Cyprian T. Lachowicz

Freefem++
Bardzo krótkie wprowadzenie[1]

Opole 17 kwietnia 2011

[1] Praca poniższa jest to swobodna wariacją na temat freefem++. Napisana została na podstawie swobodnych tłumaczeń (bardzo skróconych, z niewielkim udziałem własnym) podręczników do Freefem++ autorstwa F. Hechta, O. Pironneau, A. Le Hyaric i K. Ohtsuka oraz innych materiałów znalezionych w Internecie. Pojawiają się tu również własne przemyślenia autora

Spis treści

Rozdział 1

Podstawy

1.1. Jak zacząć

Poniżej pokażemy jak rozwiązać problem Poissona[1] wykorzystując metodę elementów skończonych. Dla danej funkcji f(x,y) poszukujemy funkcji u(x,y) spełniającej równanie

$$\begin{aligned} -\triangle u(x,y) &= f(x,y), \\ (x,y) \in \Omega,\ \triangle u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \\ u(x,y) &= 0\ \ dla\ \ (x,y) \in \partial\Omega. \end{aligned} \tag{1.1.1}$$

gdzie: Ω – powierzchnia na której analizujemy równanie
$\partial\Omega = C$ – brzeg ograniczający powierzchnię Ω.
Załóżmy, że brzeg zadany jest funkcją parametryczną

$$C = (x,y) \rightarrow\ x = \cos(t)\ , y = \sin(t), 0 \leq t \leq 2\pi$$

Program rozwiązujący równanie 1.1.1 ma postać

```
 1 : border C(t=0,2*pi){x=cos(t); y=sin(t);}
 2 : mesh Th = buildmesh (C(50));
 3 : fespace Vh(Th,P2);
 4 : func f= x*y;
 5 : Vh u,v;
 6 : problem Poisson(u,v,solver=LU) =
 7 : int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v))
 8 : - int2d(Th)( f*v )
 9 : + on(C,u=0) ;
10 : real cpu=clock();
11 : Poisson;
12 : plot(u);
13 : cout << " CPU = " << clock()-cpu << endl;
```

Numery linii oraz „:” nie są częścią kodu i napisane zostały tylko dla ułatwienia jego analizy.

Linia pierwsza Polecenie `border` umożliwia analityczną definicję brzegu `C` w postaci parametrycznie zadanego okręgu. O tym, że zdefiniowano koło, a nie otwór w powierzchni, decyduje sposób zadania parametru `t=0,2*pi`. Jeśli definiujemy `t=2*pi,0` to otrzymujemy otwór. Brzeg w naszym przykładzie nazywa się „C”. Wycięcie eliptycznego otworu z powierzchni Ω wymagało by zdefiniowania po linii nr. 1 polecenia
`border D(t=2*pi,0){x=0.1+0.3*cos(t); y=0.5*sin(t);}`
oraz przebudowy linii 2 do postaci:
`mesh Th = buildmesh (C(50)+D(50));`

Linia druga Podział powierzchni Ω na trójkąty odbywa się poleceniem `buildmesh(C(50))`, gdzie liczba `50` oznacza planowaną liczbę trójkątnych elementów skończonych.

[1] Rozdział jest swobodnym tłumaczeniem (bardzo skróconym, z niewielkim udziałem własnym) rozdziału 2 podręcznika Freefem++ F. Hechta, O. Pironneau, A. Le Hyaric i K. Ohtsuka.

Linia trzecia Słowo kluczowe `fespace Vh(Th,P2)` wraz ze swoimi parametrami `Th, P2` generuje właściwą siatkę elementów skończonych. `Th` oznacza tu wygenerowane wcześniej trójkąty, a P2 oznacza, że funkcja kształtu przyjętego izoparametrycznego elementu skończonego jest wielomianem stopnia drugiego.

Linia czwarta Definicja funkcji `f(x,y)` w postaci `f=x*y`

Linie piąta Definiowanie w przestrzeni `Vh` funkcji „u" stanowiącej rozwiązanie problemu 1.1.1. Dodatkowo definiowana jest pewna funkcja kontrolna „v" niezbędna przy wariacyjnym sformułowaniu problemu 1.1.1

Linie od szóstej do dziewiątej Wariacyjne sformułowanie równania Poissona. Mnożąc równanie (1.1.1) przez funkcję kontrolną v(x,y) i całkując po powierzchni , , Ω'' otrzymujemy

$$-\int_\Omega v\triangle u(x,y)\,dx\,dy = \int_\Omega vf\,dx\,dy \tag{1.1.2}$$

zgodnie z twierdzeniem Greena równanie Poissona można zapisać w postaci

$$a(u,v) - \ell(f,v) = 0 \;\; dla \;\; wszystkich \;\; v \tag{1.1.3}$$
$$a(u,v) = \textstyle\int_\Omega \nabla u\nabla v\,dx\,dy \tag{1.1.4}$$
$$\ell(f,u) = \textstyle\int_\Omega fv\,dx\,dy, \tag{1.1.5}$$
$$v = 0 \;\; dla \;\; \partial\Omega \tag{1.1.6}$$

Równanie (1.1.4) można zapisać przy użyciu poleceń `freefem++`

$$\int_\Omega \nabla u\nabla v\,dx\,dy \rightarrow \;\; \texttt{int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v))}$$

zaś równanie (1.1.5)

$$\int_\Omega fv\,dx\,dy \rightarrow \texttt{int2d(Th)(f*v)}$$

i wreszcie warunek brzegowy $u(x,y) = 0 \;\; dla \;\; (x,y) \in \partial\Omega$ zapisujemy

```
on(C,u=0);
```

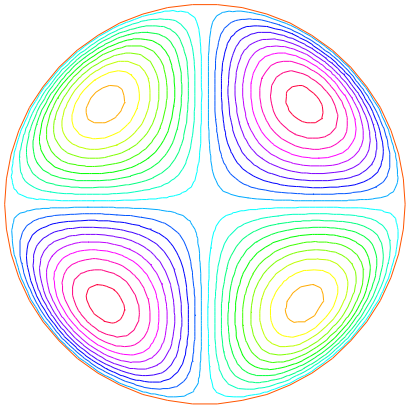
W przypadku kiedy powierzchnia Ω ma otwór D należy określić warunek brzegowy na jego krawędzi, polega to na modyfikacji polecenia `on(...)`

```
+ on(C,u=0) + on(D,u=0);
```

Linia dziesiąta Ustawiamy zegar mierzący czas obliczeń.

linia jedenasta Formalnie uruchamiamy obliczenia równania Poisson.

Linia dwunasta Wykreślamy funkcję `u` na tle powierzchni Ω.



Rysunek 1.1. Isolinie funkcji u spełniającej równanie 1.1.1.

Rozdział 2

Sformułowania wariacyjne słabe podstawowych równań fizyki matematycznej

2.1.

Rozdział 3

Przykłady zagadnień

3.1. Zadanie Lamé

Wyznaczyć rozkład przemieszczeń i naprężeń dla płaskownika z otworem centralnym obciążonego wzdłuż dwu krawędzi poziomych. Z uwagi na symetrię modelu analizie można poddać tylko jedną czwartą modelu (rys.3.1), zastępując oddziaływanie drugiej części przez odpowiednio zbudowane warunki brzegowe. Zagadnienie Lamé wynikające z uogólnionego prawa Hooke'a

$$\sigma_{ij}(\mathbf{x}) = c_{ijkl}(\mathbf{x})\varepsilon_{ij}(\mathbf{x}) \tag{3.1.1}$$

oraz definicji tensora odkształcenia

$$\varepsilon_{ij}(u) = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{3.1.2}$$

gdzie: i,j – x,y - współrzędne kartezjańskie,
symbol **,** – oznacza różniczkowanie po x lub y.
Zakładając materiał izotropowy i wprowadzając dwie stałe materiałowe λ, μ równanie Lame przyjmuje postać

$$\sigma_{ij} = \lambda\delta_i j \,\mathtt{div}u + 2\mu\varepsilon_{ij} \tag{3.1.3}$$

gdzie: div(u) – dywergencja funkcji u
δ_{ij} – delta Kroneckera.
Sformułowanie wariacyjne równania 3.1.3 przybiera postać

$$\int_\Omega [2\mu\varepsilon_{ij}(\mathbf{u})\varepsilon_{ij}(\mathbf{v}) + \lambda\varepsilon_{ii}(\mathbf{u})\varepsilon_{jj}(\mathbf{v})] = \int_\Omega \mathbf{f}\cdot\mathbf{v} + \int_\Gamma \mathbf{g}\cdot\mathbf{v},\ \forall\mathbf{v}\in V \tag{3.1.4}$$

gdzie: V – jest podprzestrzenią liniową przestrzeni Ω^2,
i jest zapisane w liniach 25, 26, 27; linie 28, 29, 30 opisują warunki brzegowe wynikające z symetrii modelu oraz uwzględniające kinematyczny sposób obciążenia górnej krawędzi. W zadaniu uwzględniono również grawitację.

```
// Zadanie lamego
// CTL @ 2005
int topplate=5;
int bottomplate=3;
int right=1;
int left=4;
int circle=2;
int gravity=0;
```

Definiowanie zarysu próbki przy użyciu polecenia `border()`

```
border a(t=0,1.8){x=0;y=2-t;label=right;};//prawa
border b(t=pi/2,0){x=0.2*cos(t);y=0.2*sin(t);label=circle;};
border c(t=0,1.8){x=0.2+t;y=0;label=bottomplate;};
border d(t=0,2){x=2;y=t;label=left;};
border e(t=0,2){x=2-t;y=2;label=topplate;};
```

Definiowanie siatki elementów skończonych i jej kreślenie

```
mesh th= buildmesh(a(20)+b(40)+c(20)+d(30)+e(30));
plot(th);
```

Definiowanie stałych niezbędnych dla zadania sprężystego

```
real E=1;
real nu=0.3;
real mu=E/(1+nu);
real lambda=E*nu/((1+nu)*(1-2*nu));
```

Definowanie właściwości siatki elementów skończonych. Rozwiązywanie zagadnienia wariacyjnego.

```
fespace Vh(th,[P1,P1]);
Vh [uu,vv],[w,s];
solve bb([uu,vv],[w,s])=
int2d(th)(2*mu*(dx(uu)*dx(w)+((dx(vv)+dy(uu))*(dx(s)+dy(w)))/4)
          +lambda*(dx(uu)+dy(vv))*(dx(w)+dy(s))/2)
          +int2d(th)(gravity*s)
 +on(left,uu=0)
 +on(topplate,vv=0.1)
 +on(bottomplate,vv=0);
```

Wykreślanie przemieszczeń i zapisywanie obrazków w pliku.

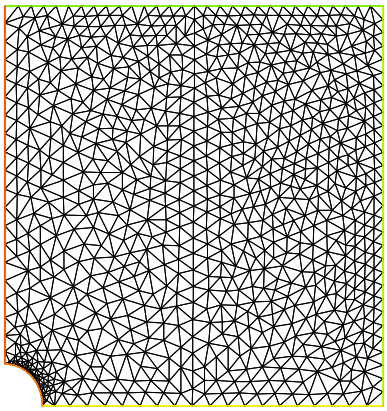
```
plot([uu,vv],wait=1);
mesh th1=movemesh(th,[x+uu,y+vv]);
plot(th1,wait=1,ps="th1.eps");
plot(th,uu,value=true,ps="uu.eps");
plot(th,vv,value=true,fill=1,ps="vv.eps");
```

Obliczenia naprężeń w płaskim stanie naprężenia

```
fespace zVh(th,[P2]);
zVh sx,sy,sxy;
sx=lambda*(dx(uu)+dy(vv))+2*mu*dx(uu);
sy=lambda*(dx(uu)+dy(vv))+2*mu*dy(vv);
sxy=mu*(dy(uu)+dx(vv));
```

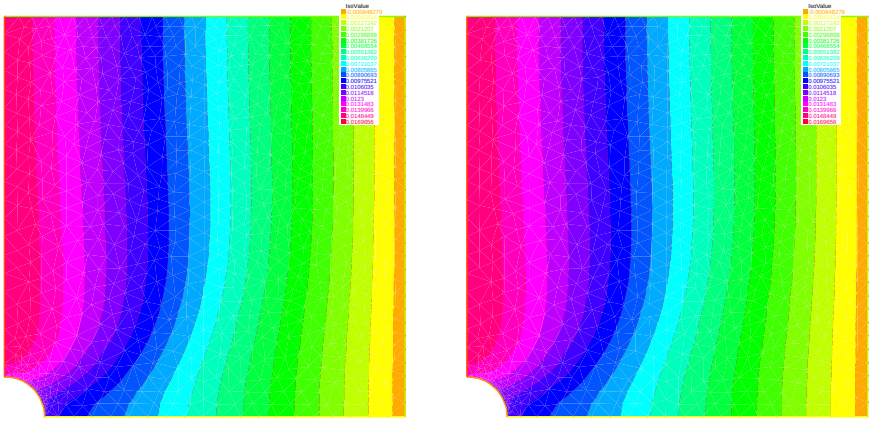
Wykreślanie map naprężeń i zapis ich do pliku

```
plot(sx,value=true,wait=1,fill=1,ps="sxx.eps");
plot(sy,value=true,wait=1,fill=1,ps="syy.eps");
plot(sxy,value=true,wait=1,fill=1,ps="sxy.eps");
```

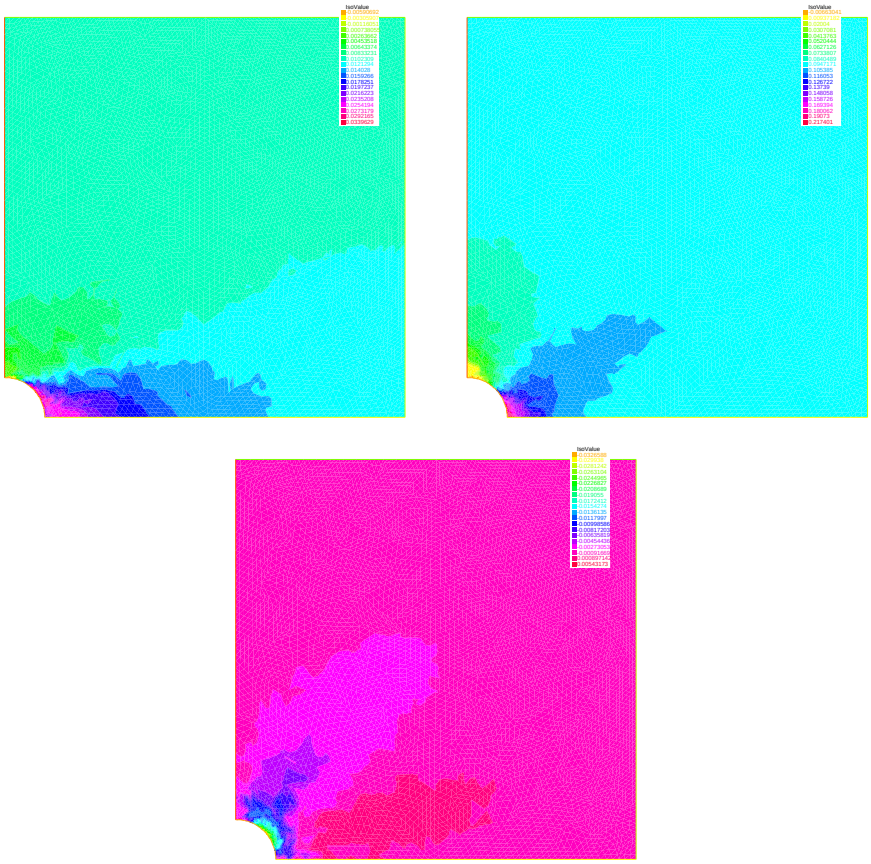


Rysunek 3.1. Siatka elementów skończonych rozpięta na jednej czwartej modelu płaskownika z otworem centralnym

Uzyskane mapy przemieszczeń modelu pokazano na rysunku 3.2. Uzyskane mapy naprężeń pokazano na rys.3.3.



Rysunek 3.2. Mapy przemieszczeń



Rysunek 3.3. Mapy naprężeń $\sigma_{xx}, \sigma_{yy}, \sigma_{xy}$

3.1.1. Duży przykład

```
// Freefem ++ płyta kwadratowa -
// Przykład zbudowano na bazie anonimowego tekstu w języku francuskim
// ===============
// ctl 2007.01.23
// ===============
//
   string nazwa,ext,trian;
   ext=".dat"; // plik pod excela
   trian=".msh"; //plik z siatką z frefem++
   nazwa="plyta_wzor_";  // nazwa pliku
//
//
// Definicja geometrii
// ====================
// Definicja punktów konturu P(x,y) : x=Px i y=Py
// --------------------------------------
//
//                          gora=4
//    (A[3],B[3]) o----------------------o (A[2],B[2])
//                |                      |
//                |                      |
//         lewy=1 |                      | prawy=3; wyskosc
//                |                      |
//                |                      |
//                |                      |
//    (A[0],B[0]) o----------------------o (A[1],B[1])
//                          dol=2; szerokosc
//
   int dlugosc=10, wysokosc=10;
   real[int] A(4), B(4);
   A[0]=0  ; B[0]=0  ;
   A[1]=dlugosc ; B[1]=0  ;
   A[2]=dlugosc ; B[2]=wysokosc ;
   A[3]=0  ; B[3]=wysokosc ;

// Definicja konturu
// --------------------------------------
   int lewy=1;  // Numery brzegów konturu
   int dolny=2; //
   int prawy=3; //
   int gorny=4; //

// Definicja brzegu okreslona funkcjami parametrycznymi x=f(t), y=g(t)
// definicja zgodna z obiegiem w lewo

   border left(t=wysokosc,0) {x=0; y=t;label=lewy;}; //
   border bottom(t=0,dlugosc) {x=t; y=0;label=dolny;}; //
   border right(t=0,wysokosc) {x=dlugosc; y=t;label=prawy;};
   border top(t=dlugosc,0) {x=t; y=wysokosc;label=gorny;};

// Generacja siatki
// ================
// Definicja podziału krawędzi
```

```
// ------------------------------------------------
   int NpSzerokosc=10 ; // liczba podziałow krawędzi
   int NpDlugosc=10 ;   //
//
// Triangularyzacja
// ----------------------------------------
   mesh siatka=buildmesh(left(NpSzerokosc)+top(NpDlugosc)+right(NpSzerokosc)+bottom(NpDlugosc));

// Kreślenie trójkątów
   plot(siatka,wait=true);
//
// Definicja  siatki elemetów skończonych oraz jej własciwości
// wykorzystano element typu P1
// ------------------------------------
   fespace Dskalar(siatka,P2);      // funkcja skalarna w każdym węźle
   fespace Dwektor(siatka,[P2,P2]); // funkcja wektorowa w każdym węźle

// Kreślenie, zapamiętywanie siatki elementów skończonych
// ------------------------------
   plot(siatka,wait=true, cmm="siatka poczatkowa");
   savemesh(siatka,nazwa+"\_siatka"+trian);

// Wazne komunikaty
// --------------------
   cout << " " << "\n";
   cout << "Oblicznia w toku"<< "\n";
   cout << " " << "\n";

// Definiowanie charakterystyk materiałowych
// =======================================
//
// --------------------------------------
   real Young= 210000;              // Moduł Young'a
   real Poisson= 0.3 ;              // Współczynnik  Poissona
   real Mu = Young/(2*(1+Poisson)); // Współczynnik Lamé (Mu)
   real Lambda = Young*Poisson/((1+Poisson)*(1-2*Poisson)); // Współczynnik Lamé (lambda)

// Obciążenia zewnętrzne
// Siły objętościowe
// ----------------
// Pole (obciążeń) naprężeń zewnętrznych

// --------------------
   real SYY=0;
   real SXX=100;
   real SXY=0;
   real SYX=SXY;

// definicja problemu
// ================================================================================
// Płaski stan naprężenia (wybor=0) lub odkształcenia (wybor=1)
// ------------------------------------------------------
//    epsilon33=-Lambda/(Lambda+2*Mu)*(epsilon11+epsilon22); // plaskie naprężenia
//    epsilon33=0; // płaskie odkształcenia
//
```

```
   int wybor=0;
   real gwiazda;
   cout << " " << "\n";
   if (wybor==0)
     {
  cout << "Plaski stan naprezenia " << "\n";
      gwiazda=-Lambda/(Lambda+2*Mu);
     }
    else
     {
  cout << "Plaski stan odksztalcenia" << "\n";
      gwiazda=0;
      cout << " " << "\n";
     }
//=========================================================================
// Równanie Lame (sformułowanie wariacyjne słabe) Płaski stan naprężenia i odkształcenia
// ----------------
// int2d(maillage){(Mu*[dx(uu)*dx(u)+dy(vv)*dy(v)+(dx(vv)+dy(uu))*(dx(v)+dy(u))/2]
// +(1+Etoile)*Lambda*[dx(uu)+dy(vv)]*[dx(u)+dy(v)]/2)*2}
// Warunki brzegowe - w przemieszczeniach ( warunek Dirichleta)
// ------------------------------------
// on(NoDE,uu=0) // pierwszy fragment brzegu (uu=0)
// on(NoAB,vv=0) // pierwszy fragment brzegu )vv=0)
//.../

// obciążenie  brzegowe siłami,
// N(N.x, N.y), wektory normalne do brzegu
// -------------------------------
// int1d(siatka,lewy)((sigmaXX*N.x+sigmaXY*N.y)*u
//+(sigmaXY*N.x+sigmaYY*N.y)*v) // obciążenie na brzegu lewy
// int1d(siatka,dolny)((sigmaXX*N.x+sigmaXY*N.y)*u
//+(sigmaXY*N.x+sigmaYY*N.y)*v) // obciążenie na brzegu dolny
//.../

// Rozwiązanie zagadnienia wariacyjnego w przemieszczeniach

// ================================================
// Definicja pola przemieszczeń [uu,vv] i zmiennych pomocniczych [u,v]
// ---------------------------------------------------------------------------------
   Dwektor [uu,vv], [u,v]; //funkcja wektorowa przemieszczeń  rozpięta na siatce

// ======================
// Rozwiązanie w przemieszczeniach
// -------------------------
   solve deplacement([uu,vv],[u,v]) =
    int2d(siatka)((Mu*(dx(uu)*dx(u)+dy(vv)*dy(v)+(dx(vv)+dy(uu))*(dx(v)+dy(u))/2)
    +(1+gwiazda)*Lambda*(dx(uu)+dy(vv))*(dx(u)+dy(v))/2)*2)
    -int1d(siatka,lewy)((SXX*N.x+SXY*N.y)*u
    +(SXY*N.x+SYY*N.y)*v)
    + on(prawy,uu=0)
    + on(prawy,vv=0);

//
// Prezentacja wyników
// Obliczania modułu przemieszczenia całkowitego dla każdego węzła siatki
```

```
// ----------------------------------------------------
   Dskalar module=sqrt(uu^2+vv^2);

// Rozwiązanie w odkształceniach i naprężeniach
// ============================
// Obliczanie składowych tensora odkształcenia
// ------------------------------------------------
   Dskalar EpsilonXX=dx(uu);
   Dskalar EpsilonYY=dy(vv);
   Dskalar EpsilonXY=(dy(uu)+dx(vv))/2;


// Płaski stan naprężenia lub odkształcenia
// zmiena gwiazda oznacza wybór
// ------------------------------------
   Dskalar EpsilonZZ=gwiazda*(EpsilonXX+EpsilonYY);
// Calcul des composantes du tenseur des contraintes
// ------------------------------------------------------------
   Dskalar SigmaXX=Lambda*(1+gwiazda)*(EpsilonXX+EpsilonYY)+2*Mu*EpsilonXX;
   Dskalar SigmaYY=Lambda*(1+gwiazda)*(EpsilonXX+EpsilonYY)+2*Mu*EpsilonYY;
   Dskalar SigmaXY=2*Mu*EpsilonXY;
   Dskalar SigmaZZ=Lambda*gwiazda*(EpsilonXX+EpsilonYY);


// Obliczanie składowych tensora naprężenia
// ---------------------------------
   Dskalar centre=(SigmaXX+SigmaYY)/2;
   Dskalar rayon=sqrt((SigmaXX-SigmaYY)^2+4*SigmaXY^2)/2;
   Dskalar sigmaI=centre+rayon;
   Dskalar sigmaII=centre-rayon;

// Obliczenia kieruków osi głownych
// -------------------------------
// Kat
   Dskalar alpharad=atan(2*SigmaXY/(SigmaXX+SigmaYY))/2;
   Dskalar alphadeg=alpharad*180/pi;

// Wektory własne naprężeń
   Dwektor [sigmaIx ,sigmaIy] , [sigmaIIx ,sigmaIIy];
   [sigmaIx ,sigmaIy]=[sigmaI*cos(alpharad),sigmaI*sin(alpharad)];
   [sigmaIIx ,sigmaIIy]=[-sigmaII*sin(alpharad),sigmaII*cos(alpharad)];

// Prezentacja wyników
// ********************
// siatka
// =========
// Kreślenie siatki po deformacji
// ---------------------------------
   mesh deforme = movemesh(siatka, [x+uu, y+vv]);
   plot(deforme,wait=true,ps=nazwa+"siatka_zdeform.eps",cmm="siatka_zdeformowana");

// Kreślenie siatki niezdeformowanej i zdeformowanej
// -------------------------
   plot(siatka,deforme,wait=true,cmm="siatka poczatkowa i zdeformowana");
```

```
// Pole przemieszczeń
// ==================
// Pole przemieszczeń wektorowo
// ----------------------------------------
   plot([uu,vv],value=true,wait=true,cmm="Pole przemieszczen");

// Pole przemieszczeń uu
// ----------------------------------------
   plot(uu,value=true,wait=true, cmm="Przemieszczenie u");

// Pole przemieszczeń vv
// ----------------------------------------
   plot(vv,value=true,wait=true,cmm="Przemieszczenie v");

// Pole przemieszczeń moduł
// ----------------------------------------
   plot(module,value=true,wait=true,cmm="Przemieszczenie całkowite (modul)");

// Pole przemieszczeń uu i vv
// ----------------------------------
   plot(uu,vv,value=true,wait=true,cmm="Przemieszczenie u i v");


// Odkształcenia
// =============
// Kreślenie epsilonXX
// ------------------------------
   plot(EpsilonXX,value=true,wait=true,cmm="EpsilonXX");

// Kreślenie epsilonYY
// ------------------------------
   plot(EpsilonYY,value=true,wait=true,cmm="EpsilonYY");

// Kreślenie epsilonXY
// ------------------------------
   plot(EpsilonXY,value=true,wait=true,cmm="EpsilonXY");


// Naprężenia
// ===========
// Kreślenie sigmaXX
// ----------------------------
   plot(SigmaXX,value=true,wait=true,fill=1,ps=nazwa+"sigmaXX.eps",cmm="SigmaXX");

// Kreślenie sigmaYY
// ----------------------------
   plot(SigmaYY,value=true,wait=true,fill=1,ps=nazwa+"sigmaYY.eps",cmm="SigmaYY");
// Kreślenie sigmaXY
// ----------------------------
   plot(SigmaXY,value=true,wait=true,fill=1,ps=nazwa+"sigmaXY.eps",cmm="SigmaXY");

// Kreślenie naprężnie głównego sigma I
// ----------------------------------------
   plot(sigmaI,value=true,wait=true,fill=1,ps=nazwa+"sigmaI.eps",cmm="SigmaI");
```

```
//Kreślenie naprężnie głównego sigma I
// ---------------------------------------
   plot(sigmaII,value=true,wait=true,fill=1,ps=nazwa+"sigmaII.eps",cmm="SigmaII");

// Isoclines (isovaleurs de alpha)
// ---------
   plot(alphadeg, value=true,wait=true,fill=0,ps=nazwa+"Isoclines.eps",cmm="isolinie stopni");

// Isochromes (isovaleurs de cisaillement max)
// ----------------------------------
   plot(rayon,value=true,wait=true,ps=nazwa+"Isochromes.eps",cmm="Isochromes");

// Isopaches (isovaleurs de sigma moyen)
// --------------------------------
   plot(centre,value=true,wait=true,fill=1,ps=nazwa+"Isopaches.eps",cmm="Isopaches");

// Isostatiques
// ------------
   plot([sigmaIx,sigmaIy],value=true,wait=true,cmm="Isostatiques Sigma I");
   plot([sigmaIIx,sigmaIIy],value=true,wait=true,cmm="Isostatiques Sigma II");
   plot([sigmaIx,sigmaIy],[sigmaIIx,sigmaIIy],value=true,wait=true,cmm="Isostatiques");


// Coupes
// // ======
// // bord AB
// // -----------------
// real[int] xAB(NpAB+1),zAB22(NpAB+1),zAB11(NpAB+1),zAB12(NpAB+1);
// int i;
// for (i=0;i<NpAB+1;i++)
// {
// x=A[0]+(B[0]-A[0])/NpAB*i;y=A[1]; // sens A->B
// xAB[i]=x;
// zAB11[i]=sigma11;
// zAB22[i]=sigma22;
// zAB12[i]=sigma12;
// }
// cout << " " << "\n";
// cout << " Coupe AB de A vers B" << "\n";
// cout << " --------------------" << "\n";
// cout << "SigmaXX sur AB - en A " << zAB11[0] << "\n";
// cout << "SigmaXX sur AB - en B " << zAB11[NpAB] << "\n";
// cout << "SigmaXX sur AB - max " << zAB11.max << "\n";
// plot([xAB,zAB11],wait=true,cmm="SigmaXX sur AB");
// {
// ofstream file(nom+"SigmaXX sur AB"+ext);
// for (i=0;i<NpAB+1;i++)
// {
// file << xAB(i) << "," << zAB11(i) << endl;
// }
// }c
// out << " " << "\n";
// cout << "SigmaYY sur AB - en A " << zAB22[0] << "\n";
// cout << "SigmaYY sur AB - en B " << zAB22[NpAB] << "\n";
```

```
// plot([xAB,zAB22],wait=true,cmm="SigmaYY sur AB");
// {
// ofstream file(nom+"SigmaYY sur AB"+ext);
// for (i=0;i<NpAB+1;i++)
// {
// file << xAB(i) << "," << zAB22(i) << endl;
// }
// }c
// out << " " << "\n";
// cout << "SigmaXY sur AB - en A " << zAB12[0] << "\n";
// cout << "SigmaXY sur AB - en B " << zAB12[NpAB] << "\n";
// plot([xAB,zAB12],wait=true,cmm="SigmaXY sur AB");
// ....
// // Rem plot fournit une fenętre vide si la sortie est nulle en tout point
```