# Projekt: Netzglättung in MATLAB

Prof. R. Hiptmair, SAM, D-MATH, ETH Zürich

Vorlesung "'Lineare Algebra und Numerische Mathematik" (D-BAUG)

**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.

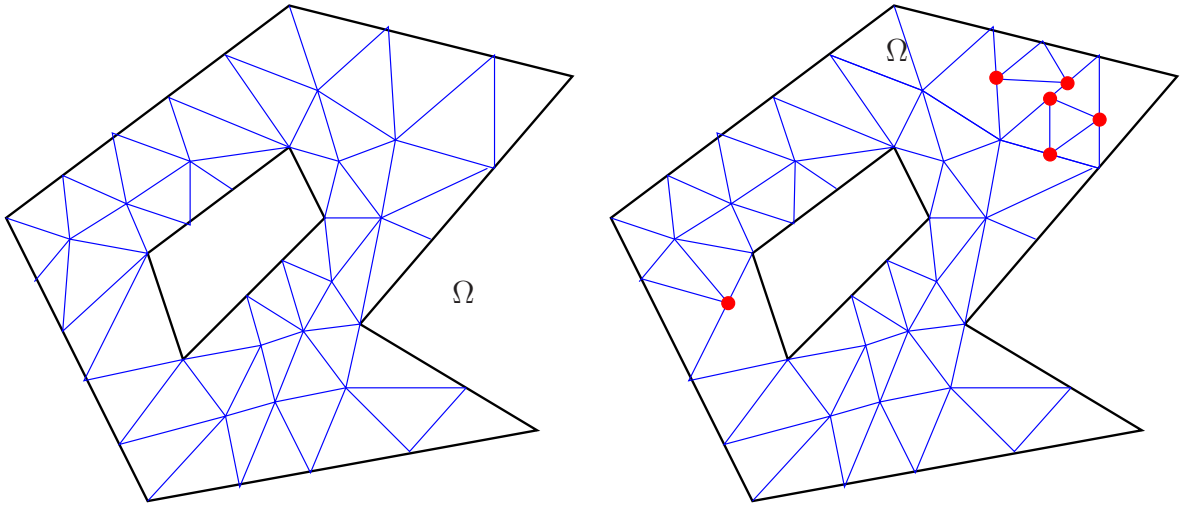


Abbildung 1: Beispiel für ein zulässiges Dreiecksnetz (links) und eine Anordnung von Knoten und Dreiecken, die die Bedingungen aus der Definition verletzt (rechts).

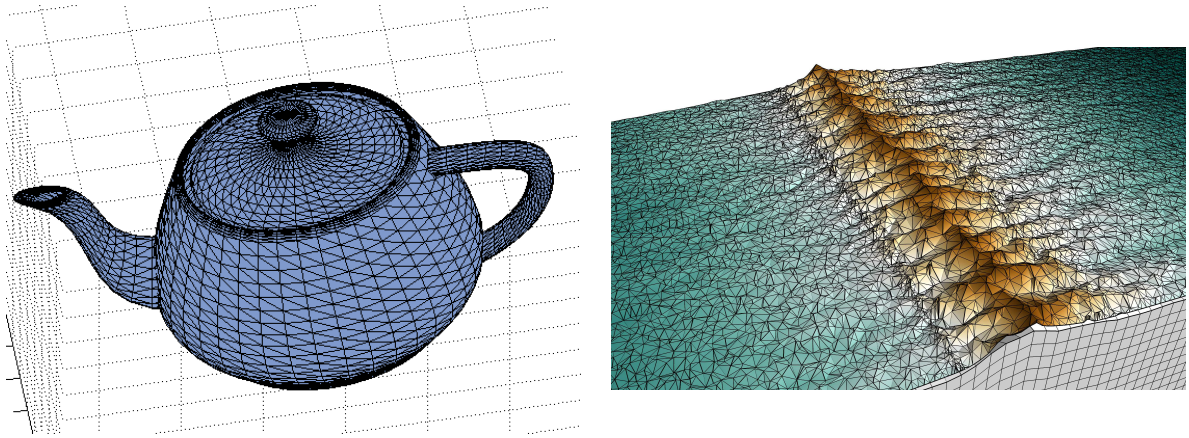

Abbildung 2: Modellierung mit Hilfe von (dreidimensionalen) Dreiecksnetzen

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';
```

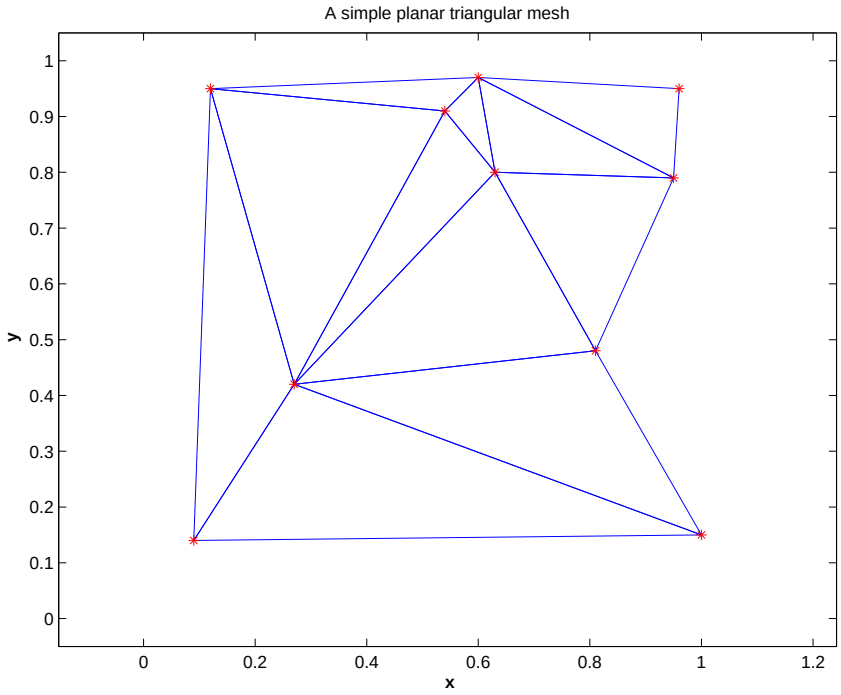


Abbildung 3: Ausgabe von `meshplotdemo.m`

**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 4 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];
```

**2.** Es gibt eine weitere undokumentierte Funktion `getinfo`, siehe Listing 3. Bekannt ist nur, dass das Argument `T` die Dreiecksliste eines Dreiecksnetzes enthalten muss, während `E` einer der Rückgabewerte von `processmesh` ist.

Listing 3: 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
```

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.

Wir führen die Notation

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

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 $\Gamma \subset \{1,\dots,N\}$ für die Menge der Nummern von Randknoten.

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

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

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

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

**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

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

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.** Überlegen Sie sich, warum die Koeffizientenmatrix des in der vorhergehenden Teilaufgabe hergeleitete linearen Gleichungssystems *diagonaldominant* ist.

**Definition.** Eine Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ heisst diagonaldominant, wenn

$$|(\mathbf{A})_{i,i}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |(\mathbf{A})_{i,j}| \quad \forall i \in \{1, \ldots, n\} \text{ .} \tag{1}$$

Dieser Begriff ist etwas schwächer als der in Abschntt 3.3 der Vorlesung engeführte Begriff der *strikten Diagonaldominanz*, der sich als hinreichend für die Invertierbarkeit einer quadratischen Matrix herausstellte, siehe Anwendung zu Satz III.3.0.P. Mit ähnlichen Argumenten lässt sich auch zeigen, dass ein Vektor nur dann im Kern einer diagonaldominantem quadratischen Matrix liegen kann, wenn alle seine Komponenten gleich sind. Für die vorliegende Koeffizientenmatrix schliessen wir daraus, dass sie invertierbar ist, denn sie hat eine Zeile, für die (1) mit $>$ gilt.

**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.

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

Listing 4: 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']);

```

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

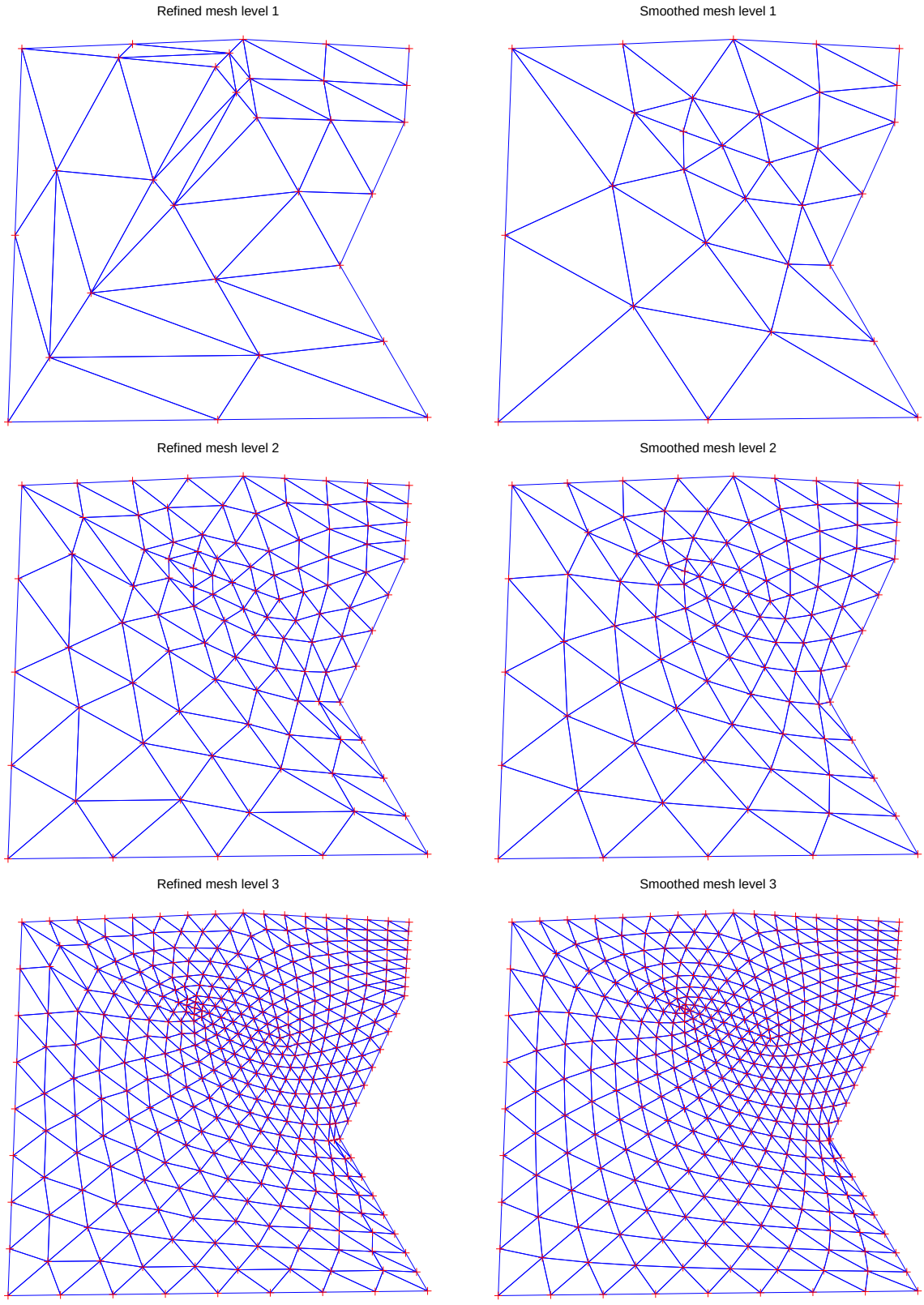


Abbildung 4: Verfeinerte und geglättete Dreiecksnetze, produziert von `meshtest.m`