Bengt Lindberg
051208

2D1240, Numeriska metoder gk2 för F2

Glesa matriser från randvärdesproblem

Vi skall i detta häfte bekanta oss med finita differensmetoder för linjära randvärdesproblem för ordinära och partiella differentialekvationer. Tonvikten ligger på Matlab-programmering, och specifikt representation och generering av de glesa[1] linjära ekvationssystem som erhålls.

1 Representation av gles matris i Matlab

I Matlab kan element a_{ij} i en gles matris A lagras som en tripel `(i,j,s)`, där `i` anger radindex, `j` kolumnindex, och `s` elementets numeriska värde. Hela matrisen lagras i tre lika långa vektorer `ivek,jvek,svek`, där vektorkomponenter med samma index ger en trippel som representerar ett matriselement.

Exempel 1
Matrisen

$$A = \begin{pmatrix} 13 & 0 & 0 & -10 \\ 0 & 17 & 0 & 0 \\ 14 & 0 & 0 & 0 \\ 0 & 0 & 12 & -15 \end{pmatrix}$$

representeras av de tre vektorerna ivek. jvek, svek med vardera sex komponenter.

```
index   ivek    jvek   svek

   1     1       1      13
   2     3       1      14
   3     2       2      17
   4     4       3      12
   5     1       4     -10
   6     4       4     -15
```

I Matlab kan en matris representeras på vanligt sätt som en tvådimensionell array eller som en gleslagrad matris. Det finns enkla kommandon för att växla mellan representationerna, se exemplet nedan.

Exempel 1, fortsättning

```
A=[13 0 0 -10
    0 17 0 0
   14 0  0 0
    0 0 12 -15];
```

[1] gles= sparse(eng.) betyder att koefficientmatrisen innehåller många nollor

```
As=sparse(A);  spy(As)

% samma matriser, genererade från vektorer
ivek=[1;3;2;4;1;4]; jvek=[1;1;2;3;4;4]; svek=[13;14;17;12;-10;-15];
Bs=sparse(ivek,jvek,svek);
B=full(Bs);
```

Kommandona `sparse, full` används bl.a. för konvertering mellan representationerna. Kommandot `spy` ger en grafisk representation av glesstrukturen (hur icke-nollorna ligger) för matrisen. Programsnutten finns i biblioteket och heter `gles1.m` .

2 Diskretisering av linjärt randvärdesproblem, ODE

Vi skall bestämma lösningen till randvärdesproblemet

$$y'' + a\sqrt{x}y' = (6-y)x, \quad y(0) = 1, y(4) = -1, 0 \leq x \leq 4$$

för $a = 2$. Problemet med lösning finns i NAM, sec. 8.7.3. För framtida bruk inför vi lite andra beteckningar och skriver problemet

$$y'' + p(x)y' = (6-y)x, \quad y(0) = 1, y(4) = -1, 0 \leq x \leq 4$$

med $p(x) = a\sqrt{x}$. Med hjälp av härledningen i NAM formulerar vi följande program, `randv1Dnaiv.m`

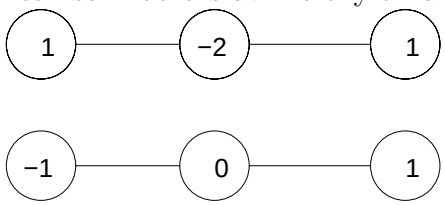
```
%Indatadel
N=19;

a=0; b=4;alfa=1; beta=-1;
p=inline('2*sqrt(x)','x');


%Förberedelser--definiera diskretiseringsnätet
h=(b-a)/(N+1);
xinre=[a+(1:N)*h];          %de inre diskretiseringspunkterna genereras
                            %    indiceras 1,2,  ... N i vektorn xinre
x=[a, xinre, b];    %alla punkterna lagras i vektorn x
                            %  de inre punkterna har index 2,3,--- N+1
                            %  i vektorn x

%-------------------------------------------------------------------
%Generera matrisen och högerledet
A=sparse(N);                %Anger att N x N matrisen A skall lagras
                            %    som en gles matris
b=zeros(N,1);
```

```
for i=1:N
  A(i,i)=-2/h^2+xinre(i);
  if i>1, A(i,i-1)=1/h^2-p(xinre(i))/(2*h); end
  if i<N, A(i,i+1)=1/h^2+p(xinre(i))/(2*h); end
  b(i)=6*xinre(i);
end
%-----------------------------------------------------------------

%korrigera första och sista komponenten av högerledsvektorn
%för randvärdena.
b(1)=b(1)-(1/h^2-p(xinre(1))/(2*h))*alfa;
b(N)=b(N)-(1/h^2+p(xinre(N))/(2*h))*beta;

%Lös ekvationssystemet och presentera lösningen
yinre=A\b;
y=[alfa; yinre; beta]';

plot(x,y)
[x((N+1)/2+1) y((N+1)/2+1) ]  %lösn. i mittpunkten
```

Koefficientmatrisen kan alternativt byggas upp från tre vektorer som får representera matriselementen, programdelen ovan "Generera matrisen och högerledet" ersätts av nedanstående rader, se programmet `randv1Dvekt.m`

```
%Generera matrisen och högerledet
%Vektororienterad generering av systemet
%Generera referenser och värden för matriselementen
i=1:N;       j=1:N;        s=-2/h^2+xinre;                    %diagonal
i=[i 2:N];   j=[j 1:N-1];  s=[s 1/h^2-p(xinre(2:N))/(2*h)];   %under
i=[i 1:N-1]; j=[j 2:N];    s=[s 1/h^2+p(xinre(1:N-1))/(2*h)]; %över

%skapa den gleslagrade matrisen från vektorerna ovan
A=sparse(i,j,s);
spy(A)

%skapa högerledet-måste vara kolumnvektor

b=6*xinre';
```

Koefficientmatrisen kan alternativt byggas genom att centrumpunkten i beräkningsmolekylerna nedan får svepa över de inre punkterna i diskretiseringsnätet och skapa matriselementen som berörs av molekylerna.



Den övre molekylen är en representation av h^2 gånger cetraldifferensapproximationen av andraderivatan. Den undre molekylen representerar $2h$ gånger en approximation till förstaderivatan. Se programmet `randv1Dgles.m` nedan.

En typisk ekvation ser ut enligt

$$\frac{y_H - 2y_C + y_V}{h^2} + p(x_C)\frac{y_H - y_V}{2h} + x_C y_C = 6x_C$$

Här står y för lösningen, index står för läget i molekylen; H = höger, C = centrumpunkten, V = vänster. Om molekylen berör en randpunkt flyttas motsvarande term över i högerledet, ty i randpunkterna är y känd.

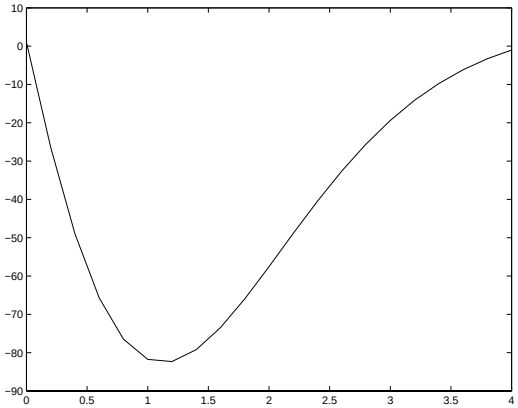
Detta förfaringssätt kan med fördel generaliseras till två- och tre-dimensionella randvärdesproblem, se senare delen av detta häfte. Genereringen av koefficientmatrisen sköts av programmet genom analys av närbelägna diskretiseringspunkter (lokalt), så hela matrisen ej behöver formuleras för hand (globalt).

```
%Indatadel
N=399;   a=0; b=4;alfa=1; beta=-1;   p=inline('2*sqrt(x)','x');

%Förberedelser--definiera diskretiseringsnätet. Detta
%förfaringssätt kan generaliseras till 2 och 3 dimensioner,
%se senare delen av häftet

h=(b-a)/(N+1);
x=[a, a+(1:N)*h, b];       %alla diskretiseringspunkterna lagras i vektorn x

G=(x>a & x<b);             %markera inre punkter i x-vektorn
inre=find(G);              %index i x-vektorn för inre punkter,radvektor
rand=find(G==0);           %index i x-vektorn för randpunkter,
                           %används ej i detta program
G=zeros(size(G));          %konvertera G från logical till double
G(inre)=(1:length(inre))'; %numrera de obekanta (i de inre punkterna)
                           %G har två uppgifter, skilja på randpunkter
                           %och inre punkter, samt numrera de
                           %obekanta. Randpunkter markeras med 0.
```

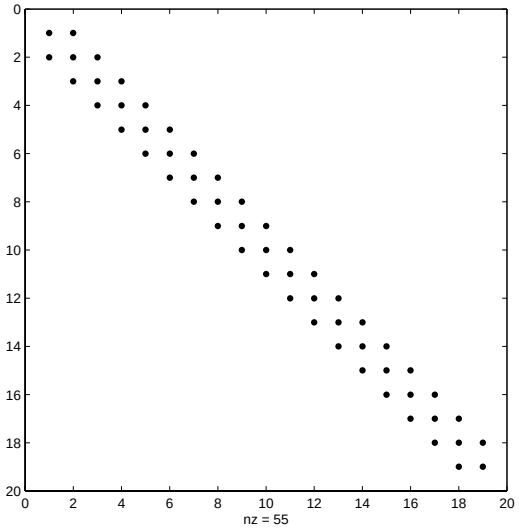
```
xi=x(inre);                %de inre x-värdena numreras på samma
                           %sätt som de obekanta
rad=[]; kol=[]; varde=[];

%loopa över alla de inre punkterna och konstuera triplar för
%de nollskilda matriselementen, rad efter rad.
for nr=inre
    %generera diagonalelementen
    i=G(nr); j=G(nr); s=-2/h^2+xi(G(nr));
    rad=[rad, i]; kol=[kol, j]; varde=[varde, s];

    %generera utomdiagonala elementen: bestäm index
    %för de två möjliga grannarna till en diskretiseringspunkt,
    %om diskretiseringspunkten har en randpunkt till granne
    %genereras inget matriselement för denna granne.
    for k= [-1 1]
      Q=G(nr+k);
      if Q ~= 0
        i=G(nr); j=Q; s=1/h^2+p(xi(G(nr)))*k/(2*h);
        rad=[rad, i]; kol=[kol, j]; varde=[varde, s];
      end
    end
end
A=sparse(rad,kol,varde);

%högerledet i ekvationssystemet, programmeras mer
%systematiskt för flerdimensionella problem

b=6*xi';
%första och sista komponenten av högerledet modifieras
%pga av randvillkoren.

b(1)=b(1)-(1/h^2-p(xi(1))/(2*h))*alfa;
b(N)=b(N)+(1/h^2+p(xi(N))/(2*h))*beta;

%Lös ekvationssystemet och presentera lösningen
yinre=A\b;  y=[alfa; yinre; beta]';

plot(x,y),    [x((N+1)/2+1) y((N+1)/2+1) ]
```

Programmet kan effektiviseras genom att eliminera foor-loopen och skapa vektorerna `i, j, s` gruppvis i stället. Programmet blir mer svårläst med den konstruktionen, så vi genomför den inte.

Om något av programmen exekveras med `N=19` erhålls nedanstående graf för den approximativa lösningen



Med kommandot `spy(A)` får vi följande bild av strukturen av icke-nollor (tridiagonal matris)



Med kommandot `full(A(1:5,1:5))` skriver vi ut den avsnittsmatris som består av de första fem raderna och kolumnerna av koefficientmatrisen `A`.

```
-49.8000   27.2361         0         0         0
 21.8377  -49.6000   28.1623         0         0
       0   21.1270  -49.4000   28.8730         0
       0         0   20.5279  -49.2000   29.4721
       0         0         0   20.0000  -49.0000
```

3 Diskretisering av Poissons ekvation, PDE

Poissons ekvation i två rumsdimensioner lyder

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

Ekvationen gäller i ett område Ω i (x, y)-planet. På randen $\partial\Omega$ är funktionen $u(x, y) = g(x, y)$ känd.

Exempel 2

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

med $f(x, y) = -2\pi^2 \sin \pi x \cos \pi y$. Området är kvadraten $\Omega = /-1 \leq x \leq 1, -1 \leq y \leq 1/$ och randvärdena ges av

$$g(x, y) = \begin{cases} 0 & \text{x=-1} \\ 0 & \text{x=1} \\ \sin \pi x & \text{y=-1} \\ \sin \pi x & \text{y=1} \end{cases}$$

3.1 Diskretisering av området

Området diskretiseras med ekvidistant steg $h = 2/(N + 1)$ i såväl $x-$led som $y-$led enligt

$$x_i = -1 + i \times h, i = 0, 1, 2, \ldots N + 1, \quad y_j = -1 + j \times h, j = 0, 1, 2, \ldots N + 1$$

Ett diskretiseringsnät med punkter (x_i, y_j) läggs över kvadraten. Följande programsnutt genererar två matriser `x` och `y` med x- och y-koordinaterna för alla diskretiseringspunkterna.

```
N=4;  hx=2/(N+1); hy=hx;
x = ones(N+2,1)*[-1, -1+(1:N)*hx, 1];
y = flipud(x'); %kasta om raderna i matrisen, sista blir första osv.

G = (x > -1) & (x < 1) & (y > -1) & (y < 1); %markera inre punkter

inre=find(G);           %returns the indices  that are
                        %non-zero, columnvector
                        % G is regarded as a vector
G = zeros(size(G));     % Convert from logical to double matrix

G(inre)=(1:length(inre))';     % Numbering of the interior nodes

Um=zeros(size(G));      %prepare solutionvector
```

De två matriserna blir med detta N-värde

```
x =

   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
   -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000
y =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    0.6000    0.6000    0.6000    0.6000    0.6000    0.6000
    0.2000    0.2000    0.2000    0.2000    0.2000    0.2000
   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000   -0.2000
   -0.6000   -0.6000   -0.6000   -0.6000   -0.6000   -0.6000
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
```

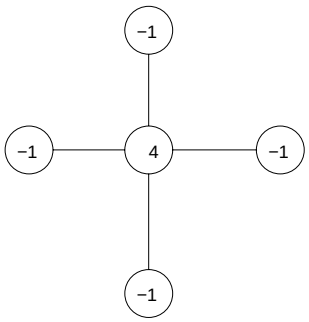
Från satsen `G= ....` genomförs en konsekutiv numrering av de inre diskretiseringspunkterna. De obekanta kommer senare att numreras på detta sätt. De punkter som inte är inre punkter markeras med nollor. Programsnutten ger förljande matris `G`.

```
G =

     0     0     0     0     0     0
     0     1     5     9    13     0
     0     2     6    10    14     0
     0     3     7    11    15     0
     0     4     8    12    16     0
     0     0     0     0     0     0
```

Om vi behöver få tillgång till x-koordinaten för den diskretiseringspunkt som har numret 14 så kan vi skriva `x(inre(14))`. Om vi vill ha y-koordinaten till den randpunkt som ligger ovanför diskretiseringspunkt nr 5 skriver vi `y(inre(5)-1)`. x-koordinaten till den randpunkt som ligger till höger om diskretiseringspunkt nr 14 skriver vi `x(inre(14)+m)` med `m=6` lika med antalet rader i matrisen `G`. En matris i Matlab kan refereras med **ett** index; matrisen uppfattas då som den vektor som erhålls om kolumnerna staplas på varandra med första kolumnen överst och sista kolumnen underst.

3.2 Diskretisering av ekvationen

De partiella derivatorna diskretiseras med hjälpa av beräkningsmolekylen



Molekylen är en representation av summan av centraldifferensapproximationer till andraderivatorna i x-led och y-led, och approximerar $-h^2(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})$

Koefficientmatrisen byggs upp genom att centrumpunkten i beräkningsmolekylen ovan får svepa över de inre punkterna i diskretiseringsnätet och skapa de matriselement som berörs av molekylen. De obekanta och ekvationerna numreras konsekutivt som i matrisen `G`.

En typisk ekvation ser ut enligt

$$-u_W - u_N + 4u_C - u_S - u_E = -h^2 f(x_C, y_C)$$

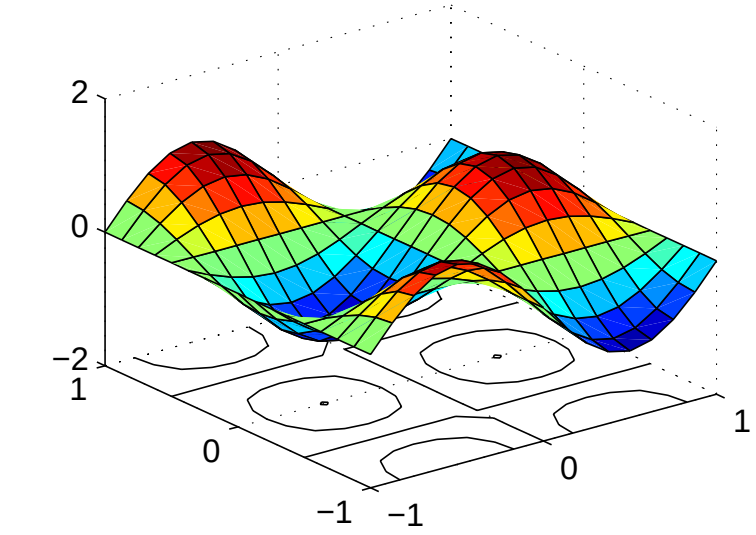
Här står u för lösningen, index för läget i molekylen; W = väst, N = norr, C = centrum, S = syd, E = öst. I de fall något eller några av värdena på vänster sida är kända från randvillkor flyttas motsvarande värde över i högerledet, dvs där beräkningsmolekylen berör en randpunkt modifieras högerledsvektorn med motsvarande randvärde, se programmet `randv2Dgles.m`.

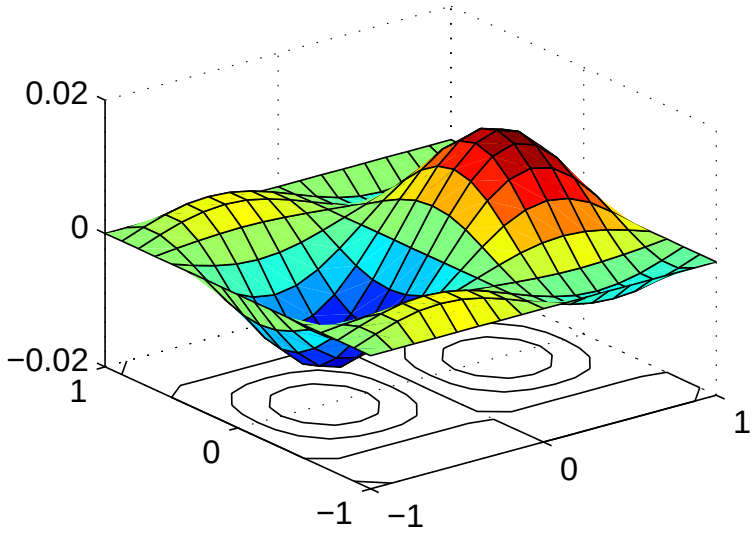
Varje inre punkt motsvarar en ekvation i ekvationssystemet, med början i den övre vänstra inre punkten. Ekvationer och obekanta numreras på samma sätt. För varje inre punkt skall du titta på dina grannar i riktningarna öst, väst, nord och syd (om du befinner dig i ett 3-dimensionellt område skall du också betrakta grannarna över och under dig). Om din granne i den aktuella riktningen också är en inre punkt skall du generera ett element i koefficientmatrisen (-1 i vårt fall). Om grannen är en randpunkt så är funktionsvärdet känt och du kan föra över den på högra sidan om likhetstecknet. Gör detta i varje riktning och ge också diagonalelementet i koefficientmatrisen sitt värde (4 i vårt fall), innan du går vidare till nästa punkt.

```
   clear, clf, hold off
   fchl=inline('-2*pi^2*sin(pi*x).*cos(pi*y)','x','y');
   fcrd=inline('sin(pi*x).*cos(pi*y)','x','y');

%-----------------------------------------------------------------------
   N=15;  hx=2/(N+1); hy=hx;
   x = ones(N+2,1)*[-1, -1+(1:N)*hx, 1];
   y = flipud(x'); %kasta om raderna i matrisen, sista blir första osv.

   G = (x > -1) & (x < 1) & (y > -1) & (y < 1); %mark inner points
                                                %the dimension of G
                                                %is the same as for
                                                %x and y

   inre=find(G);           %returns the indices  that are
                           %non-zero, columnvector
                           % G is regarded as a vector
   G = zeros(size(G));     % Convert from logical to double matrix

   G(inre)=(1:length(inre))';     % Numbering of the interior nodes

   Um=zeros(size(G));      %prepare the solutionmatrix


%generate matrix and righthandside-------------------------------------

[m,n] = size(G);  %find dimension of matrix G, m is the distance
                  %in index between elements in the same row

fG=-fchl(x(inre),y(inre))*hx^2;


% Indices of interior points
rad=[]; kol=[]; varde=[];
for nr = inre'  %här måste stå en radvektor för att vi skall ta
                %komponent för komponent


% Connect interior points to themselves with 4's. (diagonal elements)
i = G(nr); j = G(nr); s = 4;
rad=[rad; i]; kol=[kol;j]; varde=[varde;s];

% for k = north, east, south, west
 for k = [-1 m 1 -m]
   % Possible interior neighbors in k-th direction
   Q = G(nr+k);
   if Q~=0
     i=G(nr); j=Q; s=-1;
     rad=[rad; i]; kol=[kol;j]; varde=[varde;s];
   else  %boundary point--correct right hand side
     i=G(nr);
     if ( (k==-1) | (k==1) )
         xx=x(nr);yy=y(nr+k);
     else
         xx=x(nr+k); yy=y(nr);
      end
      sl=fcrd(xx,yy);
      fG(i)=fG(i)+sl; Um(nr+k)=sl;
   end

 end
end

D = sparse(rad,kol,varde);
```

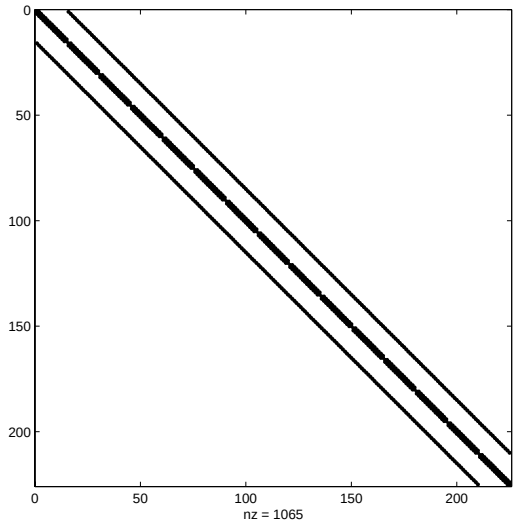
```
u=D\fG;
Usav=fcrd(x,y);%möjligt för testproblem
                %då exakta lösningen är känd --lika med randfunktionen
%subplot(2,2,2);
%surfc(x,y,Usav)

Um(inre)=u;
%Um(find(Um==0))=inf;
subplot(2,2,1)
surfc(x,y,Um)
err=Um-Usav; fel=norm(max(err),inf) %möjligt då exakt lsg känd
subplot(2,2,3)
surfc(x,y,err)
```

Då programmet exekveras med `N=15` erhålls nedanstående grafer för den approximativa lösningen respektive felet

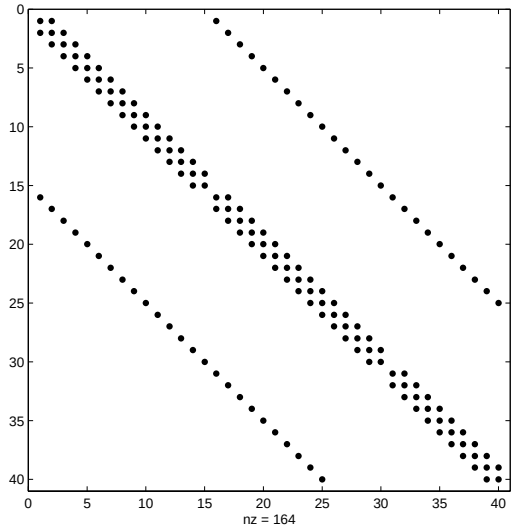




Med kommandot `spy(D)` får vi följande bild av strukturen av icke-nollor (bandmatris)



Med kommandot `spy(D(1:40,1:40))` får vi följande mer detaljerade bild av strukturen av icke-nollor i det övre vänstra hörnet av koefficientmatrisen.



Med kommandot `full(D(1:16,1:16))` skriver vi ut den avsnittsmatris som består av de första sexton raderna och kolumnerna av koefficientmatrisen `D`.

```
 4    -1     0     0     0     0     0     0     0     0     0     0     0     0     0    -1
-1     4    -1     0     0     0     0     0     0     0     0     0     0     0     0     0
 0    -1     4    -1     0     0     0     0     0     0     0     0     0     0     0     0
 0     0    -1     4    -1     0     0     0     0     0     0     0     0     0     0     0
 0     0     0    -1     4    -1     0     0     0     0     0     0     0     0     0     0
 0     0     0     0    -1     4    -1     0     0     0     0     0     0     0     0     0
 0     0     0     0     0    -1     4    -1     0     0     0     0     0     0     0     0
 0     0     0     0     0     0    -1     4    -1     0     0     0     0     0     0     0
 0     0     0     0     0     0     0    -1     4    -1     0     0     0     0     0     0
 0     0     0     0     0     0     0     0    -1     4    -1     0     0     0     0     0
 0     0     0     0     0     0     0     0     0    -1     4    -1     0     0     0     0
 0     0     0     0     0     0     0     0     0     0    -1     4    -1     0     0     0
 0     0     0     0     0     0     0     0     0     0     0    -1     4    -1     0     0
 0     0     0     0     0     0     0     0     0     0     0     0    -1     4    -1     0
 0     0     0     0     0     0     0     0     0     0     0     0     0    -1     4     0
-1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     4
```

3.3 Oregelbundet område

Poissons ekvation på ett område som utgör en delmängd av den kvadrat vi jobbade med i sektion 3.1 diskretiseras med i stort sett samma program som tidigare.

För ett L-format område enligt

$$-1 \leq x \leq 1,\ -1 \leq y \leq 1$$

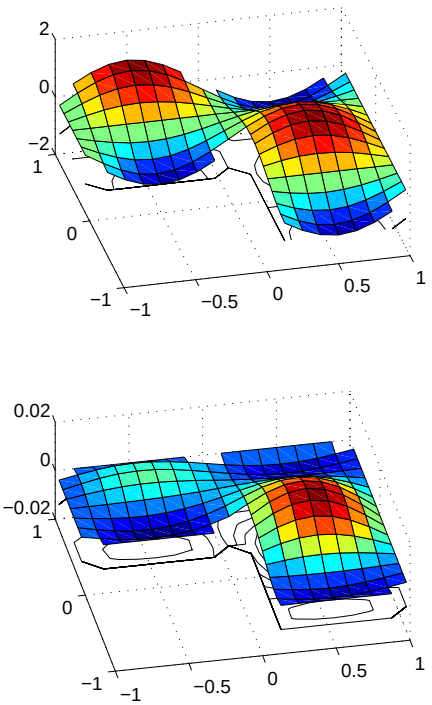
med delen i fjärde kvadranten bortskuren, ersätts satsen som ger de inre punkterna av följande

```
%L-format område
   G = (x > -1) & (x < 1) & (y > -1) & (y < 1) & ( (x > 0) | (y > 0));
```

Resten av programmet `randv2Dgles.m` fungerar utan förändringar. För utskrifterna ändras de sista satserna lite enligt

```
Um(inre)=u;
Um(find(Um==0))=inf;
subplot(2,2,1)
surfc(x,y,Um)
err=Um-Usav; fel=norm(max(err(inre)),inf)
subplot(2,2,3)
surfc(x,y,err)
```

Plotbilden från programmet blir, efter 3D-rotation av de två bilderna direkt i grafikfönstret.



I grafikfönstret finns bredvid förstoringsglasen en symbol i form av en cirkulär streckad pil. Om man klickar på den kan bilden i grafikfönstret roteras.

För besvärligare randfunktioner och högerledsfunktioner kan inte inline-funktioner användas, utan de måste skrivas som fristående Matlab-funktioner i egna filer.