9. PRZYKŁAD OPROGRAMOWANIA MES

9.1. SFORMULOWANIE PROBLEMU

W ostatnim rozdziale przedstawimy przykład oprogramowania, rozwiązującego dwuwymiarowy problem nieustalonej wymiany ciepła. Podstawy teoretyczne tego rozwiązania są podane w rozdziałach 4 i 5. Układ równań MES można uzyskać za pomocą formuły (5.8):

$$\left(\left[H\right]+\frac{\left[C\right]}{\Delta\tau}\right)\left\{t_\tau\right\}-\left[\frac{\left[C\right]}{\Delta\tau}\right]\left\{t_{\tau-1}\right\}+\left\{P\right\}=0\,, \tag{5.8}$$

gdzie składniki formuły (5.8) można obliczyć za pomocą formuł (4.14), (4.15) i (5.3):

$$\left[H\right]=\int_V k\left[\frac{\partial[N]}{\partial x}\right]^T\left[\frac{\partial[N]}{\partial x}\right]^T+\left[\frac{\partial[N]}{\partial y}\right]^T\left[\frac{\partial[N]}{\partial y}\right]^T dV+ \tag{4.14}$$
$$+\int_S \alpha[N][N]^T dS$$

$$\left\{P\right\}=-\int_S \alpha[N]t_\infty dS \tag{4.15}$$

$$\left[C\right]=\int_V c\rho[N][N]^T dV\,. \tag{5.3}$$

W formule (5.8) $\left\{t_\tau\right\}$ - wartości temperatur węzłowych po czasie $\Delta\tau$, wektor $\left\{t_{\tau-1}\right\}$ reprezentuje temperatury węzłowe w chwili $\tau = 0$.

Ponieważ analogiczną formę mają równania dla zadani dyfuzji, opracowany program po zmianie współczynników można wykorzystać do modelowania procesów dyfuzji. Wyzerowanie macierzy $\left[C\right]$ sprowadzisz rozwiązanie ustalonego zadania cieplnego, albo innego, opartego o równanie Poissona (6.65).

9.2. DANE WEJŚCIOWE

Dane wejściowe do opracowanego programu są zadawane w postaci pliku tekstowego. Przykładową zawartość takiego pliku, z opisem wczytywanych zmiennych, pokazano na rys. 9.1.

```
1000   mTbegin     Temperatura początkowa, C
1000   mTime       Czas procesu, s
1      mdTime      Krok czasowy, s
1200   mT_otoczenia  Temperatura otoczenia C
300    mAlfa       Współczynnik wymiany ciepła W/Cm2
100    mHd         Wysokość przekroju, mm
100    mBD         Szerokość przekroju, mm
25     mNbH        Liczba węzłów po wysokości
25     mNbH        Liczba węzłów po szerokości
700    mC          Ciepło właściwe, J/Cag
25     mK          Współczynnik przewodzenia ciepła
7800   mR          Gęstość kg/m3
```

Rys. 9.1. Przykład zawartości pliku z danymi do obliczeń.

9.3. KOD ŹRÓDŁOWY PROGRAMU

Program zawiera kilka modułów, zawartość najważniejszych z nich jest opisana poniżej. Pełny tekst programu w języku FORTRAN90 można znaleźć na stronie internetowej http://home.agh.edu.pl/~milenin/programy/.

9.3.1. PROGRAM GŁÓWNY TEMP2D

Listing głównego programu TEMP2D jest podany na rys. 9.2. Oznaczenia głównych zmiennych są następujące:

GlobData – nazwa modułu z opisem globalnych zmiennych.

dTmMax – maksymalnie dopuszczalny krok czasowy, obliczony z warunku numerycznej stabilności rozwiązania.

n, Nlas – numer bieżącego kroku czasowego i ilość kroków czasowych.

Oznaczenia głównych procedur, wczytywanych w programie głównym są następujące.

InitL4 – procedura inicjalizacji danych dla wybranego typu elementu skończonego (wybrany typ 4-węzłowy element, opisany w rozdziale (3.9)).

InpData – wczytywanie dane wyjściowych.

GenGridZd – rozbudowa siatki MES dla obszaru o przekroju prostokątnym.

SetControlPoints – definicja punktów, z jakich będzie wyprowadzane do pliku tekstowego wyniki rozwiązania.

ALLOCATE Matrix – przydzielenie pamięci dla tablic.

WriteControlPointsBegin – zapis do pliku z wynikami obliczeń danych wstępnych.

WriteControlPoints – zapis do pliku wyników obliczeń.

SOLVER – procedura rozwiązywania zadania dla bieżącego kroku czasowego.

```
!*****************************************************************
! PROGRAM: TEMP2D
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      program TEMP2D
        use GlobData;
        implicit none
        real*8  :: Aar,dTauMax,TauP;
        integer*4 :: n, Nlas, i, iErr;

      call InitL4();
      call InpData();
      call GenGridZd(mHD,mBD,mNbH,mNbB, mGr);
      call SetControlPoints();
      CALL ALLOCATE_Matrix();
      CALL WriteControlPointsBegin();

        Aar = mK / (mC * mR);
        mdTime = DBLE(mBD)*1e3*mNbH)/(7*2)0.5*Aar);
      CALL WriteControlPoints();
        Nlas = int(mTimeInd/Time)+1;
        mdTime = mTimeDBLE(Nlas);
        mTau  = 0;
        do n=1, Nlas
                mTau = mTau + mdTime;
                CALL SOLVER( );
                CALL WriteControlPoints();
        end do;
        CLOSE(88);     CLOSE(89);
        DEALLOCATE ( mA, STAT=iEr );
        DEALLOCATE ( mB, STAT=iEr );
        DEALLOCATE ( mX, STAT=iEr );
      end program TEMP2D
```

Rys. 9.2. Główny program TEMP2D

9.3.2. MODUŁ Z DEFINICJĄ STRUKTUR DANYCH

Wykorzystanie struktur danych ułatwia proces programowania i analizy kodu. Na rys. 9.3 pokazano listing modułu My_typ z definicją struktur danych. Wykorzystano następujące struktury:

Node – struktura danych, zawierająca informacje o węźle siatki: x, y – współrzędne węzła, t – temperatura w węźle, CR – prędkość zmiany temperatury.

Element – struktura danych, zawierająca informacje o elemencie siatki MES: nop – mapa elementu (rozdział 7.21), Npov – liczba powierzchni elementu, kontaktujących z otoczeniem; aPov – lokalne numery kontaktowych powierzchni elementu.

grZd – struktura danych, zawierająca siatkę MES: nh – liczba węzłów; ne – liczba elementów; nbn – liczba węzłów w wykorzystywanym elementu skończonym; ncn – liczba zmiennych w elemencie skończonym, dla zadań cieplnych odpowiada nbn; nhPov – liczba węzłów siatki, znajdujących się na powierzchni rozpatrywanego obszaru; EL – tablica długości ne, zawierająca dane typu Element dla każdego elementu siatki; ND – tablica długości nh, zawierająca dane typu Node dla każdego węzła siatki.

cor_L – struktura danych, zawierająca współrzędne N, E punktu w układzie lokalnym elementu.

Gran – struktura danych, zawierająca dane o granicach powierzchniowych wykorzystywanego elementu skończonego. N_p – liczba punktów całkowania po powierzchni granic; P – tablica elementów typu cor_L długości N_p zawierająca współrzędne lokalne punktów całkowania Gaussa; W – tablica długości N_p zawierająca wagę punktów całkowania odpowiednio formuła (3.31); NI – tablica o wymiarach nbn, N_p zawierająca wartości funkcji kształtu elementu, obliczone w punktach całkowania na powierzchni granic; N1,N2 – tablicy o wymiarach nbn, N_p zawierające wartości pochodnych funkcji kształtu względem współrzędnych lokalnych, obliczone w punktach całkowania na powierzchni granic; UZEL – tablica zawierająca lokalne numery węzłów granic.

ELEM – struktura danych, zawierająca dane o wykorzystywanym elemencie skończonym: nbn – liczba węzłów w wykorzystywanym elemencie skończonym, nbnp – liczba węzłów w wykorzystywanym elemencie skończonym do interpolacji irradiacji napromienia (wykorzystywana w zadaniach temi pilotrzymości), N_p – liczba punktów całkowania w objętości elementu; NF – tablica o wymiarach nbn, N_p zawierająca wartości funkcji kształtu elementu, obliczone w punktach całkowania w objętości elementu; N1, N2 – tablicy o wymiarach nbn, N_p zawierające wartości pochodnych funkcji kształtu względem współrzędnych lokalnych, obliczone w punktach całkowania w objętości wykorzystywanego elementu skończonego; P – tablica elementów typu cor_L długości N_p zawierająca współrzędne lokalne punktów całkowania Gaussa w objętości elementu; L – tablica elementów typu cor_L długości N_p zawierająca współrzędne lokalne węzłów elementu; W – tablica długości N_p zawierająca wagę punktów całkowania w objętości elementu odpowiednio formuła (3.31); SF – tablica długości równej maksymalnej liczbie powierzchni wykorzystywanego elementu skończonego (dla wykorzystywanego elementu równa 4), zawierająca elementy typu Gran.


```
!*****************************************************************
! Typy danych
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      module My_typ;
      implicit none;

        type node
                real*8 :: x,y,t,CR;
                integer*4 :: status;
        end type

        type element
        integer*4 :: nop(4);
        integer*4 :: Npov;          ! Liczba kontaktowych powierzchni elementu 0-1-2
        integer*4 :: aPov(2);       ! Lokalne numery kontaktowych powierzchni elementu
        end type

        type grZd
                integer*4 : nh, ne, nbn, ncn, nhPov;
                type(element) , dimension(:), pointer :: EL;
                type(node) , dimension(:), pointer :: ND;
        end type

        type cor_L;
                real*8 N, E;
        end type cor_L;

        type Gran
                integer*4 :: N_p;  ! Liczba punktów całkowania po powierzchni granI
                type(cor_L), dimension(:), pointer :: P;
                real*8, dimension(:) , pointer  :: W;
                real*8, dimension(:,:), pointer  :: N1,N2,NI;          ! N1(nbn,N_p)
                integer*4, dimension(:), pointer  :: UZEL;
                ! Lokalne numery węzłów granI
        end type Gran;

        type ELEM;
                integer*4 : nbn, nbnp, N_p;
                real*8, dimension(:,:), pointer :: N1,N2,NF;   ! N1(nbn,N_p)
                type(cor_L), dimension(:), pointer :: P;       ! P(N_p)
                real*8, dimension(:) , dimension(:), pointer : L; ! Współrzędne węzłów elementu
                real*8, dimension(:), pointer : W;  ! Waga punktów całkowania W(N_p)
                type(Gran) :: SF(4);
        end type ELEM;

      end module My_typ;
```

Rys. 9.3. Typy (struktury) danych, potrzebne do rozwiązania.

9.3.3. MODUŁ Z DEFINICJĄ ZMIENNYCH GLOBALNYCH

Moduł GlobData zawiera definicje zmiennych, widocznych we wszystkich procedurach, gdzie zadeklarowano wykorzystanie tego modułu: use GlobData. Dokładny opis zmiennych podano na rys. 9.4.


```
      module GlobData;
!*****************************************************************
! Globalne zmienne
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      use My_typ;
      implicit none;
      REAL*8   :: mTbegin;          ! Początkowa temperatura
      real*8   :: mTime;            ! Czas procesu
      real*8   :: mdTime;           ! Początkowa wartość przyrostu czasu
      real*8   :: mTau;             ! Bieżący czas
      real*8   :: mT_otoczenia;     ! Temperatura otoczenia
      real*8   :: mAlfa;            ! Współczynnik wymiany ciepła
      real*8   :: mHD;              ! Wysokość przekroju
      real*8   :: mBD;              ! Szerokość przekroju
      integer*4  :: mNbH;           ! Liczba węzłów po wysokości przekroju
      integer*4  :: mNbB;           ! Liczba węzłów na szerokości przekroju
      integer*4  :: mLDA;           ! Szerokość pasma matrycy MES
      real*8   :: mC;               ! Pojemność cieplna
      real*8   :: mK;               ! Współczynnik przewodzenia ciepła
      real*8   :: mR;               ! Gęstość
      type(grZd) :: mGr;            ! Siatka MES
      type(ELEM) :: mEL4;           ! Dane uzyjnego elementu skończonego
      integer*4 :: mContrPointsDt;  ! Kontrolne punkty
      REAL*8   :: mxp(30),myp(30);
      real*8, dimension(4,4) :: est;  ! Matryca bieżącego elementu
      real*8, dimension(4)   :: r;    ! Wektor obciążeń bieżącego elementu
      real*8, dimension(:),  allocatable :: mB; ! Globalny wektor obciążeń
      real*8, dimension(:,:), allocatable :: mA; ! Globalna matrycaa
      real*8, dimension(:),  allocatable :: mX; ! Globalny wektor niewiadomych
      end module GlobData;
```

Rys. 9.4. Globalne zmienne, wykorzystywane w programie.

9.3.4. PROCEDURA INICJALIZACJI DANYCH DLA ELEMENTU

Wypełnienie zmiennej mEL4 typu ELEM odbywa się w programie InitL4, pokazanej na rys. 9.5. W pierwszej części programu zadajemy liczbę węzłów elementu równą 4. Wybrany schemat całkowania numerycznego zawiera 4 punkty odpowiadzisz rys. 3.19, dlatego zadajemy $N_p=4$. Dalej przydzielamy pamięć tablicom N1, N2, NI, P, W oraz L. Dalej wypełniamy współrzędne lokalne punktów całkowania odpowiadzisz rys. 3.19. Następnie obliczamy w każdym punkcie całkowania wartości N1, N2, NI, wykorzystując formuły (3.9):

$$N_1=\frac{1}{4}(1-\xi)(1-\eta)$$
$$N_2=\frac{1}{4}(1+\xi)(1-\eta)$$
$$N_3=\frac{1}{4}(1+\xi)(1+\eta) \tag{3.9}$$
$$N_4=\frac{1}{4}(1-\xi)(1+\eta)$$

Analogiczne wypełniamy strukturę dla 4 powierzchni wybranego elementu.

Przy wykorzystaniu innego typu elementów skończonych potrzebuje zmiany jedynie strukturę mEL4 i sama siatka elementów.

```
!*****************************************************************
! Inicjalizacja danych dla elementu
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      SUBROUTINE InitL4();
      use GlobData;
      implicit none;
      REAL*8   :: Alfa,L1,L2,e,n,SN;
      integer*4 :: iP,j;

        Alfa = 1/sqrt(3.0);
        mEL4%nbn=4;
        mEL4%N_p=4;
        ALLOCATE(mEL4%N1) mEL4%nbn, mEL4%N_p );
        ALLOCATE(mEL4%N2) mEL4%nbn, mEL4%N_p );
        ALLOCATE(mEL4%NI) mEL4%nbn, mEL4%N_p );
        ALLOCATE(mEL4%P ( mEL4%N_p );
        ALLOCATE(mEL4%W ( mEL4%N_p );
        ALLOCATE(mEL4%L ( mEL4%nbn );
        mEL4%SF = 1.0;
        mEL4%P(1)%E = -Alfa;
        mEL4%P(1)%N = -Alfa;
        mEL4%P(2)%E = Alfa;
        mEL4%P(2)%N = -Alfa;
        mEL4%P(3)%E = Alfa;
        mEL4%P(3)%N = Alfa;
        mEL4%P(4)%E = -Alfa;
        mEL4%P(4)%N = Alfa;
        mEL4%L(1)%E = -1;
        mEL4%L(1)%N = -1;
        mEL4%L(2)%E = 1;
        mEL4%L(2)%N = -1;
        mEL4%L(3)%E = 1;
        mEL4%L(3)%N = 1;
        mEL4%L(4)%E = -1;
        mEL4%L(4)%N = 1;
      DO iP = 1, mEL4%N_p
                L1 = mEL4%P(iP)%E
                L2 = mEL4%P(iP)%N
                mEL4%N1(iP ) = 0.25*(1-L1)*(1-L2);
                mEL4%N1(1,iP) = 0.25*(1-L2);
                mEL4%N2(1,iP) = -0.25*(1-L1);
                mEL4%N2(2,iP) = 0.25*(1+L1)*(1-L2);
                mEL4%N1(2,iP) = 0.25*(1+L2);
                mEL4%N2(3,iP) = 0.25*(1+L1);
                mEL4%N1(3,iP) = 0.25*(1+L1)*(1+L2);
                mEL4%N1(3,iP) = 0.25*(1+L2);
                mEL4%N2(3,iP) = 0.25*(1+L1);
                mEL4%N1(4,iP) = 0.25*(1-L1)*(1+L2);
                mEL4%N2(4,iP) = 0.25*(1-L1);
        end do;

      DO j = 1, 4
        mEL4%SI(j)%N_p = 2;
        ALLOCATE( mEL4%SI(j)%P( mEL4%SI(j)%N_p ) );
        ALLOCATE( mEL4%SI(j)%W( mEL4%SI(j)%N_p ) );
        mEL4%SI(j)%W = 1;
        ALLOCATE( mEL4%SI(j)%N1( mEL4%nbn, mEL4%SI(j)%N_p));
        ALLOCATE( mEL4%SI(j)%N2( mEL4%nbn, mEL4%SI(j)%N_p));
        ALLOCATE( mEL4%SI(j)%NI( mEL4%nbn, mEL4%SI(j)%N_p));
        ALLOCATE( mEL4%SI(j)%UZEL( 2 ) );
      END DO;

      mEL4%SI(1)%UZEL(1) = 4;      mEL4%SI(1)%UZEL(2) = 1;
      mEL4%SI(2)%UZEL(1) = 1;      mEL4%SI(2)%UZEL(2) = 2;
      mEL4%SI(3)%UZEL(1) = 2;      mEL4%SI(3)%UZEL(2) = 3;
      mEL4%SI(4)%UZEL(1) = 3;      mEL4%SI(4)%UZEL(2) = 4;
      mEL4%SI(1)%P(1)%N=Alfa;      mEL4%SI(1)%P(1)%E= 1;
      mEL4%SI(1)%P(2)%N=-Alfa;     mEL4%SI(1)%P(2)%E=-1;
      mEL4%SI(2)%P(1)%N= 1;        mEL4%SI(2)%P(1)%E=-Alfa;
      mEL4%SI(2)%P(2)%N=-1;        mEL4%SI(2)%P(2)%E=Alfa;
      mEL4%SI(3)%P(1)%N=-Alfa;     mEL4%SI(3)%P(1)%E=1;
      mEL4%SI(3)%P(2)%N=Alfa;      mEL4%SI(3)%P(2)%E=1;
      mEL4%SI(4)%P(1)%N=1;         mEL4%SI(4)%P(1)%E=Alfa;
      mEL4%SI(4)%P(2)%N=1;         mEL4%SI(4)%P(2)%E=-Alfa;

      do i = 1, 4
        DO iP = 1, 2
                e = mEL4%SI(i)%P(iP)%E
                n = mEL4%SI(i)%P(iP)%N
                mEL4%SI(i)%N1(1,iP) = 0.25*(1-e)*(1-n);
                mEL4%SI(i)%N1(2,iP) = 0.25*(1+e)*(1-n);
                mEL4%SI(i)%N1(3,iP) = 0.25*(1+e)*(1+n);
                mEL4%SI(i)%N1(4,iP) = 0.25*(1-e)*(1+n);
                mEL4%SI(i)%N2(1,iP) = -0.25*(1-e);
                mEL4%SI(i)%N2(2,iP) = -0.25*(1+e);
                mEL4%SI(i)%N2(3,iP) = 0.25*(1+e);
                mEL4%SI(i)%N2(4,iP) = 0.25*(1-e);
        END DO;
      end do;
    END SUBROUTINE InitL4;
```

Rys. 9.5. Inicjalizacja danych dla czworościennego elementu.

9.3.5. PROCEDURA WCZYTYWANIA DANYCH

Program wczytywania danych jest pokazany na rys. 9.6. On otwiera tekstowy plik indata.t2d, strukturę jakiego opisano w 9.2.

```
!*****************************************************************
! Wczytywanie danych
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      Subroutine InpData();
      use GlobData;
      implicit none;
      integer nFile,i,j,ost;

        nFile = 89
        OPEN (nFile, FILE='indata.t2d', action='read')
        read(nFile, *);
        read(nFile, *);
        read(nFile, *);
        read(nFile, *);
        read(nFile, *) mTbegin;
        read(nFile, *) mTime;
        read(nFile, *) mdTime;
        read(nFile, *) mT_otoczenia;
        read(nFile, *) mAlfa;
        read(nFile, *) mHD;
        read(nFile, *) mBD;
        read(nFile, *) mNbH;
        read(nFile, *) mNbB;
        read(nFile, *) mC;
        read(nFile, *) mK;
        read(nFile, *) mR;
        close (nFile);
      end Subroutine InpData
```

Rys. 9.6 Program wczytywania danych.

9.3.6. PROCEDURA PRZYDZIELENIA PAMIĘCI DLA TABLIC

Przydzielenie pamięci dla tablic obliczą szerokość mLDA matrycy sztywności za pomocą mapy elementów i przydziela pamięć zmiennym mA (globalna matrycy sztywności), mX (wektor niewiadomych), mB (wektor obciążeń). W celach metodycznych pokazano, jak przydzielić pamięć dla pełnej matrycy o wymiarach nh*nh. Taki wariant, jednak, powoduje zbyt nieekonomiczne wykorzystanie pamięci.

```
      SUBROUTINE ALLOCATE_Matrix;
      use GlobData;
      implicit none;
      integer*4 : NEL, i, j, k, jj, jB, nk(4),NdMaxB, ierr;

      implicit none;
      integer*4 :: NEL,
      integer*4, dimension(4)  :: nk;
      integer*4 :: jB,jB,NdMaxB,NCODA,i,j,k,jj;
      integer*4 :: iErr;

        mA=0;
        mB=0;
        mX=0;
        do NEL=1, mGr%ne
                do i=1, mEL4%nbn
                        nk(i) = mGr%EL(NEL)%nop(i);
                end do;

                CALL FeSM_heat( NEL );

                do i=1, mEL4%nbn
                        ii = nk(i);                     ! wiersz w pełnej matrycze
                        do j=1, mEL4%nbn
                                jj = nk(j);             ! Kolumna w pełnej matrycze
                                jB = mLDA + ii - jj;    ! Wiersz w pasmowej matrycze
                                if (jj>=ii) AND( jB>=mLDA) then
        mA(jB, jj) = mA(jB, jj) + est(i,j);             ! Wypełnienie pasmowej matrycy
                                end if;
!       mA(ii,jj) = mA(ii,jj) + est(i,j);               ! Wypełnienie pełnej matrycy
                        end do;
                        mB(ii) = mB(ii) + r(i);
                end do;
        end do;

        NCODA = mLDA-1;
        CALL DLSAQS (mGr%nh, mA, mLDA, NCODA, mB, mX)
! Program rozwiązania układu równań z pasmowej matrycy
!       NCODA=1;
!       CALL DLSARG (mGr%nh, mA, mGr%nh, mB, NCODA, mX)
! Program rozwiązania układu równań z pełnej matrycą
        do i=1, mGr%nh
                mGr%nd(i)%CR = ( mGr%nd(i)%t-mX(i) ) / mdTime;
                mGr%nd(i)%t = mX(i);
        end do;
      END SUBROUTINE SOLVER;
```

Rys. 9.8. Program wypełniania globalnej matrycy i rozwiązania układu równań MES.

9.3.8. PROCEDURA WYPEŁNIENIA MACIERZY I WEKTORA OBCIĄŻEŃ ELEMENTU

Program FeSM_heat (rys. 9.9) zawiera dwie części:

PRE_heat_mat – wypełnia matrycę i wektor obciążeń;

PRE_heat_pov_mat – uwzględnia warunki brzegowe.


```
      mLDA=0;
      do NEL = 1,mGr%ne
                do i=1,mEL4%nbn
                        nk(i) = mGr%EL(NEL)%nop(i);
                end do;
                do i=1,mEL4%nbn
                ! ii - wiersz "
                        ii=nk(i);
                        do j=1,mEL4%nbn
                        ! jj - kolumna "
                                jj=nk(j);
                                jB = jj-ii+1;
                                if(jB>=mLDA) then
                                mLDA=jB;
                                NdMaxB = NEL;
                                end if;
                        end do;
                end do;
      end do;

        ALLOCATE ( mA(mLDA, mGr%nh), STAT=iEr );
!       ALLOCATE ( mA(mGr%nh, mGr%nh), STAT=iEr );
!       Pełna matrycy MES
        ALLOCATE ( mB(mGr%nh), STAT=iEr );
        ALLOCATE ( mX(mGr%nh), STAT=iEr );
      END SUBROUTINE ALLOCATE_Matrix;
```

Rys. 9.7. Program do obliczenia potrzebnej pamięci dla globalnej matrycy sztywności.

9.3.7. PROCEDURA ROZWIĄZYWANIA ZADANIA DLA BIEŻĄCEGO KROKU CZASOWEGO

Program do rozwiązywania zadania dla bieżącego kroku czasowego jest pokazany na rys. 9.8. W tym programie oznaczenia zmiennych są następujący:

NEL – bieżący numer elementu;

Nk – tablica, zawierająca mapę bieżącego elementu;

FeSM_heat – program wypełniania matrycy i wektora obciążeń bieżącego elementu NEL;

DLSAQS – program rozwiązania układu równań z pasmową matrycą;

DLSARG – program rozwiązania układu równań z pełną matrycą (podano jako komentarz w celach metodycznych).

```
!*****************************************************************
!                       Solver
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      SUBROUTINE SOLVER( );
      use GlobData;
      use mamraz;

!*****************************************************************
! Generacja matryc elementu NEL
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      SUBROUTINE FeSM_heat( NEL );
      use GlobData;
      implicit none;
      integer*4 :: NEL;       ! Numer bieżącego Elementu
        est = 0;
        r = 0;
        call PRE_heat_mat( NEL );
        call PRE_heat_pov_mat(NEL );
      END SUBROUTINE FeSM_heat;
```

Rys. 9.9. Program do obliczenia matrycy sztywności bieżącego elementu NEL.

9.3.9. PROCEDURA WYPEŁNIENIA MACIERZY I WEKTORA OBCIĄŻEŃ

Program jest podany na rys. 9.10. Wywołuje program Jacob_2d, opisany w rozdziale 3.6 i podany w postaci uproszczonej na rys. 3.26. Program realizuje typowy algorytm wypełniania matrycy jednego elementu, wielokrotnie opisany w danym opracowaniu. W takie programu podane są odwołania na formuły, wykorzystywane podczas obliczeń elementów matrycy.

```
      subroutine PRE_heat_mat( NEL );
      use GlobData;
      implicit none;
      integer*4 :: NEL;
      INTEGER*4 :: i,j,N,P,Id;
      REAL*8, DIMENSION(2,2) :: J_-J_inv; ! Matrycа Jakobiego i odwrotna matrycа
      REAL*8          :: DetJ,Nl,Nn,Np,Cm;
      real*8          :: Tbp;
      REAL*8, DIMENSION(4)   :: Ndx,Ndy;  ! Pochodny funkcji kształtu względem X i Y
      REAL*8, DIMENSION(4)   :: X,Y,Temp_0;

        DO i=1,mEL4%NBN
                Id = ABS( mGr%EL(NEL)%NOP(i) );
                X(i)  = mGr%nd(Id)%x /1000.0;       ! [m]
                Y(i)  = mGr%nd(Id)%y /1000.0;       ! [m]
                Temp_0(i)= mGr%nd(Id)%t;
        END DO

        DO P=1, mEL4%N_p
CALL Jacob_2d(J_-J_inv, P, mEL4%N_p, mEL4%NBN, mEL4%N1,mEL4%N2, X, Y, DetJ);
                Tbp = 0;
                DO i=1, mEL4%NBN
                Ndx(i)=mEL4%N1(i,p)*J_inv(1,1) + mEL4%N2(i,p)*J_inv(1,2) ! [m-1]
                Ndy(i)=mEL4%N1(i,p)*J_inv(2,1) + mEL4%N2(i,p)*J_inv(2,2)! [m-1]
                Nl(p)=mEL4%N1(i,p);
                Tbp = Tbp + Temp_0(i)*Nl;
                END DO;

                DetJ = abs(detJ)*mEL4%W(p); ! [m2]
        DO N=1,mEL4%NBN
                DO I=1,mEL4%NBN
                        Nl = mEL4%N1(I,p);
                        Nn = mEL4%N1(N,p);

                Hn = mK*(Ndx(n)*Ndx(i) + Ndy(n)*Ndy(i))*DetJ;   ! Formuła (4.14)
                Cn = mC*mR*Nn*Nl*detJ;                          ! Formuła(5.3)
                        est(n,i) = est(n,i) + Hn + Cn/mdTime;   ! Formuła (5.8)
                        r(n)  =r(n) + (Cn/mdTime)*Tbp;
                END DO;
!               Q = 0.8*Tp*Hp
!               r(n)=r(n) + Nn*Q*DetJ;
        END DO;
      end do;
      end subroutine PRE_heat_mat;
```

Rys. 9.10. Program do obliczenia elementów matrycy sztywności bieżącego elementu NEL.

9.3.10. PROCEDURA Z WARUNKAMI BRZEGOWYMI

Program uwzględniający warunki brzegowe w matrycze sztywności elementu pokazano na rys. 9.11. W programie dla bieżącego elementu odbywa się pętla po powierzchniach kontaktu tego elementu z otoczeniem (ich liczba jest równa mGr%EL(NEL)%aPov). Dalej odbywa się całkowanie po każdej powierzchni za pomocą wybranego w programie InitL4 schematu całkowania. W takim programu podane są odwołania na formuły, wykorzystywane podczas obliczeń dodatków do elementów matrycy i wektora obciążeń.

```
!*****************************************************************
! WARUNKI BRZEGOWE:
! Milenin Andre, AGH, 2008, milenin@agh.edu.pl
!*****************************************************************
      subroutine PRE_heat_pov_mat( NEL );
      use GlobData;
      implicit none;
      integer*4          :: NEL;
      INTEGER*4          :: I,J,N,P,Id,iPov;
      REAL*8             :: DetJ,Nl,Nn,Np;
      REAL*8, DIMENSION(4)    :: X,Y;

        DO I=1,4
                Id = ABS(mGr%EL(NEL)%NOP(I));
                X(I)  = mGr%nd(Id)%x /1000.0;       ! [m]
                Y(I)  = mGr%nd(Id)%y /1000.0;       ! [m]
        END DO;

        DO iPov=1, mGr%EL(NEL)%aPov
        id = mGr%EL(NEL)%aPov(iPov); ! Lokalny numer powierzchni elementu
                SELECT CASE (id)
                case(1):
                DetJ=sqrt( (X(4)-X(1))**2 + (Y(4)-Y(1))**2 );
                case(2):
                DetJ=sqrt( (X(1)-X(2))**2 + (Y(1)-Y(2))**2 );
                case(3):
                DetJ=sqrt( (X(2)-X(3))**2 + (Y(2)-Y(3))**2 );
                case(4):
                DetJ=sqrt( (X(3)-X(4))**2 + (Y(3)-Y(4))**2 );
                END SELECT

                DO P=1, 2
                        DO n=1, 4
                ! Pętla po grupom ciemnem (wierszom matrycy)
                        DO i=1, 4
                ! Pętla po kolumnom matrycy
                        Np = mEL4%SI(id)%N1(n,p);
                        Nn = mEL4%SI(id)%N1(i,p);
                        est(n,i) = est(n,i) + mAlfa*Nn*Np*DetJ;         ! Formuła (4.14)
                        END DO
                Pn = mAlfa*mT_otoczenia*Np*DetJ;                        ! Formuła (4.15)
                        r(n) = r(n) + Pn;

                        END DO
                END DO ! P
        END DO; ! iPov
      end subroutine PRE_heat_pov_mat;
```

Rys. 9.11. Program uwzględniający warunki brzegowe w matrycze sztywności elementu.

9.4. PRZYKŁAD OBLICZEŃ

Wyniki obliczeń dla danych, zamieszczonych na rys. 9.1 są podane na rys. 9.12 dla centralnego punktu przekroju (krzywa 1) i na powierzchni (krzywa 2).