УДК 519.688:539.3
Пастернак В.В., к.т.н.; Пастернак Я.М., д.ф.-м.н.
Луцький національний технічний університет

ОБ'ЄКТНО-ОРІЄНТОВАНА РЕАЛІЗАЦІЯ МЕТОДУ ГРАНИЧНИХ ЕЛЕМЕНТІВ ТРИВИМІРНОЇ ТЕРМОМАГНІТОЕЛЕКТРОПРУЖНОСТІ

Пастернак В.В., Пастернак Я.М. Об'єктно-орієнтована реалізація методу граничних елементів тривимірної термомагнітоелектропружності. Розглянуто принципи побудови об'єктно-орієнтованого сирцевого коду розробленого раніше методу граничних елементів тривимірної термомагнітоелектропружності. Вказано на переваги такої реалізації порівняно із процедурною. Запропоновано під'єднувати розроблені програми як модулі інтерпретованих мов високого рівня для створення потужних комплексів інженерного розрахунку.
Ключові слова: метод граничних елементів; сирцевий код; системи інженерних розрахунків

Пастернак В.В., Пастернак Я.М. Объектно-ориентированная реализация метода граничных элементов трехмерной термомагнитоэлектроупругости. Рассмотрены принципы построения объектно-ориентированного исходного кода разработанного ранее метода граничных элементов трехмерной термомагнитоэлектроупругости. Указано на преимущества такой реализации по сравнению с процедурной. Предложено подсоединять разработанные программы как модули интерпретированных языков высокого уровня для создания мощных комплексов инженерных расчетов.
Ключевые слова: метод граничных элементов; исходный код; системы инженерных расчетов

Pasternak V.V., Pasternak Ia.M. Object-oriented programming of boundary element method for 3D thermomagnetoelectroelasticity. The paper considers principles for development of object-oriented source code for 3D boundary element approach obtained earlier for 3D thermomagnetoelectroelasticity. It shows the advantages of the proposed approach comparing to the procedural one. It proposes to link the developed programs as modules for high-level interpreted programming languages for development of powerful computer aided engineering systems.
Keywords: boundary element method; source code; computer aided engineering

Вступ. На цей час у розробці програмного забезпечення, зокрема й комплексів інженерних розрахунків, паралельно використовується декілька парадигм програмування: процедурне, об'єктно-орієнтоване, шаблонне тощо. Це стосується і програмної реалізації методу граничних елементів (МГЕ). Традиційно тут використовується процедурне програмування мовою FORTRAN, адже розробка МГЕ розпочалася ще у 60-х роках XX століття [1], коли такі концепції були практично єдиними. З тих пір опубліковано велику кількість монографій та підручників (напр., [2–4]), де подано сирцевий код низки програм аналізу двовимірних та тривимірних задач теплопровідності, пружності та термопружності тіл із отворами, глобулярними включеннями та тріщинами.

Проте в два останні десятиліття при реалізації нових концепцій МГЕ все частіше використовується об'єктно-орієнтований підхід [5–7], що дає можливість інкапсулювати дані, а завдяки механізмам успадковування та поліморфізму є потужним засобом повторного використання коду та створення бібліотек для інтерпретованих мов високого рівня. Однак подані на даний час концепції програмної реалізації МГЕ є радше прикладами використання об'єктного підходу без орієнтації на його прикладне застосування у системах інженерних розрахунків та проектування (CAE – Computer Aided Engineering).

Тому у цій роботі здійснено опис основних етапів програмної реалізації раніше розробленого методу граничних елементів тривимірної термомагнітоелектропружності структурно-неоднорідних тіл [8], що дає можливість гнучкої розширюваності коду та його поєднання із готовими модулями для створення системи інженерних розрахунків.

Основні співвідношення та обчислювальна схема МГЕ [8]. Узагальнені у [8] визначальні співвідношення лінійних стаціонарних термомагнітоелектропружності та теплопровідності мають такий вигляд:

$$\tilde{\sigma}_{Ij,\,j} + \tilde{f}_I = 0\,,\;\; h_{i,i} - f_h = 0\,; \tag{1}$$

$$\tilde{\sigma}_{Ij} = \tilde{C}_{IjKm}\tilde{u}_{K,m} - \tilde{\beta}_{Ij}\theta\,,\;\; h_i = -k_{ij}\theta_{,j}\,, \tag{2}$$

де

$$
\begin{aligned}
&\tilde{u}_i = u_i,\ \tilde{u}_4 = \phi,\ \tilde{u}_5 = \psi\ ;\ \tilde{f}_i = f_i,\ \tilde{f}_4 = -q,\ \tilde{f}_5 = b_m;\\
&\tilde{\sigma}_{ij} = \sigma_{ij},\ \tilde{\sigma}_{4j} = D_j,\ \tilde{\sigma}_{5j} = B_j;\\
&\tilde{C}_{ijkm} = C_{ijkm},\ \tilde{C}_{ij4m} = e_{mij},\ \tilde{C}_{4jkm} = e_{jkm},\ \tilde{C}_{4j4m} = -\kappa_{jm},\\
&\tilde{C}_{ij5m} = h_{mij},\ \tilde{C}_{5jkm} = h_{jkm},\ \tilde{C}_{5j5m} = -\mu_{jm},\\
&\tilde{C}_{4j5m} = -\gamma_{jm},\ \tilde{C}_{5j4m} = -\gamma_{jm};\\
&\tilde{\beta}_{ij} = \beta_{ij},\ \tilde{\beta}_{4j} = -\chi_j,\ \tilde{\beta}_{5j} = \nu_j\quad (i,j,k,m = 1,2,3).
\end{aligned}
\tag{3}
$$

Тут σ_{ij} – компоненти тензора напружень; h_i – компоненти вектора густини теплового потоку; D_i – електричне зміщення; B_i – індукція магнітного поля; f_i – об'ємні сили; q – густина вільних зарядів; f_h – густина розподілених джерел (стоків) тепла; b_m – об'ємна густина постійного електричного струму, яка для діелектрика є нульовою; u_i – переміщення точок тіла; ϕ – електричний потенціал; ψ – потенціал стаціонарного магнітного поля; θ – зміна температури порівняно з відліковою; C_{ijkm} – пружні сталі; k_{ij} – коефіцієнти теплопровідності; β_{ij} – модулі теплового розширення (коефіцієнти теплових напружень); e_{ijk} – п'єзоелектричні сталі; κ_{ij} – діелектричні сталі матеріалу; h_{ijk} – п'єзомагнітні сталі; μ_{ij}, γ_{ij} – магнітна та електромагнітна проникності матеріалу; χ_i – піроелектричні коефіцієнти; ν_i – піромагнітні коефіцієнти. Тензори з компонентами C_{ijkm}, k_{ij}, κ_{ij}, μ_{ij}, γ_{ij} та β_{ij} вважаються симетричними. У формулах прийняте правило Айнштайна підсумовування за індексом, що повторюється. Кома в індексах означає операцію диференціювання за координатою, індекс якої стоїть після коми, тобто, $u_{i,j} \equiv \partial u_i / \partial x_j$. Великі індекси змінюються від 1 до 5, а маленькі від 1 до 3, тобто $I = 1,2,...,5$, $i = 1,2,3$.

Відповідні їм крайові інтегральні рівняння теплопровідності та термомагнітоелектропружності отримані у формі

$$
\begin{aligned}
\frac{1}{2}\theta(\mathbf{x}_0) = &\iint_{\partial \mathsf{B}} \Theta^*(\mathbf{x},\mathbf{x}_0) h_n(\mathbf{x})\, dS(\mathbf{x}) - \mathrm{CPV}\iint_{\partial \mathsf{B}} H^*(\mathbf{x},\mathbf{x}_0)\theta(\mathbf{x})\, dS(\mathbf{x})\\
&- \iiint_{\mathsf{B}} \Theta^*(\mathbf{x},\mathbf{x}_0)\, f_h(\mathbf{x})\, dV(\mathbf{x}),
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
\frac{1}{2}\tilde{u}_I(\mathbf{x}_0) = &\iint_{\partial \mathsf{B}} U_{IJ}(\mathbf{x},\mathbf{x}_0)\tilde{t}_J(\mathbf{x})\, dS(\mathbf{x}) - \mathrm{CPV}\iint_{\partial \mathsf{B}} T_{IJ}(\mathbf{x},\mathbf{x}_0)\tilde{u}_J(\mathbf{x})\, dS(\mathbf{x})\\
&+ \iint_{\partial \mathsf{B}} \left[R_I(\mathbf{x},\mathbf{x}_0)\theta(\mathbf{x}) + V_I(\mathbf{x},\mathbf{x}_0) h_n(\mathbf{x}) \right] dS(\mathbf{x})\\
&+ \iiint_{\mathsf{B}} U_{IJ}(\mathbf{x},\mathbf{x}_0)\, \tilde{f}_J(\mathbf{x})\, dV(\mathbf{x}) - \iiint_{\mathsf{B}} V_I(\mathbf{x},\mathbf{x}_0)\, f_h(\mathbf{x})\, dV(\mathbf{x}),
\end{aligned}
\tag{5}
$$

де $\mathbf{x}_0 \in \partial \mathsf{B}$, а межа $\partial \mathsf{B}$ тіла B є вважається гладкою у точці колокації $\mathbf{x}_0$.

Для розв'язування інтегральних рівнянь (4), (5) у [8] запропоновано такий алгоритм МГЕ:

1. межа тіла розбивається на розривні квадратичні чотиристоронні елементи;
2. для кожного елемента задаються функції форми для інтерполяції крайових функцій за вузловими значеннями, зокрема, для крайових елементів тріщин чи клинів останні враховують степеневу особливість розв'язку;
3. здійснюється числове інтегрування отриманих після перших двох кроків означених поверхневих інтегралів із застосуванням нелінійних відображень та спеціальних квадратур для побудови системи лінійних алгебричних рівнянь стосовно невідомих вузлових значень крайових функцій;

4. розв'язується отримана система рівнянь та визначається термомагнітоелектронапружений стан у бажаних точках тіла, а також коефіцієнти інтенсивності фізико-механічних полів у точках особливої геометрії.

Перші два кроки пов'язані з геометрією задачі та крайовими умовами, тому їх доцільно інкапсулювати в один абстрактний тип даних, що описуватиме граничні елементи. Інші два кроки пов'язані безпосередньо із розв'язуванням задачі, тому в об'єктній моделі вони віднесені до іншого класу.

Об'єктно-орієнтована реалізація МГЕ термомагнітоелектропружності. Для чотиристоронніх граничних елементів створено клас, декларативну частину якого подано нижче.

```
class BoundaryElement
{
private:
//Вузли топології елемента звичайні, нерозривні
    double NodesX[3][3],
           NodesY[3][3],
           NodesZ[3][3];

    double Lagr1DTopol(int p, double t);
    double DiffLagr1DTopol(int p, double t);

public:
//Вузли задання крайових умов розривні
    double tP[5][3][3],
           tM[5][3][3],
           uP[5][3][3],
           uM[5][3][3],
           thetaP[3][3],
           thetaM[3][3],
           hnP[3][3],
           hnM[3][3];

    BEType ElementType;
    HTType ElementHTType;

    BoundaryElement(void);
    BoundaryElement(double X[3][3], double Y[3][3], double Z[3][3]);
    void SetDefaultShapeFunctions(void);
    void SetShapeFunctions(int DispShFn, int TracShFn,
                           int TempShFn, int HeatShFn);
    void SetElement(double X[3][3], double Y[3][3], double Z[3][3]);
    void Position(double xi, double eta, double x[3]);
    void DiffR(double xi, double eta, double dr_dxi[3], double dr_deta[3]);
    void Normal(double xi, double eta, double n[3], double &J);
    void RotateElement(double xC[3], double phi_x,
                       double phi_y, double phi_z);
    void TranslateElement(double x, double y, double z);

    double ShapeFunctionDisp(const int n, const double xi, const double eta);
    double ShapeFunctionTrac(const int n, const double xi, const double eta);
    double ShapeFunctionTemp(const int n, const double xi, const double eta);
    double ShapeFunctionHeat(const int n, const double xi, const double eta);

};
```

Як видно з лістингу, приватна частина класу містить три 3х3-масиви, що містять абсциси, ординати та аплікати 9 вузлів граничного елемента. Також клас має приватні методи `Lagr1DTopol` та `DiffLagr1DTopol` що задають одновимірні інтерполяційні поліноми Лагранжа та їх похідні, які використовуються для апроксимації елемента поверхнею другого порядку

$$\mathbf{x}(\xi,\eta) = \sum \mathbf{x}_{ij} L_i(\xi) L_j(\eta). \tag{6}$$

Відкрита частина класу містить масиви для зберігання крайових умов у вузлах інтерполяції (`tP`, `tM`, `uP`, `uM`, `thetaP`, `thetaM`, `hnP`, `hnM`), а також змінні, що відображають які з умов є заданими (`ElementType`, `ElementHTType`). Віднесення цих змінних до відкритої частини класу хоч і порушує «хороший тон» об'єктного програмування, проте є зручним та зменшує код програми.

Окрім створених методів задання вузлів елемента (`SetElement`) та функцій форми (`SetShapeFunctions`) зручними у користуванні є методи `RotateElement`, `TranslateElement`, що дають можливість повертання та паралельного перенесення граничного елемента. Зокрема, для створення зображеної на рис. 1 сітки дископодібної тріщини достатньо задати вузли 1 внутрішнього та 1 зовнішнього елемента, а потім шляхом копіювання і повертання створити решту елементів.

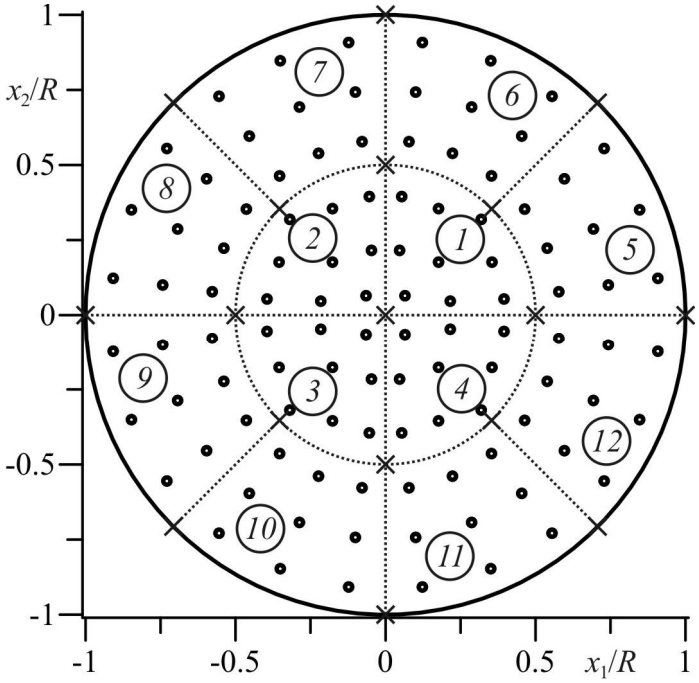

Рис. 1. Сітка розбиття для дископодібної тріщини

Граничноелементні методи розв'язування інтегральних рівнянь (4), (5) інкапсульовані в класі `ThermoSolid3D`, лістинг якого подано нижче (для компактності викладу не подано захищених методів обчислення ядер рівнянь та їхнього інтегрування).

```
class ThermoSolid3D
{
protected:
    double CC[5][3][5][3],
           beta[5][3],
           kk[3][3];

    double SS[5][3][5][3],
           alpha[5][3];

    BoundaryElement *Elements;
    int NElements;

    WinklerInclusion3D *WinklerInclusions;
    int NWinklerInclusions;

    int NForces;
    ConcFactor *Forces;

    int NHeatSources;
```

```
    HeatSource *HeatSources;

    double ThetaInf;
    double SigmaInf[5][3], UI_j_Inf[5][3];

public:
    ThermoSolid3D(double C1[5][3][5][3],
                  double beta1[5][3],
                  double k1[3][3]);
    ~ThermoSolid3D();
    void AllocateElements(const int Number);
    void SetBEGeometry(const int Elem, double X[3][3],
                       double Y[3][3], double Z[3][3]);
    void RotateBE(const int Elem, double xC[3],
                  double phi_x, double phi_y, double phi_z);
    void TranslateBE(const int Elem, double x, double y, double z);
    void SetBEType(const int Elem, const int type=1, const int HTtype=1);
    void SetBEConditions(const int Elem,
                         double HT_BCp[3][3], double HT_BCm[3][3],
                         double MEE_BCp[5][3][3],
                         double MEE_BCm[5][3][3]);

    void BEMSolver(void);
    void HTTemperature(double x0[3], double &theta);
    void HTHeatFlux(double x0[3], double h[3]);
    void Displacement(double x0[3], double u[5]);
    void Stress(double x0[3], double sigma[5][3]);
    void FIF(int Element, int flag, double x,
             double x0[3], double KK[5], double Kt[2]);
    void SetShapeFunctions(int Element, int DispShFn,
                           int TracShFn, int TempShFn, int HeatShFn);
    void SetSigmaInf(double sigma[5][3]);
    void SetThetaInf(const double theta);
    void AddCForce(ConcFactor Force);
    void DeleteCForces(void);
    void AddHeatSource(HeatSource Source);
    void DeleteHeatSources(void);
};
```

Закрита частина класу містить дані про фізичні властивості матеріалу тіла, вказівник на масив граничних елементів, пам'ять під який виділяється динамічно методом `AllocateElements`. Також захищена частина класу містить динамічні масиви, що задають розташування та інтенсивність зосереджених джерел тепла та розширених сил, навантаження на безмежності тощо.

Методи `SetBEGeometry`, `RotateBE`, `TranslateBE`, `SetBEType`, `SetBEConditions`, `SetShapeFunctions` класу у свою чергу викликають відповідні методи класу граничних елементів для маніпуляцій з останніми.

Метод `BEMSolver` є реалізацією запропонованої у [8] схеми граничноелементного розв'язування інтегральних рівнянь (4), (5). Після цього методи `HTTemperature`, `HTHeatFlux`, `Displacement`, `Stress` дають можливість обчислити температуру, тепловий потік, розширені переміщення та напруження у довільній точці тіла. Крім цього, метод `FIF` дає можливість визначати коефіцієнти інтенсивності фізико-механічних полів на фронті тріщин та тонких податних слабо проникних включень.

За допомогою програми SWIG [9] отриманий об'єктний код інтегровано до середовища інтерпретованої мови високого рівня Python. Завдяки цьому створено комплекс інженерних розрахунків на основі побудованої бібліотеки та додатків NumPy, SciPy, matplotlib.

Висновки. Завдяки об'єктно-орієнтованій структурі створений сирцевий код методу граничних елементів тривимірної термомагнітолектропружності структурно неоднорідних тіл дає можливість його поєднання з розвиненими багатомодульними програмними комплексами інтерпретованих мов високого рівня, завдяки чому можна створювати потужні комплекси інженерних розрахунків. Окрім цього, із

використанням механізму успадковування розроблений код можна легко поширювати і на інші класи задач, а саме для півпростору чи біматеріалу, що забезпечує ефективне повторне використання коду та гнучкість моделі.

1. Cheng A.H.D. Heritage and early history of the boundary element method / A.H.D. Cheng, D.T. Cheng // Engineering Analysis with Boundary Elements. – 2005. – **29**, 3. – P. 268–302.
2. Бенерджи П. Методы граничных элементов в прикладных науках / П. Бенерджи, Р. Баттерфилд. – М.: Мир, 1984. – 494 с.
3. Dominguez J. Boundary elements in dynamics / J. Dominguez. – Boston: Computational Mechanics Publications, 1993. – 700 p.
4. Beer G. The boundary element method with programming / G. Beer, I. Smith, C. Duenser. – New York: Springer, 2008. – 496 p.
5. Lage C. The application of object-oriented methods to boundary elements / C. Lage // Comput. Methods Appl. Mech. Engrg. – 1998. – **157**. – P. 205–213.
6. Qiao H. Object-oriented programming for the boundary element method in two-5al heat transfer analysis / H. Qiao // Advances in Engineering Software. – 2006. – **37**. – P. 248–259.
7. Wang W. Object-oriented programming in boundary element methods using C++ / W. Wang, X. Ji, Y. Wang // Advances in Engineering Software. – 1999. – **30**. – P. 127–132.
8. Pasternak Ia. A comprehensive study on Green's functions and boundary integral equations for 3D anisotropic thermomagnetoelectroelasticity / Ia. Pasternak, R. Pasternak, H. Sulym // Engineering Analysis with Boundary Elements. – 2016. – **64**. – P. 222–229.
9. http://www.swig.org/