCK, Cracow University of Technology, Warszawska 24, 31-155 Kraków, Poland, 1993.

[25] K. Banaś, L. Demkowicz: Entropy stable gas dynamics simulations by adaptive finite elements. W: S. Wagner, E.H. Hirchel, J. Periaux, R. Piva, red., *Proceedings of the Second European Fluid Dynamics Conference ECCOMAS 94*, Vol. II, s. 97–104, Wiley, 1994.

[26] K. Banaś, L. Demkowicz: Entropy controlled adaptive finite element simulations for compressible gas flow. *Journal of Computational Physics*, 126, s. 181–201, 1996.

[27] K. Banaś, L. Demkowicz: New quasi-natural artificial viscosity models for compressible fluid flow, with improved entropy production mechanism. *Mechanika Teoretyczna i Stosowana*, 35, s. 233–248, 1997.

[28] K. Banaś, J. Płażek: Parallel h-adaptive simulations of inviscid flows by the finite element method. *Mechanika Teoretyczna i Stosowana*, 35, s. 249–262, 1997.

[29] K. Banaś, M.F. Wheeler: Preconditioning GMRES for discontinuous Galerkin approximations. *Computer Assisted Mechanics and Engineering Sciences*, 11, s. 47–62, 2004.

[30] K. Banaś, J. Płażek: Parallel iterative solvers for the finite element method. W: A. Garstecki, J. Rakowski, red., *Proceedings of the XIIIth Polish Conference on Computer Methods in Mechanics*, Vol. I, s. 115––120, 1997.

[31] K. Banaś, J. Płażek: Dynamic load balancing for the preconditioned GMRES solver in a parallel, adaptive finite element Euler code. W: J.-A. Désidéri, C. Hirsch, P. Le Tallec, M. Pandolfi, J. Périaux, red., *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, s. 1025–1031, Wiley, 1996

[32] K. Banaś, B. Wrana: Porównanie jednokrokowych metod całkowania nieliniowych równań ruchu. W: J. Orkisz et al., red., *Materiały IX Konferencji Metody Komputerowe w Mechanice*, s. 31–38, 1989.

[33] W. Bangerth: Using modern features of C++ for adaptive finite element methods: Dimension–independent programming in deal.II. W: *Proceedings of the IMACS 2000 World Congress*, 2000.

[34] W. Bangerth, G. Kanschat: Concepts for object-oriented finite element software – the deal.II library. Preprint SFB 359, Universitat Heidelberg, 1999.

[35] B. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.van der Vorst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.

[36] P. Bastian: Load balancing for adaptive multigrid methods. *SIAM Journal on Scientific Computing*, 19(4), s. 1303–1321, 1998.

[37] P. Bastian, K. Birken, K. Johannsen, S. Lang, N. Neuss, H. Rentz-Reichert, C. Wieners. UG – a flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1(1),

s. 27–40, 1997. Strona internetowa: `http://cox.iwr.uni-heidelberg.de/ug`.

[38] P. Bastian, V. Reichenberger: Multigrid for higher order discontinuous Galerkin finite elements applied to groundwater flow. Preprint 2000-37, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, University of Heidelberg, 2000.

[39] M.W. Beall, M.S. Shephard: A general topology-based mesh data structure. *International Journal for Numerical Methods in Engineering*, 40, s. 1573–1596, 1997.

[40] M.W. Beall, M.S. Shephard: An object-oriented framework for reliable numerical simulations. *Engineering with Computers*, 15, s. 61–72, 1999.

[41] R. Beck, B. Erdman, R. Roitzsch: An object-oriented adaptive finite element code: design issues and application in hyperthermia treatment planning. W: E. Arge, A.M. Bruaset, H.P. Langtangen, red., *Modern software tools for scientific computing*, s. 105–123, Birkhauser Press, 1997.

[42] E.B. Becker, G.F. Carey, J.T. Oden: *Finite Elements. An Introduction.* Prentice Hall, 1981.

[43] T. Belytschko, T.J.R. Hughes, red.: *Computational methods for transient analysis.* Elsevier, 1983.

[44] K. Birken, R. Rühle: Dynamic Distributed Data: efficient, portable and easy to use. W: E. D'Hollander, G.R. Joubert, F.J. Peters, D. Trystram, red., *Parallel Computing: State-of-the-Art and Perspectives.* Elsevier, 1996.

[45] J.H. Bramble: *Multigrid Methods.* Longman Scientic & Technical, 1993. Pitman Research Notes in Mathematics Series #294.

[46] J.J. Burton, L.R. Nackman: *Scientific and Engineering C++.* Addison-Wesley, 1994.

[47] R. Buyya: *High Performance Cluster Computing.* Prentice Hall, 1999.

[48] X.C. Cai, D. Gropp, E. Keyes, G. Melvin, P. Young: Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. Report TR 96-39, ICASE, 1996.

[49] X.C. Cai, M. Sarkis: A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 21, s. 792–797, 1999.

[50] X.C. Cai, W.D. Gropp, D.E. Keyes: A comparison of some domain decomposition and *ILU* preconditioned iterative methods for nonsymmetric elliptic problems. *Journal of Numerical Linear Algebra with Applications*, 1(5), s. 477–504, 1994.

[51] G.F. Carey, A. Bose, B. Davis, C. Harle, R. McLay: Parallel computation of viscous flows. W: A. Tentner, red., *Proceedings of the 1997 Simulation Multiconference. High Performance Computing '97*, s. 34–40, SCS, 1997.

[52] P.G. Ciarlet: *The Finite Element Method for Elliptic Problems*. North--Holland, 1978.

[53] P. Ciarlet Jr, F. Lemour, B.F. Smith: On the influence of the partitioning schemes on the efficiency of overlapping domain decomposition methods. Technical report CAM 94-23, UCLA, 1994.

[54] R.L. Clay, K.D. Mish, I.J. Otero, L.M. Taylor, A.B. Williams: An Annotated Reference Guide to the Finite Element Interface (FEI) Specification. Sandia Report SAND99-8229, Sandia National Laboratories, 1999.

[55] R.W. Clough: The finite element method in plane stress analysis. W: *Proceedings of the 2nd ASCE Conference on Electronic Computation*, 1960.

[56] B. Cockburn, G. Karniadakis, C. Shu, red.: *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Vol. 11, *Lecture Notes in Computational Science and Engineering*. Springer, 2000.

[57] T.H. Cormen, C.E. Leiserson, R.L. Rivest: *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[58] R. Courant: Variational methods for the solution of problems of equilibrium and vibration. *Bulletin of the American Mathematical Society*, 49, s. 1–23, 1943.

[59] Strona domowa „15th International Conference on Domain Decomposition Methods". Strona internetowa: `http://www.mi.fu-berlin.de/dd15/index.php`.

[60] L. Demkowicz, J.T. Oden, W. Rachowicz, O. Hardy: Towards a universal *hp* adaptive finite element strategy, Part.1 Constrained approximation and data structure. *Computer Methods in Applied Mechanics and Engineering*, 77, s. 79–112, 1989.

[61] L. Demkowicz, J.T. Oden, W. Rachowicz, O. Hardy: An *h-p* Taylor--Galerkin finite element method for compressible Euler equations. *Computer Methods in Applied Mechanics and Engineering*, 88, s. 363–396, 1991.

[62] L. Demkowicz, W. Rachowicz, P. Devloo: A fully automatic *hp*--adaptivity. *Journal of Scientific Computing*, 17, s. 117–142, 2002.

[63] L. Demkowicz, A. Safjan, K. Banaś: Stabilized FE methods for linear acoustics. W: M. Hafez, J.C. Heinrich, red., *Proceedings of the Xth International Conference on Finite Elements in Fluids*, s. 138–143, 1998.

[64] A. Dervieux, B.v. Leer, J. Periaux, A. Rizzi, red.: *Numerical simulation of compressible Euler flows*. Vieweg, Braunschweig, 1989.

[65] J. Donea: A Taylor-Galerkin method for convective transport problems. *International Journal for Numerical Methods in Engineering*, 20, s. 101––119, 1984.

[66] M. Dryja: A method of domain decomposition for 3-D finite element problems. W: R. Glowinski, G. H. Golub, G. A. Meurant, J. Périaux, red., *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, s. 43–61, SIAM, 1988.

[67] M. Dryja, O.B. Widlund: Towards a unified theory of domain decomposition algorithms. W: T. Chan et al., red., *IIIrd International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, 1990.

[68] M. Dryja, O.B. Widlund: Some recent results on Schwarz type domain decomposition algorithms. W: A. Quarteroni, Y. A. Kuznetsov, J. Périaux, O. B. Widlund, red., *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, Vol. 157, *Contemporary Mathematics*, s. 53–62, AMS, 1994.

[69] M. Dubiner: Spectral methods on triangles and other domains. *Journal of Scientific Computing*, 6(4), s. 345–390, 1991.

[70] S.C. Eisenstat, H.F. Walker: Globally convergent inexact Newton methods. *SIAM Journal on Optimization*, 4, s. 393–422, 1994.

[71] S.C. Eisenstat, H.F. Walker: Choosing the forcing terms in inexact Newton methods. *SIAM Journal on Scientific Computing*, 17, s. 16–32, 1996.

[72] *Engineering with Computers*, 15, 1999.

[73] K. Eriksson, D. Estep, P. Hansbo, C. Johnson: *Computational Differential Equations*. Cambridge University Press, 1996.

[74] C. Farhat: A simple and effective automatic FEM decomposer. *Computers & Structures*, 28, s. 579–602, 1988.

[75] J.S.R.A. Filho, P.R.B. Devloo: Object-oriented programming in scientific computations: the beginning of a new era. *Engineering with Computers*, 8, s. 81–87, 1991.

[76] Finite element method universal resource. Strona internetowa: `http://femur.wpi.edu/`.

[77] I. Foster, C. Kesselman: *The Grid: blueprint for a new computing infrastructure*. Morgan Kaufmann, San Francisco, 1998.

[78] G. Golub, J.M. Ortega: *Scientific Computing with Introduction to Parallel Computing*. Academic Press, San Diego, 1993.

[79] *Grand Challenges: High Performance Computing and Communications. The FY 1992 U.S. Research and Development Program. Supplement to the President's Fiscal Year 1992 Budget. p.7.*

[80] M. Grochowski, R. Schaefer, P. Uhruski: Diffusion based scheduling in the agent-oriented computing system. W: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Waśniewski, red., *Parallel Processing and Applied Mathematics, Proceedings of Vth International Conference, PPAM 2003*, *Lecture Notes in Computer Science*, Vol. 3019, s. 431–438. Springer, 2004.

[81] W. Gropp, E. Lusk, N. Doss, A. Skjellum: A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6), s. 789–828, September 1996.

[82] W.D. Gropp, D.E. Keyes, L.C. McInnes, M.D. Tidriri: Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. *International Journal of High Performance Computing Applications*, 14, s. 102–136, 2000.

[83] J. Gustafson, G. Montry, R. Benner: Development of parallel methods for a 1024-processor hypercube. *SIAM Journal on Scientific and Statistical Computing*, 9(4), s. 609–638, 1988.

[84] W. Hackbusch: *Multigrid methods and applications*. Springer, 1985.

[85] W. Hackbusch: *Iterative solution of large sparse systems of linear equations*. Springer, 1994.

[86] P. Hansbo: Explicit Streamline Diffusion Finite Element Methods for the Compressible Euler Equations in Conservation Variables. *Journal of Computational Physics*, 109, s. 274–288, 1993.

[87] P. Hansbo, C. Johnson: Adaptive streamline diffusion method for compressible flow using conservation variables. *Computer Methods in Applied Mechanics and Engineering*, 87, s. 267–280, 1991.

[88] B. Hendrickson, K. Devine: Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 184, s. 485–500, 2000.

[89] M.R. Hestenes, E.L. Stiefel: Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49, s. 409–436, 1952.

[90] C. Hirsch: *Numerical Computation of Internal and External Flows*. Wiley, 1988.

[91] T.J.R. Hughes: *Finite element method – Linear Static and Dynamic Finite Element Analysis*. Prentice Hall, 1987.

[92] T.J.R. Hughes, G. Engel, L. Mazzei, M.G. Larson: A comparison of discontinuous and continuous Galerkin methods based on error estimates, conservation, robustness and efficiency. W: *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Vol. 11, *Lecture Notes in Computational Science and Engineering*, s. 135–146, Springer, 2000.

[93] T.J.R. Hughes, G.M. Hulbert: Space-time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering*, 66(3), s. 339–363, 1988.

[94] Z. Johan, T.J.R. Hughes: A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids. *Computer Methods in Applied Mechanics and Engineering*, 87, s. 281–304, 1991.

[95] C. Johnson: Discontinuous Galerkin finite element methods for second order hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 107, s. 145–157, 1993.

[96] T. Jurczyk, B. Glut, J. Kitowski: An empirical comparison of decomposition algorithms for complex finite element meshes. W: *Parallel Processing and Applied Mathematics. 4th International Conference, PPAM 2001. Revised Papers*, Vol. 2328, *Lecture Notes in Computer Science*, s. 493–501, Springer, 2002.

[97] Y. Kallinderis: Adaptive hybrid prismatic-tetrahedral grids. *International Journal for Numerical Methods in Fluids*, 20, s. 1023–1037, 1995.

[98] C.T. Kelley, D.E. Keyes: Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35, s. 508–523, 1998.

[99] D. Kershaw: The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics*, 26, s. 43–65, 1978.

[100] M. Kleiber: *Incremental Finite Element Modelling in Non-Linear Solid Mechanics*. Wiley, 1989

[101] M. Kleiber, red.: *Handbook of Computational Solid Mechanics. Survey and Comparison of Contemporary Methods*. Springer, 1998

[102] D.A. Knoll, D.E. Keyes: Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193, s. 357–397, 2004.

[103] J. Krok, P. Leżański, J. Orkisz, P. Przybylski, R. Schaefer: Basic concepts of an open distributed system for cooperative design and structure analysis. *Computer Assisted Mechanics and Engineering Sciences*, 3, s. 169–186, 1996.

[104] J. Kucwaj: The algorithm of adaptation by using graded mesh generator. *Computer Assisted Mechanics and Engineering Sciences*, 7, s. 615–624, 2000.

[105] J. Kucwaj, K. Banaś: Algorytm generowania siatek hybrydowych w wielospójnych obszarach płaskich. W: *Materiały XIII Krajowej Konferencji Mechaniki Płynów*, Vol. II, s. 301–306, 1998.

[106] V. Kumar, A. Grama, A. Gupta, G. Karypis: *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjaming-Cummings, 1994.

[107] H.P. Langtangen: *Computational Partial Differential Equations, Numerical Methods and Diffpack Programming.* Springer, 1999.

[108] H.P. Langtangen, A.M. Bruaset, E. Quak, red.: *Advances in software tools for scientific computing.* Springer, 2000.

[109] P.D. Lax, B. Wendroff: Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13, s. 217–237, 1960.

[110] P. LeTallec: *Domain decomposition method in computational mechanics.* Computational Mechanics Advances. North Holland, 1994.

[111] LINPACK. Strona internetowa: `http://www.netlib.org/linpack/`.

[112] R. Lohner: Moore's law and the diminishing importance of parallel computing. *IACM expressions. The bulletin for the International Association for Computational Mechanics*, (13), s. 6–8, February 2003. Strona internetowa: `http://www.cimne.upc.es/iacm`.

[113] G. R. Luecke, W.H. Lin: Scalability and performance of OpenMP and MPI on a 128-processor SGI Origin 2000. *Concurrency and Computation: Practice and Experience*, 13(10), s. 905–928, 2001.

[114] R.I. Mackie: Object oriented programming of the finite element method. *International Journal for Numerical Methods in Engineering*, 35, s. 425––436, 1992.

[115] R.I. Mackie: An object-oriented approach to calculation control in finite element programs. *Computers and Structures*, 77, s. 461–474, 2000.

[116] H.A. Mang, F.G. Rammerstorfer, J. Eberhardsteiner, red.: *Proceedings of the Fifth World Congress on Computational Mechanics*. Vienna University of Technology, CDROM, 2002. Strona internetowa: `http://wccm.tuwien.ac.at`.

[117] V. Mann, M. Parashar: Engineering an interoperable computational collaboratory on the Grid. *Concurrency and Computation. Practice and Experience.*, 14, s. 1569–1593, 2002.

[118] S.F. McCormick, red.: *Multigrid Methods*. Frontiers in Applied Mathematics, Vol. 6. SIAM, 1987.

[119] J.M. Melenk, K. Gerdes, C. Schwab: Fully discrete *hp*-finite elements I: fast quadrature. *Computer Methods in Applied Mechanics and Engineering*, 190, s. 4339–4364, 2001.

[120] Message passing interface forum. Strona internetowa: `www.mpi-forum.org`.

[121] K. Nakajima: OpenMP/MPI hybrid vs. flat MPI on the Earth Simulator: Parallel iterative solvers for finite element method. GeoFEM Report 2003-007, RIST/TOKYO, 2003. Strona internetowa: `http://geofem.tokyo.rist.or.jp/report_en/2003_007.html`.

[122] G. Nelissen, P.F. Vankeirsbilck: Electrochemical modelling and software genericity. W: E. Arge, A.M. Bruaset, H.P Langtangen, red., *Modern software tools for scientific computing*, s. 81–103, Birkhauser Press, 1997.

[123] Object oriented numerics. Strona internetowa: `http://www.oonumerics.org/oon/`.

[124] J.T. Oden: Some historic comments on finite elements. W: *Proceedings of the ACM conference on History of scientific and numeric computation*, s. 125–130, ACM Press, 1987.

[125] J.T. Oden, I. Babuŝka, C.E. Baumann: A discontinuous *hp* finite element method for diffusion problems. *Journal of Computational Physics*, 146, s. 491–519, 1998.

[126] J.T. Oden, T. Belytschko, I. Babuŝka, T.J.R. Hughes: Research directions in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 192, s. 913–922, 2003.

[127] J.T. Oden, A. Patra: A parallel adaptive strategy for *hp* finite element computations. *Computer Methods in Applied Mechanics and Engineering*, 121, 1995.

[128] T. Olas, K. Karczewski, A. Tomas, R. Wyrzykowski: FEM computations on clusters using different models of parallel programming. W: *Parallel Processing and Applied Mathematics. 4th International Conference, PPAM 2001. Revised Papers*, Vol. 2328, *Lecture Notes in Computer Science*, s. 170–182, Springer, 2002.

[129] SIAM Working Group on CSE Education. Graduate education in Computational Science and Engineering. *SIAM Review*, 43, s. 163–177, 2001.

[130] OpenMP. Strona internetowa: `http://www.openmp.org`.

[131] C. Özturan, H.L. deCougny, M.S. Shephard, J.E. Flaherty: Parallel adaptive mesh refinement and redistribution on distributed memory computers. *Computer Methods in Applied Mechanics and Engineering*, 119, s. 123–137, 1994.

[132] *Parallel Computing*, 23(9), 1997.

[133] D.L. Parnas: On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), s. 1053–1058, 1972.

[134] A. Patra, J. Long, A. Laszloffy: Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive *hp* finite element simulations. *Parallel Computing*, 26, s. 1765–1788, 2000.

[135] J. Płażek, K. Banaś, J. Kitowski: Efficiency comparison of explicit and implicit parallel programming for a FEM problem on HP Exemplar systems. W: M. Bubak, J. Mościński, red., *Proceedings of the International Conference on High Performance Computing on Hewlett-Packard Systems*, s. 181–188, 1997.

[136] J. Płażek, K. Banaś, J. Kitowski: Comparison of implicit and explicit parallel programming models for a finite element simulation algorithm. W: B. Kagstrom, J. Dongarra, E. Elmroth, J. Wasniewski, red., *Proceedings of the 4th International Workshop on Applied Parallel Computing, PARA'98*, Vol. 1541, *Lecture Notes in Computer Science*, s. 433–437, Springer, 1998.

[137] J. Płażek, K. Banaś, J. Kitowski: Finite element message-passing/DSM simulation algorithm for parallel computers. W: P. Sloot, M. Bubak, B. Hertzberger, red., *Proceedings of the International Conference on High Performance Computing and Networking*, Vol. 1401, *Lecture Notes in Computer Science*, s. 878–880, Springer, 1998.

[138] J. Płażek, K. Banaś, J. Kitowski: Implementation issues of computational fluid dynamics algorithms on parallel computers. W: *Recent Advances in Parallel Virtual Machine and Message Passing Interface. 6th European PVM/MPI User's Group Meeting. Proceedings*, Vol. 1697, *Lecture Notes in Computer Science*, s. 349–355, Springer, 1999.

[139] J. Płażek, K. Banaś, J. Kitowski: Scalable CFD computations using message-passing and distributed shared memory algorithms. W: *Recent Advances in Parallel Virtual Machine and Message Passing Interface. 7th European PVM/MPI User's Group Meeting. Proceedings*, Vol. 1908, *Lecture Notes in Computer Science*, s. 282–288, Springer, 2000.

[140] J. Płażek, K. Banaś, J. Kitowski: Comparison of message passing and shared memory implementations of the GMRES method on MIMD computers. *Scientific Programming*, 9, s. 195–209, 2001.

[141] J. Płażek, K. Banaś, J. Kitowski, K. Boryczko: Exploiting two-level parallelism in FEM applications. W: B. Hertzberger, P. Sloot, red., *Proceedings of the International Conference on High Performance Computing and Networking*, Vol. 1225, *Lecture Notes in Computer Science*, s. 878–880, Springer, 1997.

[142] S. Prudhomme, J.T. Oden, T. Westermann, J. Bass, M.E. Botkin: Practical methods for a posteriori error estimation in engineering applications. *International Journal for Numerical Methods in Engineering*, 56, s. 1193––1224, 2003.

[143] A. Quarteroni, A. Valli: *Numerical Approximation of Partial Differential Equations*. Springer, 1994.

[144] W. Rachowicz: An anisotropic h-type mesh refinement strategy. *Computer Methods in Applied Mechanics and Engineering*, 109, s. 169–181, 1993.

[145] W. Rachowicz: An overlapping domain decomposition preconditioner for an anisotropic *h*-adaptive finite element method. *Computer Methods in Applied Mechanics and Engineering*, 127, s. 269–292, 1995.

[146] W. Rachowicz: *Adaptacyjna metoda elementów skończonych do rozwiązywania równań Naviera-Stokesa dla przepływów ściśliwych.* Instytut Podstawowych Problemów Techniki PAN, Warszawa, 1997.

[147] W. Rachowicz, L. Demkowicz: An *hp*-adaptive finite element method for electromagnetics – part I: Data structure and constrained approximation. *Computer Methods in Applied Mechanics and Engineering*, 187, s. 307––337, 2000.

[148] W. Rachowicz, L. Demkowicz: An *hp*-adaptive finite element method for electromagnetics – part II: A 3D implementation. *International Journal for Numerical Methods in Engineering*, 53, s. 147–180, 2002.

[149] E.S. Raymond: *The Cathedral and the Bazaar.* O'Reilly, 2001.

[150] J.-F. Remacle, B.K. Karamete, M.S. Shephard: Algorithm Oriented Mesh Database. Report 5, SCOREC, 2000.

[151] J.-F. Remacle, O. Klaas, J.E. Flaherty, M.S. Shephard: A Parallel Algorithm Oriented Mesh Database. *Engineering with Computers*, 18, s. 274–284, 2002.

[152] B. Rivière, M.F. Wheeler, V. Girault: Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. Part I. *Computational Geosciences*, 3, s. 337–360, 1999.

[153] J. Rumbaugh, I. Jacobson, G. Booch: *The Unified Modeling Language Reference Manual.* Addison-Wesley, 1998.

[154] Y. Saad: *Iterative methods for sparse linear systems.* PWS Publishing, 1996.

[155] Y. Saad, M. Schultz: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7, s. 856–869, 1986.

[156] A. Safjan, J.T. Oden: High-order Taylor-Galerkin methods for linear hyperbolic systems. *Journal of Computational Physics*, 120, s. 206–230, 1995.

[157] K. Schloegel, G. Karypis, V. Kumar: Graph partitioning for high performance scientific simulations. W: J. Dongarra, I. Foster, G. Fox, K. Kennedy, A. White, L. Torczon, W. Gropp, red., *Sourcebook of Parallel Computing*. Morgan Kaufmann, 2002.

[158] D. Schotzau, C. Schwab: Time discretization of parabolic problems by the *hp*-version of the discontinuous Galerkin finite element method. *SIAM Journal on Numerical Analysis*, 38(3), s. 837–875, 2000.

[159] H.A. Schwarz: Über Einige Abbildungsaufgaben. *Journal für Die Reine und Angewandte Mathematik*, 70, s. 105–120, 1869.

[160] F. Shakib, T.J.R. Hughes, Z. Johan: A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 89, s. 141–219, 1991.

[161] S.J. Sherwin, G.E. Karniadakis: Tetrahedral *hp* finite elements: algorithms and flow simulations. *Journal of Computational Physics*, 124, s. 14–45, 1996.

[162] B. Smith, P. Bjorstad, W. Gropp: *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equation*. Cambridge University Press, Cambridge, 1996.

[163] B.F. Smith: An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM Journal on Scientific and Statistical Computing*, 13(1), s. 364–378, 1992.

[164] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra: *MPI: The Complete Reference*. MIT Press, 1996.

[165] Standard Performance Evaluation Corporation. Strona internetowa: `http://www.spec.org`.

[166] T. Sterling, P. Messina, P.H. Smith: *Enabling Technologies for Petaflops Computing (Scientific and Engineering Computation)*. MIT Press, 1995.

[167] J.R. Stewart, W.R. Witkowski, K.D. Copps, H.C. Edwards, J.D. Zepper: Advanced technologies for parallel adaptive multi-physics simulation. *Proceedings of the Fifth World Congress on Computational Mechanics*. Vienna University of Technology, CDROM, 2002. Strona internetowa: `http://wccm.tuwien.ac.at`.

[168] J. Stoer, R. Bulirsch, red.: *Introduction to Numerical Analysis*, Vol. 12, *Text in Applied Mathematics*. Springer, 1991.

[169] C. Szyperski: *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 1999.

[170] P. Tauzowski, M. Kleiber: Sensitivity analysis for viscoelastic bodies in object-oriented finite element environment. *Computer Assisted Mechanics and Engineering Sciences*, 10(2), s. 223–238, 2003.

[171] Top500. Strona internetowa: `http://www.top500.org`.

[172] A.B. Tucker, red.: *The computer science and engineering handbook*. CRC Press, 1996.

[173] W.W. Tworzydło, J.T. Oden, E.A. Thornton: Adaptive implicit/explicit finite element method for compressible viscous flow. *Computer Methods in Applied Mechanics and Engineering*, 95, s. 397–440, 1992.

[174] A.J. van der Steen, J.J. Dongarra: Overview of recent supercomputers. Technical report, 2003. Strona internetowa: `http://www.phys.uu.nl/~steen/web03/overview.html`.

[175] T.L. Veldhuizen: Arrays in blitz++. W: *Proceedings of the 2nd International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'98)*, Lecture Notes in Computer Science. Springer, 1998. Strona internetowa: `http://www.oonumerics.org/blitz`.

[176] P. Wesseling: *An Introduction to Multigrid Methods*. Wiley, 1992.

[177] S.O. Wille, O. Staff, A.F.D. Loula: Block and full matrix ILU preconditioners for parallel finite element solvers. *Computer Methods in Applied Mechanics and Engineering*, 191, s. 1381–1394, 2001.

[178] K. Wilson: Grand Challenges to Computational Science. *Future Generation Computer Systems*, 5, s. 171, 1989.

[179] R. Winkelmann, J. Häuser, R.D. Williams: Strategies for parallel and numerical scalability of CFD codes. *Computer Methods in Applied Mechanics and Engineering*, 174, s. 433–456, 1999.

[180] G. Wittum: On the robustness of ILU smoothing. *SIAM Journal on Scientific and Statistical Computing*, 10, s. 699–717, 1989.

[181] R. Wyrzykowski, T. Olas, N. Sczygiol: Parallel finite element modeling of solidification processes. W: *Parallel Computation. 4th International ACPC Conference. Proceedings*, Vol. 1557, *Lecture Notes in Computer Science*, s. 183–195, Springer, 1999.

[182] R. Wyrzykowski, T. Olas, N. Sczygiol: Object-oriented approach to finite element modeling on clusters. W: *Applied Parallel Computing. 5th International Workshop, PARA 2000. Proceedings*, Vol. 1947, *Lecture Notes in Computer Science*, s. 250–257. Springer, 2001.

[183] O.C. Zienkiewicz: *Metoda elementów skończonych.* Arkady, 1972.

[184] O.C. Zienkiewicz, R.L. Taylor: *Finite element method. Vol 1-3.* Butterworth Heinemann, 2000.

[185] O.C. Zienkiewicz, J.Z. Zhu: Adaptivity and mesh generation. *International Journal for Numerical Methods in Engineering*, 32, s. 783–810, 1991.

[186] T. Zimmermann, Y. Dubois-Pelerin, P. Bomme: Object-oriented finite element programming: I. Governing Principles. *Computer Methods in Applied Mechanics and Engineering*, 98, s. 291–303, 1992.