Projekt: Netzglättung in MATLAB

Vorlesung “Lineare Algebra und Numerische Mathematik”

19. Februar 2014

Definition 1 (Dreiecksnetz)**.** *Unter einem (planaren) Dreiecksnetz $\mathcal{M}$ versteht man eine Menge $\mathcal{N}$ von $N \in \mathbb{N}$ paarweise verschiedenen Punkten in der Ebene, zusammen mit einer Menge $\mathcal{T}$ von $M \in \mathbb{N}$ Dreiecken mit Eckpunkten aus $\mathcal{N}$, so dass folgende zwei Bedingungen erfüllt sind:*

1. *die Dreiecke überlappen sich nicht,*
2. *für je zwei verschiedene Dreiecke aus $\mathcal{T}$ trifft genau eine der folgenden Situationen zu:*
   (a) *die beiden Dreiecke haben keinen Punkt gemeinsam,*
   (b) *die beiden Dreiecke haben genau einen Eckpunkt aus $\mathcal{N}$ gemeinsam,*
   (c) *die beiden Dreiecke haben genau die Verbindungsstrecke (einschliesslich der Endpunkte) von zwei Punkten aus $\mathcal{N}$ gemeinsam.*

Die Punkte in $\mathcal{N}$ heissen auch Knoten des Netzes, die Dreiecke in $\mathcal{T}$ Maschen des Netzes, und jede Verbindungsstrecke zweier Knoten, die als Dreiecksseite vorkommt, wird als Kante bezeichnet.

Abbildung 1 illustriert diese Definition. Dreiecksnetze sind von fundamentaler Bedeutung für Computergraphik, Architektursoftware, Landschaftsmodelle, und Computersimulationen in Ingenieuranwendungen, siehe Abbildung 2.

Die Knoten und Dreiecke eines Netzes denken wir uns immer von 1 bis N bzw. M durchnummeriert.

In MATLAB wird ein planares Netz durch folgende Daten beschrieben:

- Spaltenvektor $\mathbf{x} \in \mathbb{R}^N$, der die x-Koordinaten der Knoten enthält,
- Spaltenvektor $\mathbf{y} \in \mathbb{R}^N$ von y-Koordination der Knoten
- und eine $M \times 3$-Matrix $\mathbf{T}$, deren Zeilen jeweils die drei Nummern der Eckpunkte eines Dreiecks enthalten.

MATLAB stellt die Funktion `triplot` zur Verfügung, die aus diesen Daten das Netz zeichnet, siehe Listing 1. Dieses MATlAB-Skript steht auch zum Download zur Verfügung.

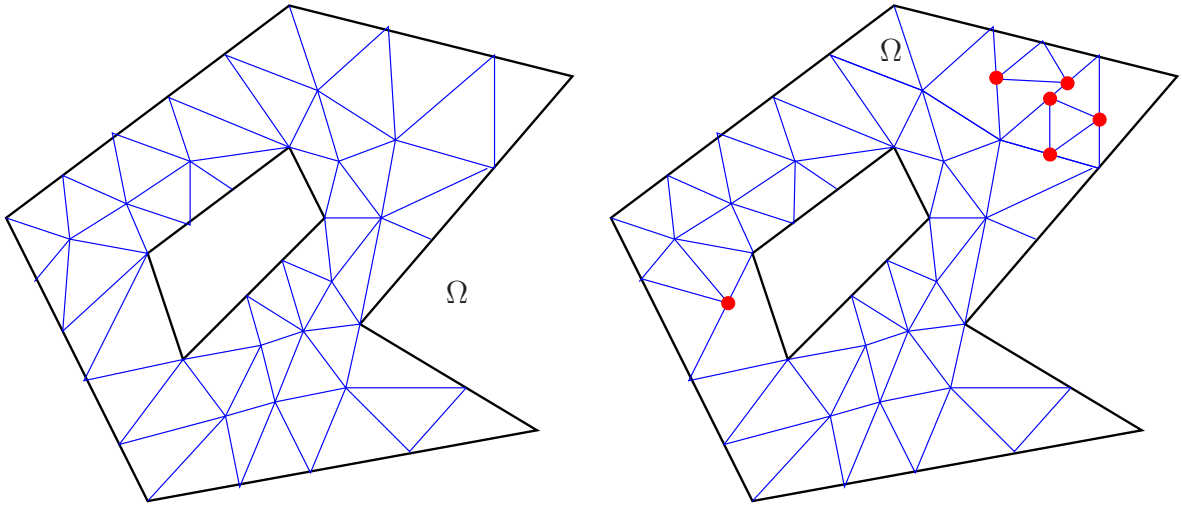

Abbildung 1: Beispiel für ein zulässiges Dreiecksnetz (links) und eine Anordnung von Knoten und Dreiecken, die die Bedingungen aus der Definition verletzt (rechts).

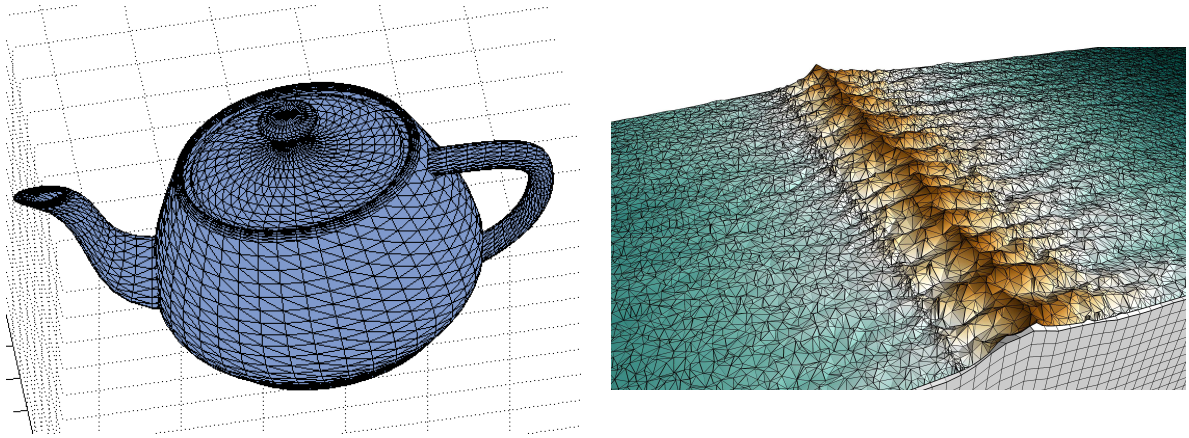

Abbildung 2: Modellierung mit Hilfe von (dreidimensionalen) Dreiecksnetzen

Listing 1: (meshplotdemo.m) Zeichnen eines planaren Netzes

```
% MATLAB demonstration for visualizing a planes triangular mesh
% Initialize node coordinates
% First the x-coordinates
x = [1.0;0.60;0.12;0.81;0.63;0.09;0.27;0.54;0.95;0.96];
% Next the y-coordinates
y = [0.15;0.97;0.95;0.48;0.80;0.14;0.42;0.91;0.79;0.95];
% Then specify triangles through the indices of their vertices.
   These
% indices refer to the ordering of the coordinates as given in the
% vectors x and y.
T = [8 2 3;6 7 3;5 2 8;7 8 3;7 5 8;7 6 1;...
     4 7 1;9 5 4;4 5 7;9 2 5;10 2 9];
% Call the plotting routine; draw mesh with blue edges
triplot(T,x,y,'b-'); title('A simple planar triangular mesh');
xlabel('{\bf x}'); ylabel('{\bf y}');
axis([-0.05 1.05 -0.05 1.05]); axis equal;
```

```
% Mark nodes with red stars
hold on; plot(x,y,'r*');

% Save plot a vector graphics
print -depsc2 'meshplot.eps';
```

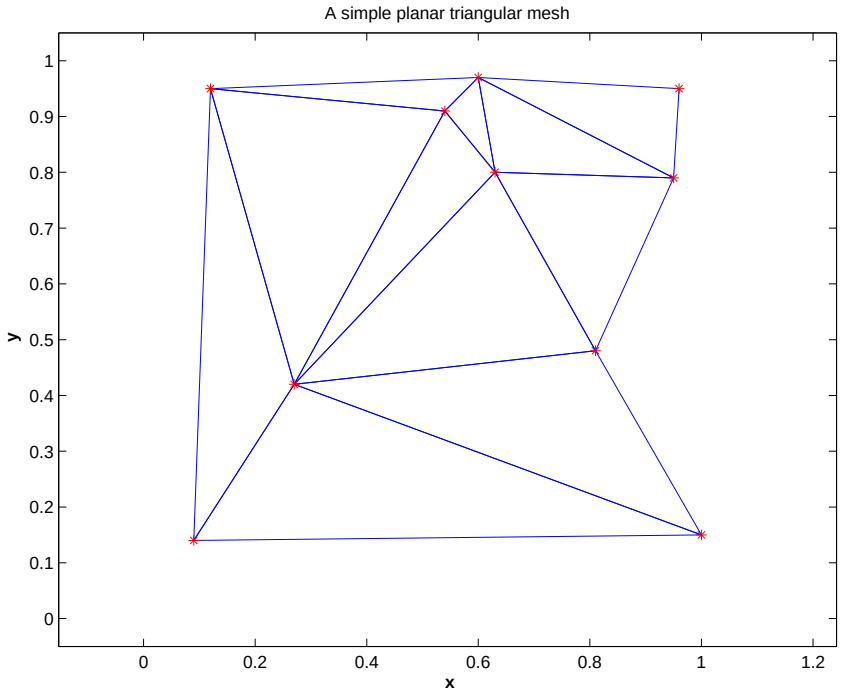

Abbildung 3: Ausgabe von `meshplotdemo.m`

Definition 2. *Eine Randkante eines planaren Netzes ist eine Kante, die nur Seite eines einzigen Dreiecks ist. Die Endpunkte von Randkanten heissen Randknoten*

1. Zur Verfügung steht auch die MATLAB-Funktion aus Listing 2 (auch verfügbar online), die leider nicht dokumentiert wurde[1] Beschreiben Sie genau, welchen Zweck diese Funktion erfüllt und wie dieser Zweck erreicht wird. Dazu müssen Sie die Dokumentation zu den MATLAB-Kommandos `find` und `sort` studieren.

Hinweis. Ein MATLAB-Skript `meshtest.m`, siehe Listing 9 ist online verfügbar und demonstriert die Verwendung von `processmesh`.

Listing 2: Undokumentierte Funktion zu Teilaufgabe 1

```
function [E,Eb] = processmesh(T)
N = max(max(T)); M = size(T,1);
T = sort(T')';
C = [T(:,1) T(:,2); T(:,2) T(:,3); T(:,1) T(:,3)];
% Wow! A creative way to use 'sparse'
```

[1] Wir hoffen, dass alle Studierenden im Kurs alle ihre Codes gut und ausführlich dokumentieren ("There are two kinds of programmers: those who document their codes, and those who will").

```
A = sparse(C(:,1),C(:,2),ones(3*M,1),N,N);
[I,J] = find(A > 0); E = [I,J];
[I,J] = find(A == 1); Eb = [I,J];
```

Listing 3: Dokumentierte Funktion zu Teilaufgabe 1

```
function [E,Eb] = processmesh(T)
% MATLAB function extracting edge information of a mesh
% Argument 'T' is a N x 3 matrix containing the vertex numebers of
% each of the N triangles of a triangular mesh.
%
% Return value E is a matrix whose rows correspond to the edges of
    the
% mesh and give the indices of the endpoints of the edges. Return
% value Eb is the same data structure for edges on the boundary.
%
% Number of nodes of the triangular mesh
N = max(max(T));
% Number of triangles of the mesh
M = size(T,1);
% Sorts the vertices of each triangle in ascending order (Matrix
    needs
% to be transposed, because 'sort' acts on columns)
T = sort(T')';
% Create a matrix C with two dolumns; the first columns contains the
% numbers of first points of edges, the second the number of second
% points
C = [T(:,1) T(:,2); T(:,2) T(:,3); T(:,1) T(:,3)];
% The matrix C is used to initialize a sparse matrix A through
% MATLAB's 'sparse' command. The rows and the columns of the matrix
% correspond to nodes of the mesh. If A(i,j) is different from zero
% the two nodes with numbers i and j are connected by an edge. Note
% that A(i,j) is equal to 2, if [i,j] is an interior edge. For a
% boundary edge [i,j] A(i,j) == 1, because an interior edge is
% contained twice in the matrix C.
A = sparse(C(:,1),C(:,2),ones(3*M,1),N,N);
% MATLAB's find command gives all pairs of row and column indices of
% matrix elements satisfying a certain condition. This is used to
% build the matrices E and Eb
[I,J] = find(A > 0); E = [I,J];
[I,J] = find(A == 1); Eb = [I,J];
```

2. Es gibt eine weitere undokumentierte Funktion `getinfo`, siehe Listing 5. Bekannt ist nur, dass das Argument `T` die Dreiecksliste eines Dreiecksnetzes enthalten muss, während `E` einer der Rückgabewerte von `processmesh` ist.

Listing 4: Undokumentierte Funktion zu Teilaufgabe 2

```
function ET = getinfo(T,E)
% Argument T: N x 3 matrix, nodes of triangles
% (N = number of triangles of the mesh)
```

```
% Argument E: L x 2 matrix, endpoints of edges
% (L = number of edges of the mesh)
%
% return value: ET is a N x 3 -matrix (N = no. of triangles)
% whose rows contain the index numbers of the edges of the
% the triangles.
L = size(E,1);   % Number of edges
N = max(max(T)); % Number of nodes
% Create a sparse N x N - matrix A, for which A(i,j) is zero, if the
% nodes i and j are not connected by an edge, and for which A(i,j)
   gives
% the INDEX of the edge from i to j, if it exists. The index of an
   edge
% corresponds to its row number in the E matrix
A = sparse(E(:,1),E(:,2),(1:L)',N,N);
% Loop over all triangles
ET = [];
for tri=T'
  % The node numbers of the current triangles are contained
  % in the loop variable tri (row vector of length 3)
  % Eloc is a 3x3-matrix that contains the numbers of
  % the edges of the triangles: Eloc(i,j) gives the number
  % of the edge connecting vertex i and vertex j, 1<=i,j<=3,
  % of the triangle
  Eloc = full(A(tri,tri)); Eloc = Eloc + Eloc';
  % The three edge numbers are stored in a row of the matrix
  % ET: order: edge [1,2] - edge [1,3] - edge [2,3]
  ET = [ET; Eloc([8 7 4])];
end
```

Listing 5: Undokumentierte Funktion zu Teilaufgabe 2

```
function ET = getinfo(T,E)

N = max(max(T));
L = size(E,1); A = sparse(E(:,1),E(:,2),(1:L)',L,L);

ET = [];
for tri=T'
  Eloc = full(A(tri,tri)); Eloc = Eloc + Eloc';
  ET = [ET; Eloc([8 7 4])];
end
```

Listing 6: Dokumentierte Funktion zu Teilaufgabe 2

```
function ET = getinfo(T,E)
% Argument T: N x 3 matrix, nodes of triangles
% (N = number of triangles of the mesh)
```

```
% Argument E: L x 2 matrix, endpoints of edges
% (L = number of edges of the mesh)
%
% return value: ET is a N x 3 -matrix (N = no. of triangles)
% whose rows contain the index numbers of the edges of the
% the triangles.
L = size(E,1);   % Number of edges
N = max(max(T)); % Number of nodes
% Create a sparse N x N - matrix A, for which A(i,j) is zero, if the
% nodes i and j are not connected by an edge, and for which A(i,j) gives
% the INDEX of the edge from i to j, if it exists. The index of an edge
% corresponds to its row number in the E matrix
A = sparse(E(:,1),E(:,2),(1:L)',N,N);
% Loop over all triangles
ET = [];
for tri=T'
  % The node numbers of the current triangles are contained
  % in the loop variable tri (row vector of length 3)
  % Eloc is a 3x3-matrix that contains the numbers of
  % the edges of the triangles: Eloc(i,j) gives the number
  % of the edge connecting vertex i and vertex j, 1<=i,j<=3,
  % of the triangle
  Eloc = full(A(tri,tri)); Eloc = Eloc + Eloc';
  % The three edge numbers are stored in a row of the matrix
  % ET: order: edge [1,2] - edge [1,3] - edge [2,3]
  ET = [ET; Eloc([8 7 4])];
end
```

Ein Netz wird *verfeinert*, indem man seine Knoten zusammen mit den Mittelpunkten aller Kanten als Knoten eines neuen Netzes nimmt. Die Dreiecke des neuen Netzes sind alle die Dreiecke, die entstehen, wenn man alle Dreiecke des alten (“groben”) Netzes in vier kongruente Dreiecke mit jeweils der halben Seitenlänge zerlegt.

3. Schreibe eine MATLAB-Funktion

```
function [x_ref,y_ref,T_ref] = refinemesh(x,y,T)
```

die die Daten des groben Netzes als Argumente nimmt und daraus die entsprechenden Daten für das neue, verfeinerte Netz generiert.

Hinweis: `processmesh` leistet hier gute Dienste.

Listing 7: MATLAB-Funktion aus Teilaufgabe 3

```
function [x_ref,y_ref,T_ref] = refinemesh(x,y,T)
% MATLAB function for regular refinement of a triangular mesh
% Arguments:
```

```
% 'x': column vector of x-coordinates of nodes of mesh
% 'y': column vector of y-coordinates of nodes of mesh
% 'T': N x 3 - matrix of numbers of nodes of triangles
% (see documentation of MATLAB's triplot function)
% Return values:
% 'x_ref', 'y_ref' proivide the coordinates of the nodes
% of the refined mesh.
% 'T_ref' gives the triangles of the refined mesh.

% First obtain information on the (boundary) edges
% E and Eb are matrices whose rows contain the numbers
% of the endpoints of edges
[E,Eb] = processmesh(T);
% Append midpoints of edges to list of vertices
x_ref = [x;0.5*(x(E(:,1))+x(E(:,2)))];
y_ref = [y;0.5*(y(E(:,1))+y(E(:,2)))];

% Fetch information about the edges of triangles
% ET is a matrix whose rows contain the numbers of
% the edges of the triangles of the mesh
ET = getinfo(T,E);
% Build new list of triangles
Nt = size(T,1), % Number of triangles
Nv = length(x); % Number of vertices
T_ref = [];
for l = 1:Nt
  % First son triangle
  T_ref = [T_ref; T(l,1), ET(l,2)+Nv, ET(l,3)+Nv];
  % Second son triangle
  T_ref = [T_ref; T(l,2), ET(l,1)+Nv, ET(l,3)+Nv];
  % Third son triangle
  T_ref = [T_ref; T(l,3), ET(l,1)+Nv, ET(l,2)+Nv];
  % Fourth (interior) son triangle
  T_ref = [T_ref; ET(l,1)+Nv, ET(l,2)+Nv, ET(l,3)+Nv];
end
```

Wir führen die Notation

$$S(i) := \{j \in \{1, \ldots, N\} : \text{ Knoten } i \text{ und } j \text{ sind durch eine Kante verbunden}\}$$

für die Menge der Indices der "Nachbarn" eines Knotens ein. Weiter bezeichne im Folgenden $\mathbf{x}^i \in \mathbb{R}^2$ die Position des Knotens mit dem Index i. Schliesslich stehe $\Gamma \subset \{1, \ldots, N\}$ für die Menge der Nummern von Randknoten.

Definition 3. *Ein Dreiecksnetz heisst geglättet, wenn* [2]

$$\mathbf{x}^i = \frac{1}{\sharp S(i)} \sum_{j \in S(i)} \mathbf{x}^j \quad \textit{für alle} \quad i \in \{1, \ldots, N\} \setminus \Gamma \, ,$$

d.h. jeder innere Knoten liegt im Schwerpunkt aller seiner Nachbarn.

4. Wir schreiben die Koordinaten aller *inneren* Knoten sukzessive in den Spaltenvektor $\mathbf{z} \in \mathbb{R}^n$, $n := 2(N - \sharp\Gamma)$, gemäss der Vorschrift

$$z_i = \begin{cases} x_1^i & \text{, wenn } 1 \le i \le \frac{n}{2} \, , \\ x_2^{i-\frac{n}{2}} & \text{, wenn } \frac{n}{2} + 1 \le i \le n \, . \end{cases}$$

Für ein *geglättetes* Dreiecksnetz löst $\mathbf{z}$ ein lineares Gleichungssystem $\mathbf{A}\mathbf{z} = \mathbf{b}$. Beschreibe die Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ und den Rechte-Seite-Vektor $\mathbf{b} \in \mathbb{R}^n$.

5. Warum hat das in der vorhergehenden Teilaufgabe hergeleitete lineare Gleichungssystem immer eine eindeutige Lösung.

Hinweis. Folge den Überlegungen aus Abschnitt 3.8.1 der Vorlesung, die zu dem Schluss führten, dass die linearen Gleichungssysteme aus Netzwerkmodellen immer eine eindeutige Lösung haben.

6. Unter Beibehaltung der Verbindungsrelationen der Knoten (Konnektivität) und der Positionen der Randknoten lässt sich jedes Dreiecksnetz durch Verschieben innerer Knoten in ein geglättetes Netz transformieren.

Schreiben Sie eine MATLAB-Funktion

```
function [xs,ys] = smoothmesh(x,y,T),
```

die diese Transformation für ein Dreiecksgitter, das wie oben beschrieben durch `x`, `y` und `T` spezifiziert ist, durchführt. In den Spaltenvektoren `xs` und `ys` sollen die neuen Knotenkoordinaten zurückgegeben werden.

Listing 8: MATLAB-Funktion zur Netzglättung

```
function [xs,ys] = smoothmesh(x,y,T)
% MATLAB function for smoothing a planar triangular mesh by moving all
% vertices to the barycenter of their neighboring nodes. The arguments
% x and y contain the node coordinates, whereas T encodes the triangle
% node incidence relationship.
%
% Arguments are the same as for 'refinemesh'. The function returns
% the new node positions of the smoothed mesh. Note that the
% connectivity is not affected by the smoothing.

```

[2] Das Symbol $\sharp$ steht für die Kardinalität einer Menge, das ist die Anzahl ihrer Elemente.

```
% Number of nodes of the mesh
Nv = length(x);

% First obtain the edges of the mesh. E will be a Ne x 2-matrix whose
% rows give the indices of the nodes adjacent to an edge. Eb contains
% only the boundary edges in the same format.

[E,Eb] = processmesh(T);
% Number of edges of the mesh
Ne = size(E,1);
% Obtain indices of interior vertices. This is done by first
% extracting all the vertex numbers that occur in the list of boundary
% edges. These vertices will be flagged by setting entries in a vector
% of index numbers to zero.
idx = (1:Nv)'; idx([Eb(:,1);Eb(:,2)]) = 0;
% Indices of vertices on the boundary
int_idx = find(idx > 0);
% Indices of vertices in the interior
bd_idx = find(idx == 0);
% Assemble the (saymmetric) graph Laplacian matrix: this is that
% sparse symmetric quadratic matrix, whose size agrees with the total
% number of vertices, and whose off-diagonal entries are -1, whenever
% two vertices are connected by an edge. The diagonal entry is set
% such that the sum of all entries in each row is equal to zero.
L = sparse(E(:,1),E(:,2),ones(Ne,1),Nv,Nv); L = L+L';
L = spdiags(sum(L)',[0],Nv,Nv) - L;
% Remove rows and columns corresponding to vertices on the boundary
% using MATLAB's sub-matrix extraction capability.
A = L(int_idx,int_idx);
% Build right hand side vector. This vector depends on the positions
% of the nodes on the boundary. It can be obtained by multiplying the
% vector of either x or y coordinates of nodes on the boundary with
% that columns of the graph Laplacian matrix that corresponds to nodes
% on the boundary.
B = [x(bd_idx),y(bd_idx)];
B = -L(int_idx,bd_idx)*B;
% Solve linear system for both x- and y-coordinates simultaneously
X = A\B;
% Build output vectors by merging new positions of interior nodes and
% fixed positions of nodes on the boundary.
xs = x; ys = y;
xs(int_idx) = X(:,1); ys(int_idx) = X(:,2);
```

Hinweis. Die neuen Positionen der inneren Knoten des geglätteten Netzes lassen sich natürlich durch Lösung des linearen Gleichungssystems aus Teilaufgabe **4** finden.

Listing 9: Testrahmen für MATLAB-Funktionen zu Dreiecksnetzen

```
% MATLAB Skript for testing mesh handling functions that were
% developed as part of the project Netzglaetungfor the course
% Lineare Algebra und Numerische Mathematikfor D-BAUG at ETH
% Zurich.

figure('name','Triangular mesh');
% Initialize basic mesh data and plot mesh. Note that the variables of
% a MATLAB script are persistent!
meshplotdemo;

% Obtain relevant information calling a function whose purpose has to
% be determined, see sub-problem 1.
[E,Eb] = processmesh(T);

% Add index numbers to nodes and triangles of the mesh. This is useful
% for understanding the coding of mesh information and also the
% meaning of the data sotred in the matrices E and Eb.
for l=1:length(x)
  text(x(l),y(l),num2str(l),'fontsize',14,'color','m');
end
for l=1:size(T,1)
  text(sum(x(T(l,:)))/3,sum(y(T(l,:)))/3,num2str(l),'color','b');
end

% Guess what is plotted and annotated here. This offers a key hint at
% the output of 'processmesh'
k = 1;
for e = E'
  % Note the use of sub-vector extraction in MATLAB.
  plot(x(e),y(e),'k-');
  text(0.5*sum(x(e)),0.5*sum(y(e)),num2str(k),'color','k');
  k = k+1;
end
% What does this line of code do?
for e = Eb', plot(x(e),y(e),'r-'); end

% Obtain some more interesting information on the mesh
disp('Output of getinfo');
ET = getinfo(T,E),

% Loop: Successive smoothing and refinement of the mesh plus graphical
```

```
% rendering, which is displayed as Figure 4 of the project sheet.
for l=1:3
  % refine the mesh and draw it
  [x,y,T] = refinemesh(x,y,T);
  figure('name',['Refined mesh level ' num2str(l)]);
  triplot(T,x,y,'b-'); title(['Refined mesh level '
     num2str(l)]);
  axis([0 1 0 1]); axis equal; axis off;
  hold on; plot(x,y,'r+');
  print('-depsc2',['rmesh' num2str(l) '.eps']);

  % Smooth the mesh and draw it again
  [x,y] = smoothmesh(x,y,T);
  figure('name',['smoothed mesh level ' num2str(l)]);
  triplot(T,x,y,'b-'); title(['Smoothed mesh level '
     num2str(l)]);
  axis([0 1 0 1]); axis equal; axis off;
  hold on; plot(x,y,'r+');
  print('-depsc2',['smesh' num2str(l) '.eps']);
end
```

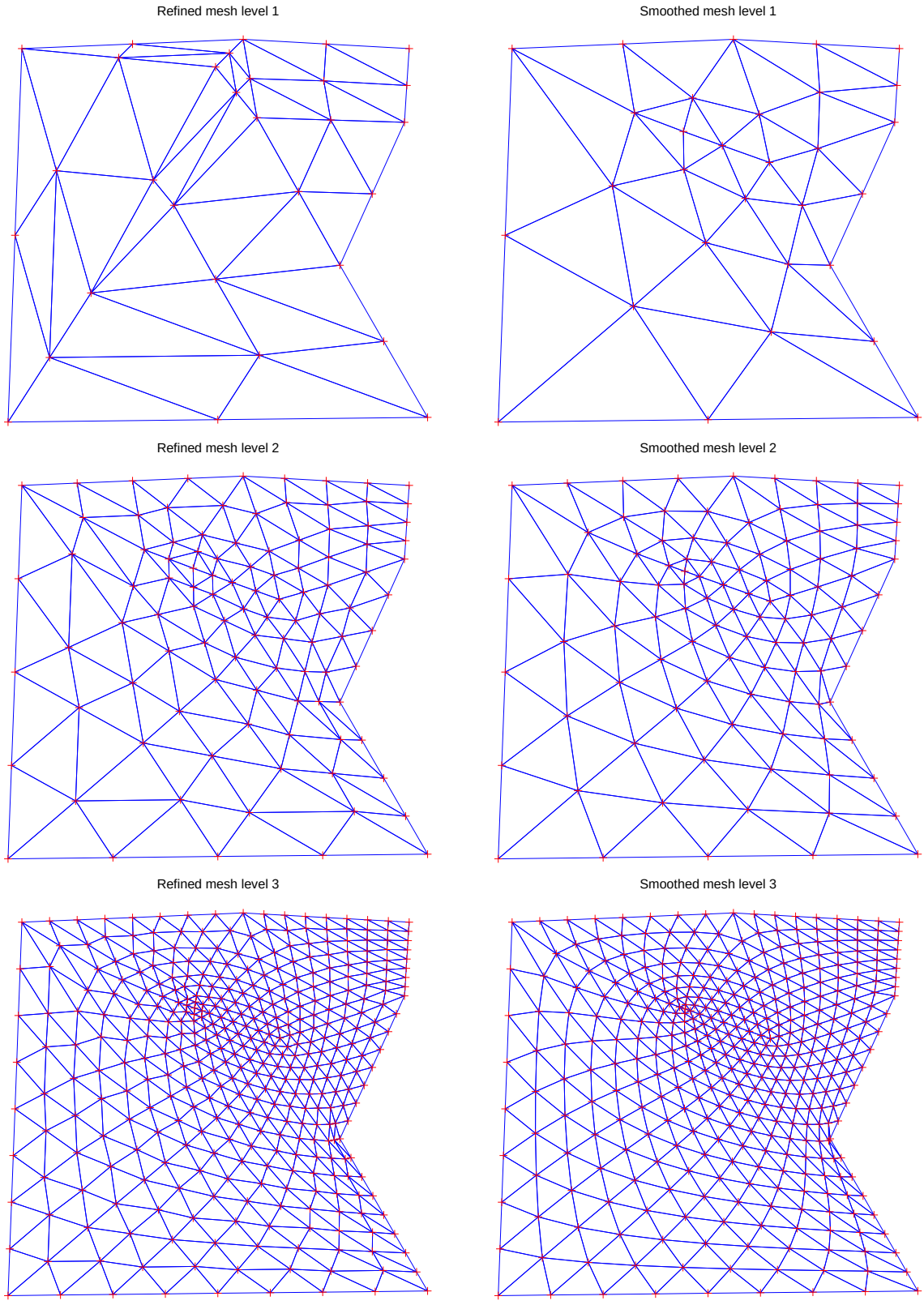

Abbildung 4: Verfeinerte und geglättete Dreiecksnetze, produziert von `meshtest.m`