Paweł WIELEBA
Jan SIKORA

OBIEKTOWA BIBLIOTEKA NUMERYCZNA METODY ELEMENTÓW BRZEGOWYCH

STRESZCZENIE *Artykuł przedstawia obiektową bibliotekę numeryczną implementującą Metodę Elementów Brzegowych (MEB). W pierwszym rzędzie projekt implementowany jest dla środowisk pracujących pod kontrolą systemów operacyjnych Linux/Unix. Biblioteka jest implementowana w języku C++. Jednak jej projekt i architektura umożliwiają implementację również w innych językach obiektowych zachowując API (tłum. z ang. Programowalny Interfejs Aplikacji). Zastosowanie UML (tłum. z ang. Ujednolicony Język Modelowania) wraz z obiektowym projektowaniem znacząco ułatwia proces tworzenia specyfikacji, projektowania, implementacji i testowania. Implementacja algorytmu MEB nie jest zwykłym odwzorowaniem. Użycie zaawansowanych, lecz upowszechnionych wśród inżynierów otwartego oprogramowania, narzędzi do zarządzania kodem źródłowym, wersjonowaniem, tworzeniem wydań, czy wspomaganiem procesu konfiguracji, kompilacji, budowy i instalacji, znacząco ułatwia, a w wielu wypadkach umożliwia tworzenie i rozwój biblioteki numerycznej MEB. Jednak użycie narzędzi programistycznych nie jest warunkiem wystarczającym, lecz tylko koniecznym. Występują znaczące obszary MEB, które muszą być specjalnie traktowane. Na początku wypada wspomnieć o dwóch: osobliwość wynikająca z metody mająca swoje odwzorowanie we*

Słowa kluczowe: *MEB, biblioteka obiektowa, biblioteka numeryczna, GNU, GPL, model wieloobszarowy*

---

mgr inż. Paweł WIELEBA
e-mail: p.wieleba@iem.pw.edu.pl

prof. dr hab. Jan SIKORA
e-mail: j.sikora@iel.waw.pl

Zakład Metrologii i Badań Nieniszczących
Instytut Elektrotechniki

PRACE INSTYTUTU ELEKTROTECHNIKI, zeszyt 239, 2008

współczynniku c(r), obliczanym dla każdego równania MEB, oraz osobliwość pojawiająca się podczas całkowania numerycznego elementów brzegowych. Kolejne komplikacje pojawiają się podczas obliczeń problemów wieloobszarowych, np. gdy problem jest nieliniowy, ale można go opisać kilkoma liniowymi obszarami. Wówczas złożoność obliczeniowa, jak i zapotrzebowanie na pamięć rośnie, więc musi być skompensowane, aby implementacja MEB, w opisywanym przypadku biblioteka, była konkurencyjna porównując do innych nielicznych implementacji MEB, ale również konkurencyjna do innych metod rozwiązywania równań różniczkowych cząstkowych, zwłaszcza Metody Elementów Skończonych (MES). Wybór zaawansowanych narzędzi programistycznych o ugruntowanej pozycji w środowisku otwartego oprogramowania ma również duże znaczenie dla użytkownika, ponieważ nie wymaga nauczenia się nowego interfejsu, jak i metod konfiguracji, kompilacji, budowania i instalacji. Użytkownik może od razu skupić się na wykorzystaniu oprogramowania do rozwiązywania zadań i problemów. Z punktu widzenia użytkownika bardzo ważne są porty i pakiety źródłowe, ponieważ znacząco ułatwiają i przyspieszają instalację biblioteki w systemach operacyjnych, wraz z późniejszymi aktualizacjami. Potwierdzenie zasadności tworzenia tego typu produktu zostało opisane w artykule. Ponadto załączone są podstawy MEB wyszczególniając te, które wpływają na architekturę i kontrolę przepływu informacji procesu. Opisane są również przykłady wykorzystania biblioteki, zarówno łatwe, przeznaczone dla początkujących użytkowników MEB i biblioteki, jak i zaawansowane pokazujące możliwości przedstawionej biblioteki MEB. Opisywana biblioteka przeznaczona jest zarówno do dydaktyki, dla naukowców jak i do wykorzystania w przemyśle.

1. WSTĘP

W ciągu ostatnich trzech dekad można zaobserwować szybki rozwój metody elementów brzegowych (MEB) [1]. MEB znajduje zastosowanie m.in. w takich dziedzinach jak: elektrotechnika, mechanika, energetyka lotnictwa, akustyka, medycyna [2]. Wraz ze zwiększaniem się mocy obliczeniowej komputerów metody numeryczne elementów brzegowych, czy skończonych z powodzeniem znajdują zastosowanie w rozwiązywaniu coraz trudniejszych problemów [3], zachowując przy tym akceptowalny poziom błędu. Spośród wymienionych metod numerycznych, szerzej stosowana jest Metoda Elementów Skończonych (MES). Istnieje wiele pakietów oprogramowania zarówno komercyjnego, bezpłatnego jak i otwartego implementującego MES.

MES z powodzeniem jest stosowane w przemyśle, nauce i dydaktyce do modelowania problemów fizycznych [7]. Natomiast Metoda Elementów Brzegowych, pomimo swoich zalet, co w przypadku wielu problemów przekłada

się na jej wyższość w porównaniu z MES nie doczekała się tak szerokiej implementacji. W tabeli 1 znajduje się zestawienie dostępnych implementacji MEB o otwartym kodzie źródłowym.

TABELA 1
Oprogramowanie otwarte MEB

Biblioteka (programy)	**Środowisko (język)**	**Licencjonowanie (dystrybucja)**	**Zastosowanie**
ABEM [10] (Kirkup)	Fortran	komercyjny, otwarty	akustyka, równania Laplace'a i Helmholza
LibBem [11]	C++	pół-komercyjny	równanie Laplace'a
BEMLIB [12] (Pozrikidis)	Fortran	GPL	równania Laplace'a, Helmholtza, Stokes flow
BIEPACK [13]	Fortran	darmowy, otwarty	równanie Laplace'a

W tabeli 2 znajdują się implementacje komercyjne.

TABELA 2
Oprogramowanie komercyjne MEB

Biblioteka (programy)	**Środowisko (język)**	**Zastosowanie**
BEASY [14]	Windows or Unix binaries	inżynieria konstrukcji
Integrated Engineering Software [15]	Windows only	pola, fale, analiza termiczna
GPBEST [16]	Windows or Unix	akustyka, analiza termiczna
Concept Analysis [17]	Windows	analiza naprężeń

Należy tutaj nadmienić, że nie istnieje uniwersalna biblioteka MEB o otwartym kodzie źródłowym spełniająca wymagania postawione przez biblioteki systemowe w środowiskach systemów operacyjnych Unix/Linux, czy standardy wytyczone przez środowisko otwartego oprogramowania GNU. Implementacje o zamkniętym kodzie źródłowym, z natury rzeczy nie pozwalają na dogłębną analizę ich struktury. Natomiast wskazane otwarte implementacje nie posiadają jednolitego API, pozwalającego rozwiązywać różne problemy opisywane przez różne równania.

Większość implementacji MEB z tabeli 1 zostało zaprogramowanych w języku Fortran, który obecnie poza nielicznymi wyjątkami, nie jest wykorzystywany przy pisaniu oprogramowania użytkowego i systemowego. W naturalny sposób powoduje to ogromne ograniczenie populacji, która ma możliwości rozwijania oprogramowania, czy wprowadzania poprawek. Obecnie proces dydaktyczny nie uwzględnia nauki języka Fortran, ponieważ w ogólności jest on nieprzydatny na obecnym rynku pracy. W latach siedemdziesiątych i osiemdziesiątych Fortran był szeroko stosowany w obliczeniach numerycznych na superkomputerach [8]. Wówczas wypracowany kod jest również wykorzystywany obecnie. W latach dziewięćdziesiątych powstał nowy standard języka Fortran95 oraz w roku 2003 nowe rozszerzenia, umożliwiające częściowo podejście obiektowe, zostały ustandaryzowane. Jednak dynamika rozwoju tego języka maleje i znacznie mniej osób swobodnie się nim posługuje. Jest to jedna z przyczyn, dla których część wymienionych implementacji z tabeli nr 2 nie jest rozwijanych, a pozostałe nie mają rozbudowanej wokół nich społeczności.

Stworzenie uniwersalnej biblioteki MEB o obiektowej budowie, otwartym kodzie źródłowym i wykonanej w technologii bibliotek systemowych znanych ze środowisk Unix/Linux, a niedostępnej dzisiaj na rynku jest zasadne. Dzięki temu jest tworzona platforma przeznaczona do implementacji obecnych i przyszłych zastosowań MEB.

2. METODA ELEMENTÓW BRZEGOWYCH

Termin Metoda Elementów Brzegowych został zaproponowany w latach siedemdziesiątych przez Carlosa A. Brebbia [5] i obowiązuje do dziś opisując metodę numeryczną opisywaną w tym artykule i implementowaną przez tytułową bibliotekę numeryczną. Głównym rywalem dla MEB jest MES. MES jest szerzej stosowany, pomimo tego, że w wielu wypadkach lepszym wyborem do rozwiązania zadanego problemu byłoby użycie MEB. Jednak w niektórych wypadkach użycie MEB jest nieefektywne [4]. Tabela 3 pokazuje zalety i wady obydwu metod.

Obydwie metody MEB i MES służą do rozwiązywania równań różniczkowych cząstkowych. Jednak MEB wymaga aby równanie było sformułowane w postaci całkowej BIE (ang. Boundary Integral Equation), natomiast MES rozwiązuje równania w postaci różniczkowej PDE (ang. Partial Differential Equation). Równanie (1) przedstawia równanie całkowe BIE, używane przez MEB.

TABELA 3
Wady i zalety MEB

Zalety MEB	Wady MEB
● Mniejsza liczba równań w porównaniu do metod obszarowych ● Dyskretyzowany jest tylko brzeg (Γ) obszaru (Ω) ● Obliczanie wartości potencjału *u* lub jego pochodnej *q* w dowolnym węźle wewnątrz obszaru (Ω): ○ Nie wymaga tworzenia nowej siatki modelu ○ Błąd w dowolnym węźle wewnętrznym zależy od błędów dyskretyzacji, aproksymacji i wyznaczonych wartości wszystkich elementów brzegowych ○ Obliczenia mogą być wykonane tylko w wybranych punktach, a nie w całym obszarze	● Macierz **A** liniowego układu równań jest pełna i niesymetryczna, charakteryzuje się gorszym współczynnikiem uwarunkowania niż w MES ● Całkowanie osobliwe jest skomplikowane [2,3] ● Elementy brzegowe muszą być jednoznacznie skierowane (wektor normalny **n** skierowana do zewnątrz) ● Dla postawionego problemu wymagane jest rozwiązanie fundamentalne (funkcja Greena) ● Traci swoje zalety dla problemów charakteryzujących się: ○ niekorzystną geometrią ○ nieliniowym lub anizotropowym środowiskiem ● Obszar musi być zdyskretyzowany dla równań Laplace'a czy Helmholtza

$$c_i u_i \quad + \quad \int_{\Gamma} u \frac{\partial G}{\partial n} d\Gamma \quad = \quad \int_{\Gamma} qG d\Gamma \quad + \quad \int_{\Omega} fG d\Omega \qquad (1)$$

W równaniu (1) zastosowano następujące oznaczenia: c_i – współczynnik równania całkowego, u – potencjał, $q = \frac{\partial u}{\partial n}$, G – funkcja Greena, f – funkcja obszarowa, Γ – brzeg, Ω – obszar.

Równanie (1) jest rozwiązywane przez tytułową bibliotekę, jednak aby można je było rozwiązać numerycznie, należy je przekształcić do postaci macierzowej.

$$A_u u \quad = \quad B_q q \quad + \quad F \qquad (2)$$

Kolejnym krokiem w procesie działania algorytmu MEB jest nałożenie warunków ciągłości na interfejsie (dotyczy tylko modeli wieloobszarowych). Następnie należy wprowadzić do równania warunki brzegowe zadane dla rozwiązywanego problemu. Po przeprowadzeniu tych działań równanie (2) przekształca się do postaci:

$$\mathbf{A}\mathbf{x}_{\mathbf{uq}} = B \tag{3}$$

W kolejnym etapie liniowy układ równań (3), jest rozwiązywany wybranym algorytmem. Można zastosować np. dekompozycję LU czy jeden z algorytmów iteracyjnych GMRES.

Po rozwiązaniu równania macierzowego (3), wartości potencjału $\mathbf{u}$ i pochodnej $\mathbf{q}$ we wszystkich węzłach na brzegu Γ są znane. Etap ten kończy obliczenia, w przypadku gdy zadanie określało jedynie znalezienie rozkładu potencjału i pochodnej na brzegu obszaru.

W przypadku gdy niezbędna jest znajomość wartości potencjału w określonych węzłach obszaru należy wykonać dodatkowe obliczenia według wzoru z równania (4). Na tym etapie wymagane jest całkowanie wszystkich elementów brzegowych dla każdego węzła wewnętrznego, jednak nie jest to całkowanie osobliwe, pod warunkiem, że węzeł wewnętrzny nie znajduje się bardzo blisko elementu brzegowego.

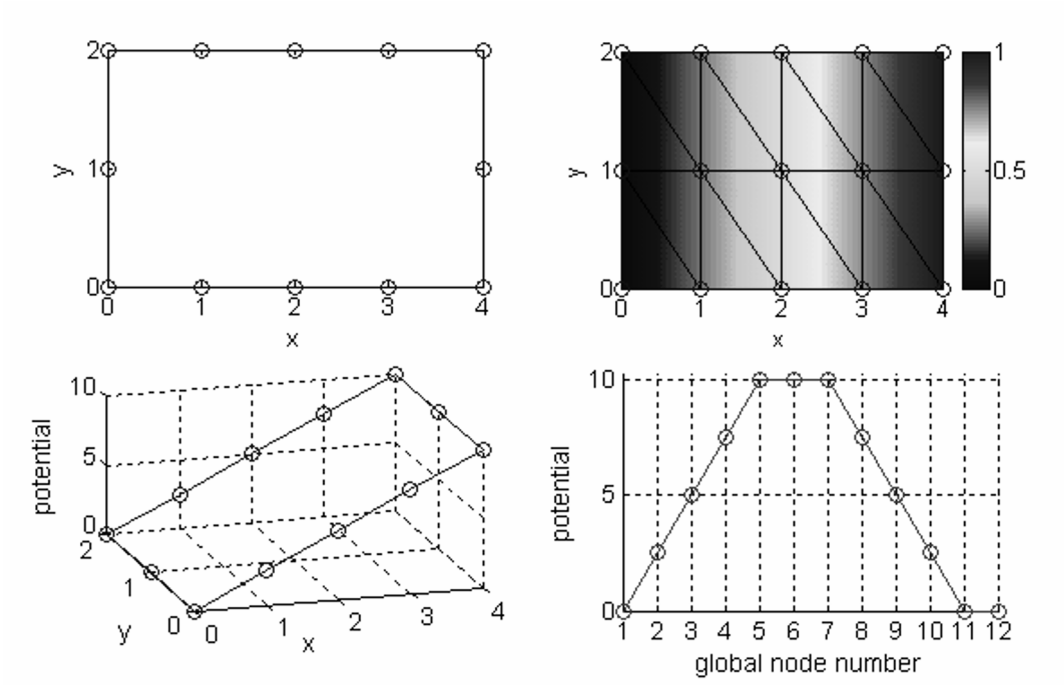
$$u_i = -\int_{\Gamma} u \frac{\partial G}{\partial n} d\Gamma + \int_{\Gamma} qG d\Gamma + \int_{\Omega} fG d\Omega \tag{4}$$

Wyznaczenie powyższego równania kończy proces obliczeń MEB.

3. ZASTOSOWANIE

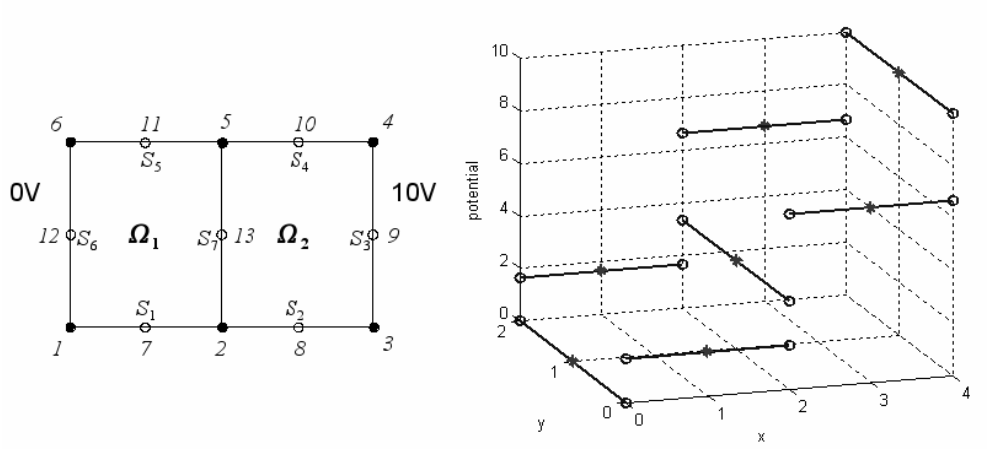
Architektura opisywanej biblioteki, pozwala na rozwiązywanie różnych problemów metodą MEB. Aby umożliwić komunikację między modelem np. kondensator z rysunku 1, a biblioteką została stworzona referencyjna aplikacja, dzięki której można wprowadzić dane i wykonać obliczenia. Na potrzeby aplikacji i biblioteki utworzony został format danych wejściowych i wyjściowych. Format ten oparty jest na tekstowym pliku, zgodnym z gramatyką opisującą macierze w M-plikach używanych przez pakiet MATLAB i jest całkowicie z nim zgodny. Dane wprowadzane są w postaci macierzy o określonych nazwach. Dzięki temu wyniki można poddać dalszej obróbce przy użyciu pakietu MATLAB bez żadnej konwersji. Przykładem może być wizualizacja wyników i modelu. W przypadku wykorzystania biblioteki w innej aplikacji dane dostarczane są w postaci macierzy wykorzystującej standardowe typy danych dostępne w STL (Standard Template Library) dostarczanej razem z kompilatorem C++.

Metoda Elementów Brzegowych bardzo dobrze nadaje się do obszarów jednorodnych. Na rysunku 1 pokazano model kondensatora płaskiego wraz z rozkładem potencjału między okładkami kondensatora do których przyłożono napięcie 0 V i 10 V odpowiednio dla lewej i prawej. Górny i dolny brzeg jest izolowany, więc pochodna potencjału $q = \frac{\partial u}{\partial n} = 0$. W modelu został zastosowany element brzegowy liniowy do aproksymacji brzegu oraz funkcji pola i jej pochodnej.



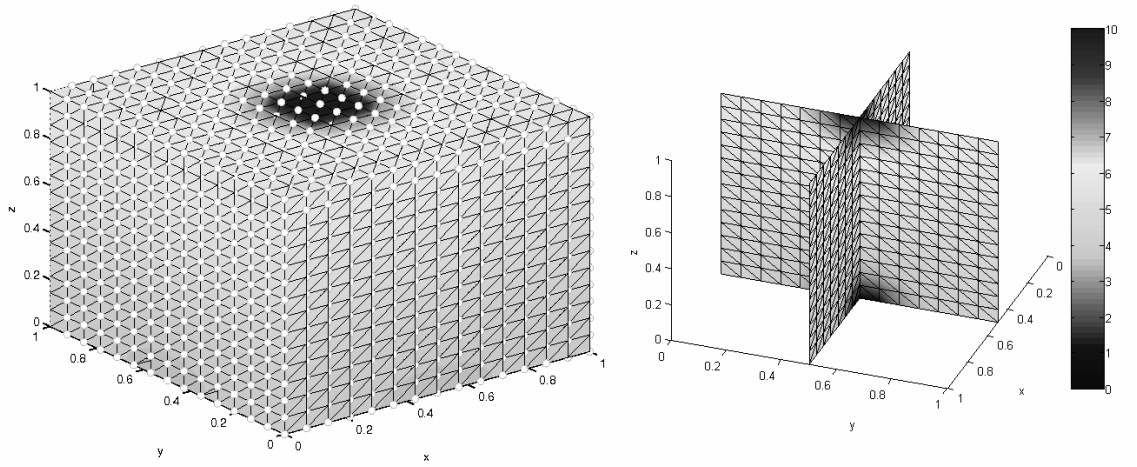
Rys. 1. Kondensator płaski - przykład 2D

Metoda Elementów Brzegowych umożliwia również efektywne rozwiązywanie zagadnień niejednorodnych strefowo. Na rysunku 2 pokazano model kondensatora płaskiego, gdzie przestrzeń między okładkami wypełniona jest materiami o różnym współczynniku materiałowym. Obszar wypełniony materiałem o współczynniku 2 został oznaczony jako Ω_1, natomiast obszar wypełniony materiałem o współczynniku 1 jako Ω_2. Do okładek przyłożono napięcie odpowiednio 0 V (lewa) i 10 V (prawa). Górny i dolny brzeg jest izolowany, więc pochodna potencjału $q = \frac{\partial u}{\partial n} = 0$. W modelu tym brzeg został zdyskretyzowany elementami liniowymi, natomiast aproksymacji potencjału oraz jego pochodnej dokonano elementami stałymi.



Rys. 2. Przykład dwuobszarowy

Na rysunku 3 zaprezentowano przykład użycia MEB do znalezienia rozkładu potencjału w cegle do której przyłożono dwie elektrody z potencjałem 0 V (w środku górnego brzegu) i 10 V (w środku dolnego brzegu). Pozostałe brzegi są izolowane, więc pochodna potencjału $q = \frac{\partial u}{\partial n} = 0$. W przykładzie tym



Rys. 3. Przykład 3D

dokonano dyskretyzacji brzegu oraz aproksymacji potencjału i jego pochodnej elementami brzegowymi liniowym trójkątnymi. Po prawej stronie rysunku 3

został przedstawiony rozkład potencjału wewnątrz cegły, na dwóch wybranych przecinających się płaszczyznach.

4.OBIEKTOWOŚĆ

Wraz z rozwojem systemów informatycznych, zwiększa się ich złożoność i w konsekwencji liczba linii kodu oraz liczba osób, która nad nimi pracuje. Projektowanie i zarządzanie takimi systemami jest trudne, dlatego wprowadzono nowe techniki programowania i projektowania. Programowanie obiektowe ułatwia pisanie, poprawianie i ponowne wykorzystanie tego samego kodu, w ramach jednego lub wielu projektów. W ogólności programowanie obiektowe różni się od proceduralnego tym, że dane i procedury są ze sobą związane w ramach klas materializowanych w postaci obiektów, będących instancjami klas. Dzięki zastosowaniu obiektowości w projekcie zwiększa się jego przejrzystość oraz jakość. Część jego abstrakcyjnych elementów tj. komponentów i klas z łatwością może zostać wykorzystane w innych projektach. Dzięki temu przedłuża się czas życia poszczególnych fragmentów kodu, czas na znajdowanie błędów i wprowadzanie poprawek. Zmniejsza się również prawdopodobieństwo „śmierci" danego projektu, gdyż jego część wciąż „żyje" w innych projektach i jest utrzymywana oraz dostosowywana do nowych wersji bibliotek i kompilatorów. Dzięki temu koszty utrzymania projektu maleją i można dłużej go utrzymywać.

Ważnym elementem w systemach informatycznych jest dokumentacja zarówno przedprojektowa, projekt jak i ostateczna (zaimplementowana) wersja projektu. Zapisywanie projektu w postaci kodu źródłowego jest bardzo nieefektywne, ponieważ ten etap życia projektu cechuje się częstymi zmianami architektury. Ponadto czytanie kodu źródłowego jest relatywnie trudne i również nieefektywne. Z tego powodu stworzono UML (ang. Unified Modelling Language, tłum. Ujednolicony Język Modelowania), który umożliwia zobrazowanie architektury i kodu źródłowego w postaci diagramów. Diagramy mogą przedstawiać projekt na różnych poziomach złożoności, tj. od ogólnej architektury w postaci głównych komponentów do diagramów sekwencji pokazujących kolejność wywołań poszczególnych funkcji na obiektach. Dzięki zastosowaniu ustandaryzowanych, ujednoliconych elementów określających np. komponenty, klasy, relacje, akcje każda osoba znająca ten język z łatwością zapozna się z budową danego projektu [9]. Jest to szczególnie ważne na etapie projektowania oraz wdrażania nowych architektów i programistów do pracy z oprogramowaniem.

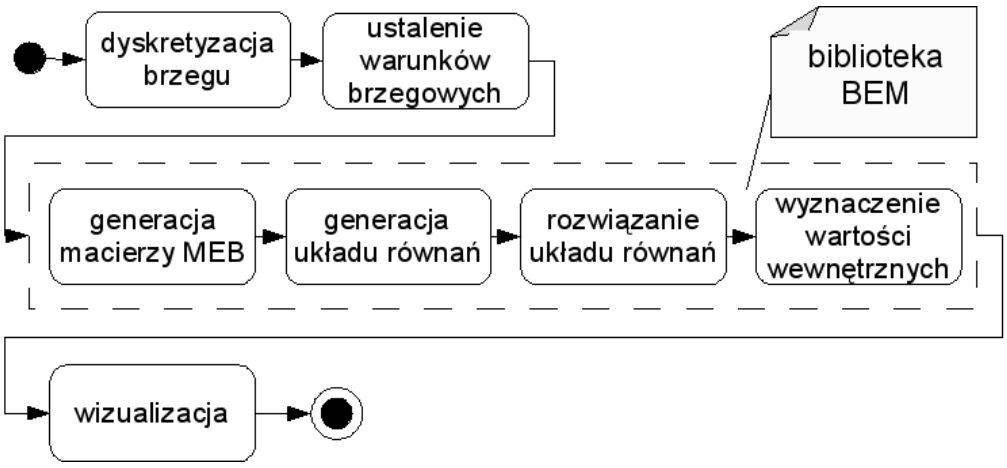
Obecnie programowanie obiektowe jest bardzo popularne. Ogromne środowiska graficzne znane z komputerów osobistych takie jak KDE dla Linux, czy bardzo popularny system Windows, przeglądarka Mozilla Firefox i system wyszukiwania Google są napisane obiektowo. Wszystkie aplikacje pisane w języku JAVA, zgodnie ze sztuką, są obiektowe.

Zalety programowania obiektowego i jego popularność spowodowały, że zostało wybrane do realizacji opisywanego projektu. Natomiast projektowanie i dokumentacja tworzona jest przy użyciu języka modelowania UML.

5. OBIEKTOWOŚĆ W BIBLIOTECE MEB

Kolejnym etapem projektu jest implementacja metody elementów brzegowych opisanej w rozdziale 3. Na podstawie MEB należy napisać algorytmy, które następnie są implementowane w wybranym języku programowania. W tym rozdziale przedstawione zostaną tylko ogólne diagramy, pokazujące ogólną koncepcję biblioteki i jej architektury.

Na rysunku 4 przedstawiono diagram aktywności zadania, które jest rozwiązywane metodą MEB.



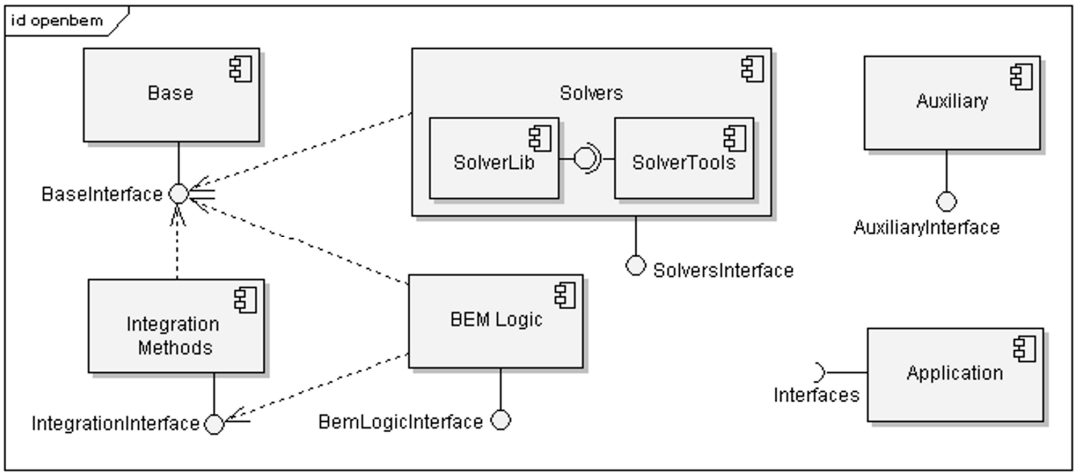
Rys. 4. Diagram aktywności procesu MEB

Akcje, które są realizowane przez opisywaną bibliotekę zostały objęte linią przerywaną. Biblioteka umożliwia otrzymanie danych wygenerowanych na etapach pośrednich między generacją macierzy MEB, a wyznaczaniem

wartości wewnętrznych. Dzięki temu możliwe jest wykorzystanie innego oprogramowania w celu przeprowadzenia częściowych obliczeń np. rozwiązywanie układu równań Ax = b może być realizowane przez zewnętrzny solwer.

Biblioteka nie zawiera i nie będzie zawierać w przyszłości modułów służących do generacji siatki, ustanowienia warunków brzegowych poprzez wybór elementów i wizualizacji. Wymienione akcje realizowane są w aplikacjach zewnętrznych.

Kolejnym diagramem tutaj zaprezentowanym jest diagram komponentów przedstawiony na rysunku 5.



Rys. 5. Diagram komponentów biblioteki MEB

Biblioteka składa się z pięciu komponentów. Wszystkie są niezbędne, ale wydaje się, że najbardziej interesujący jest komponent „bem”. Zawiera on klasy z zaimplementowanym algorytmem MEB. Rozwiązania fundamentalne (funkcje Greena) dla równań Laplace'a, Poissona, dyfuzji i Helmholtza. Elementy dyskretyzacji brzegowe i obszarowe. Wybrane funkcje kształtu.

Komponent „base” zawiera definicje kontenerów używanych przez pozostałe komponenty do wymiany danych.

Komponent „integration” zawiera klasy implementujące algorytmy całkowania takie jak np. kwadratura Gaussa.

Rozwiązywanie układu równań jest pozostawione komponentowi solver, który oprócz definicji własnych solwerów jak np. dekompozycja LU, posiada zaimplementowane wrappery do zewnętrznych bibliotek jak np. Sparskit2 implementującej solwery iteracyjne np. GMRES.

Natomiast komponent „auxiliary” zawiera pomocnicze funkcje wykorzystywane przez pozostałe komponenty, a klasy te nie występują w STL.

Wydaje się, że powyższe diagramy wprowadzają w architekturę biblioteki, natomiast inne, będą się ukazywać w kolejnych publikacjach.

6. PODSUMOWANIE

Wiele zastosowań Metody Elementów Brzegowych zostało opisanych w literaturze, na przestrzeni ostatnich lat. Jej efektywność w wielu wypadkach jest większa niż metod alternatywnych np. MES. Jednak dotąd, MEB nie doczekał się implementacji uniwersalnej, obiektowej biblioteki MEB o otwartym kodzie źródłowym spełniającej wymagania postawione przez biblioteki systemowe w środowiskach systemów operacyjnych Unix/Linux, czy standardy wytyczone przez środowisko otwartego oprogramowania GNU. Przyczyną zapewne są trudności w implementacji m.in. całkowania numerycznego osobliwości. Opisywana biblioteka ma zapełnić lukę na rynku oprogramowania i zwiększyć zainteresowanie MEB. Jej projekt obiektowy jest niezależny od platformy i języka programowania, a wybór architektury ma wspomóc tworzenie społeczności wokół niej.

LITERATURA

1. Wrobel L.C.: The boundary element method. Vol. 1, John Wiley & Sons, 2002.
2. Aliabadi M.H.: The boundary element method. Vol. 2, John Wiley & Sons, 2002.
3. Sikora J.: Boundary Element Method for Impedance and Optical Tomography, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2007.
4. Jabłoński P.: Metoda elementów brzegowych w analizie pola elektromagnetycznego, Wydawnictwo Politechniki Częstochowskiej, Częstochowa 2003.
5. Brebbia C.A. Boundary element method in engineering, Pentech Press, 1978.
6. Beer G.: Programmint the Boundary Element Method. An Introduction for Engineers, John Wiley & Sons, 2001.
7. Zienkiewicz O.C.:The Finite Element Method, 5th edition, Butterworth-Heinemann, September 2000.
8. Levesque John M., Joel W. Williamson: A Guidebook to Fortran on Supercomputers. Academic Pr, 1989.
9. Śmiałek M.: UML 2.0. Metody modelowania obektowego, Helion, 2005.

10. ABEM: http://www.boundary-element-method.com/.
11. LibBem: http://www.mis.mpg.de/scicomp/software/libbem/libbem.html.
12. BEMLIB: http://dehesa.freeshell.org/BEMLIB/bemlib.shtml.
13. BIE: http://www.math.uiowa.edu/~atkinson/bie.html.
14. BEASY: http://www.beasy.com/.
15. Integrated Engineering Software: http://www.integratedsoft.com/bem.asp.
16. GPBEST: http://www.gpbest.com/.
17. Concept Analysis: http://www.conceptanalyst.com/.

Rękopis dostarczono dnia 3.12.2008 r.
Opiniował: prof. dr hab. inż. Antoni CIEŚLA

OBJECTIVE BOUNDARY ELEMENT METHOD LIBRARY

Paweł WIELEBA, Jan SIKORA

ABSTRACT *This article presents objective Boundary Element Method library, primarily implemented for Unix/Linux operating systems in C++, but easily portable to other platforms with installable GNU compiler. The library's design and architecture allow its implementation in other objective languages preserving the structure and project. The objective design and used Unified Modeling Language (UML) substantially simplifies process of creating specification, designing, implementing and testing. Implementing Boundary Element Method is not straight forward. Usage of advanced tools for managing source code, versioning, releases as well as managing process of configuration, compilation, build and installation, which are standard among open source engineers, simplifies and even in some circumstances makes development of numerical BEM library possible. However development tools are not the major condition which must be fulfilled. There are substantial areas which have to be specially treated. Two of them can be mentioned first: singularity coming from the method which is expressed by c(r) coefficient in every generated BEM equation and singularity coming from the numerical integration algorithm of boundary elements. Further complications arise when multi-region problems are to be solved. Multiple regions are introduced when problem is not linear, but for example linear in consecutive domains. Then computational complexity rises and have to be compensated to keep the BEM implementation and particularly the discussed library competitive*

comparing to other BEM software and other numerical methods of solving partial differential equations. Usage of advanced tools which are settled down among open source community is a very important feature from the users point of view, because it does not require to learn new user interface as well as compilation, build or installation procedures. The user can start using library immediately. Very important feature for the user are ports, and source packages, which simplify process of installation on owns workstations and calculation units. The need for the library implementing BEM is substantiated in the article. Basics of Boundary Element Method from the library's point of view are described in the article stressing on the source code architecture needs and control flow of the process. Moreover there are simple and advanced real-life examples of usage and application presented as the library is being designed for both didactics as well as scientist and industrial groups.