Projekt: Winkelschätzung für Triangulierungen

Vorlesung “Lineare Algebra und Numerische Mathematik”

19. Februar 2014

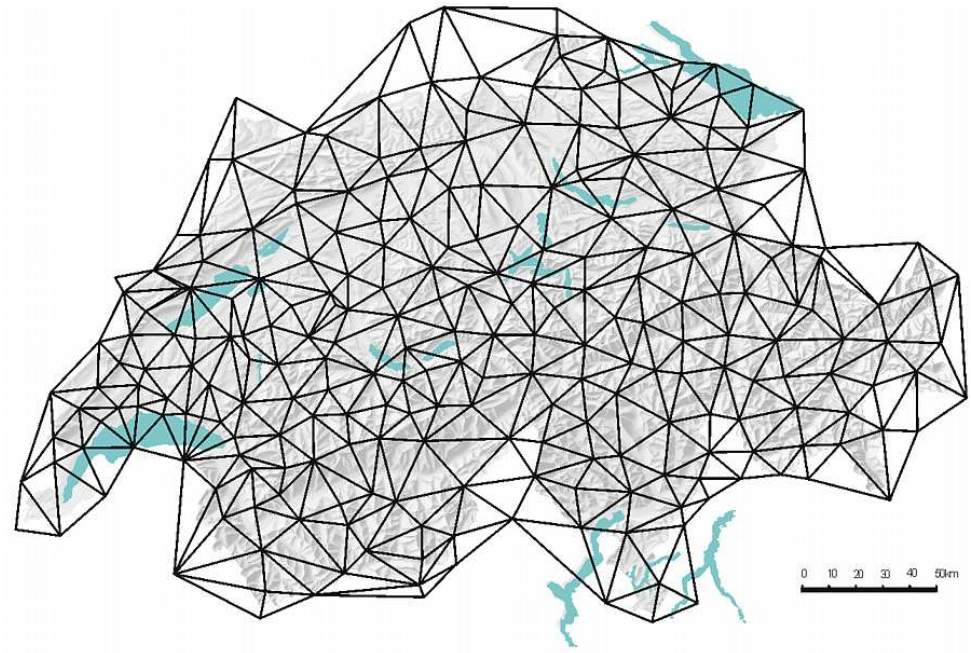

Abbildung 1: Ein Dreiecksnetz zur Vermessung der Schweiz

Dieses Projekt baut auf dem Projekt “Netzglättung in MATLAB” auf, das bereits publiziert worden ist. Aus diesem Projekt werden benötigt:

- Das Konzept eines planaren Dreiecksnetzes, siehe [1, Def. 1],
- Die MATLAB-Datenstruktur, die ein planares Dreiecksnetz beschreibt,
- Funktionen, die die Kanten eines Netzes berechnen.

(Planare) Dreiecksnetze waren und sind für die Vermessung von grosser Bedeutung, insbesondere vor der Entwicklung der lasergestützten Entfernungsmessung, denn bis dahin liessen sich Entfernungen nur sehr grob bestimmen, Winkel jedoch viel genauer. Siehe dazu den entsprechenden Wikipedia-Artikel. Insbesondere C.F. Gauss war einer der Pioniere des Einsatzes von Dreiecksnetzen zur Vermessung. Gauss war übrigens auch der Erfinder der Methode der kleinsten Quadrate, siehe dazu den Wikipedia-Artikel, die in Abschnitt 3.9 der Vorlesung eingeführt worden ist und in diesem Projekt eine grosse Rolle spielen wird:

Die Grundlagen seines Verfahrens hatte Gauss schon 1795 im Alter von 18 Jahren entwickelt. Basis war eine Idee von Pierre-Simon Laplace, die Beträge von Fehlern aufzusummieren, so dass sich die Fehler zu Null addieren. Gauss nahm stattdessen die Fehlerquadrate und konnte die künstliche Zusatzanforderung an die Fehler weglassen.

Gauss benutzte dann das Verfahren intensiv bei seiner Vermessung des Königreichs Hannover durch Triangulation. 1821 und 1823 erschien die zweiteilige Arbeit sowie 1826 eine Ergänzung zur *Theoria combinationis observationum erroribus minimis obnoxiae* (Theorie der den kleinsten Fehlern unterworfenen Kombination der Beobachtungen), in denen Gauss eine Begründung liefern konnte, weshalb sein Verfahren im Vergleich zu den anderen so erfolgreich war: Die Methode der kleinsten Quadrate ist in einer breiten Hinsicht optimal, also besser als andere Methoden.

Gegeben ist die *Inzidenzinformation* zu einem planaren Dreiecksnetz, also die Information, welche Knoten zu welchen Dreiecken (jeweils nummeriert von 1 bis N bzw. M) gehören. Diese Inzidenzinformation ist enthalten in einer $M \times 3$-Matrix $\mathbf{T}$, deren Zeilen jeweils die drei Nummern der Eckpunkte eines Dreiecks enthalten, siehe [1].

Gemessen werden nun alle Winkel der Dreiecke des Netzes; sie lassen sich ebenfalls in einer $M \times 3$-Matrix $\mathbf{M}$ abspeichern, wobei die Zeilen wiederum den Dreiecken entsprechen und der k. Eintrag der j. Zeile ($k \in \{1, 2, 3\}$, $j \in \{1, \ldots, N\}$) den Innenwinkel repräsentiert, der dem k. Eckpunkt des Dreiecks gegenüber liegt.

1. Schreiben Sie eine MATLAB-Funktion

```
function W = meshangles(x,y,T)
```

die die Winkel des durch das Tripel `(x,y,T)` beschriebenen planaren Dreiecksnetzes berechnet und in dem oben angegebenen Format in der Matrix `W` abspeichert. Wie in [1] enthalten die Spaltenvektoren `x`, `y` der Länge N die x_1- und x_2-Koordinaten der Knoten des Netzes, während die $M \times 3$-Matrix `T` die Inzidenzinformation übergibt.

Listing 1: Berechnung der Winkel im Dreiecksnetz

```
function W = meshangles(x, y, T)
% MATLAB project "Winkelschaetzung", Task 1:
% Given a triangular mesh and the set of vertices, we compute the
% angle matrix associated with that triangulation. To do so, we use
% simple trigonometric relations. Namely, we use that the cos of an
% angle between two vectors v and w is given as a
% dot(v,w)/(norm(v)*norm(w))

W = zeros(size(T));
% Vectorized implementation, making use of MATLAB's capability to
    apply
% functions to vector components
a1 = [x(T(:, 2)) - x(T(:, 1)) y(T(:, 2)) - y(T(:, 1))];
a2 = [x(T(:, 3)) - x(T(:, 2)) y(T(:, 3)) - y(T(:, 2))];
```

```
a3 = [x(T(:, 1)) - x(T(:, 3)) y(T(:, 1)) - y(T(:, 3))];
% Compute length of edges in a vectorized fashion
l1 = sqrt(a1(:,1).^2+a1(:,2).^2);
l2 = sqrt(a2(:,1).^2+a2(:,2).^2);
l3 = sqrt(a3(:,1).^2+a3(:,2).^2);
% Note that a third argument to 'dot' selects whether the Euclidean
% inner product of rows or columns of a matrix should be formed
W(:, 1) = acos(-dot(a1,a3,2)./l1.*l3);
W(:, 2) = acos(-dot(a1,a2,2)./l1.*l2);
W(:, 3) = acos(-dot(a3,a2,2)./l3.*l2);
```

2. Implementieren Sie eine MATLAB-Funktion

`function Vb = getInteriorNodes(T),`

die aus der in `T` übergebenen Inzidenzinformation für ein planares Dreiecksnetz die Indexnummern der Knoten im Innern des Netzes (also nicht auf dem Rand) berechnet und in dem Spaltenvektor `Vb` zurückgibt. Der Rand eines Netzes ist definiert in [1, Def. 2].

Hinweis: Die Funktion `processmesh` aus [1, Teilaufgabe 1] leistet gute Dienste. Wenn Ihnen die Implementierung von `getInteriorNodes` nicht gelingen sollte, dann steht auch eine verschlüsselte Version in `getInteriorNodes.p` zum Download bereit.

Listing 2: Bestimmung der Indices von Netzknoten auf dem Rand

```
function Vb = getBdNodes(T)
% MATLAB project "Winkelschätzung", Task 1:
% We want to find the indices of the boundary nodes.
% (This function is just a simple modification of processmesh.m,
% from MATLAB project II)

% Number of nodes of the mesh
N = max(max(T));
% Number of triangles of the mesh
M = size(T,1);
% Sort vertices of each triangle in ascending order
T = sort(T')';
% Again, a creative use of MATLAB's 'sparse'
C = [T(:,1) T(:,2); T(:,2) T(:,3); T(:,1) T(:,3)];
% Nonzero entries of the matrix C correspond to edges.
% Entries = 1 indicate edges on the boundary.
A = sparse(C(:,1),C(:,2),ones(3*M,1),N,N);
% Find indices of vertices belonging to edges on the boundary.
[I,J] = find(A == 1);
% The MATLAB commenand 'union' builds the union set of two arrays
% Double entries are eliminated in the process
Vb = union(I,J);
```

3. Nun sind die gemessenen Winkel leider mit Messfehlern behaftet und Sie sollen die Gausssche Methode der kleinsten Quadrate anwenden, um aus den in der Winkelmatrix $\mathbf{W}$ enthaltenen Werten Schätzungen für die tatsächlichen Werte der Winkel zu erhalten. Winkel werden generell im Bogenmass angegeben.

Dazu benutzt man wie in dem einleitenden Beispiel zu Abschnitt 3.9.1 der Vorlesung, dass die Winkel in der Ebene noch weitere lineare Beziehungen erfüllen:

(W1) Die Innenwinkel jedes Dreiecks addieren sich zu π.

(W2) Die Summe der an einem *inneren* Knoten anliegenden Winkel ist 2π.

Beschreiben Sie die Grösse und Strukur des überbestimmten linearen Gleichungssystems, das der Winkelschätzung zugrundeliegt, allgemein für ein planares Dreiecksnetz mit N Knoten und M Dreiecken. Was ist die maximale Anzahl der von Null verschiedenen Einträge seiner Koeffizientenmatrix? Um welchen Typ von Matrix handelt es sich daher?

Die folgende Konvention zur Nummerierung der Winkel soll zur Anwendung kommen: der i. Winkel im Dreieck k, also der Winkel, der dem i. Eckpunkt gegenüber liegt, hat die Nummer $3(k-1)+i$. Dies gibt auch den Index der entsprechenden Komponente im Winkelvektor.

Hinweis: Die Zeilen der Matrix lassen sich in drei verschiedene Kategorien einteilen, was eine spezifische Blockstruktur der Matrix induziert.

4. Zeigen Sie, dass die Koeffizientenmatrix aus der vorhergehenden Teilaufgabe maximalen Rang hat.

Hinweis: Hinschauen! Es reicht bereits, nur einen Block der Matrix zu betrachten.

5. Implementieren Sie eine MATLAB-Funktion

```
function A = angleestmat(T),
```

die die Koeffizientenmatrix des überbestimmten linearen Gleichungssystems aus Teilaufgabe 3 berechnet. Dabei wird die Inzidenzinformation für das planare Dreiecksnetz in `T` übergeben.

Hinweis. Das Netz kann sehr viele Dreiecke und Knoten enthalten. Vergessen Sie daher nicht, die Matrix effizient abzuspeichern. Die Funktion `getInteriorNodes` ist hier sehr nützlich.

Listing 3: Berechnung der Koeffizientenmatrix des überbestimmten Gleichungssystems für die Winkelschätzung im Dreiecksnetz

```
function A = angleestmat(T)
% MATLAB project "Winkelschaetzung", Task 5:

% We build our matrix from three blocks. The first block is simply the
% identity matrix. The second block relates all of the angles which
% belong to a triangle and should sum up to π, so it is a block
% diagonal matrix, with blocks [1 1 1] on the diagonal. The third block
```

```
% corresponds to all of the inner vertices. The sum of angles at the
% inner vertices is 2*pi, so we have to find all of the appropriate
% angles related to that inner vertex.

N = max(max(T)); % Number of nodes
M = size(T, 1);  % Number of triangles
% Buidling the second block using MATLAB's 'sparse' command
B2 = sparse(reshape(repmat(1:M, 3, 1), 3*M, 1), (1:3*M)',
   ones(3*M,1), M, 3*M);

% Building the third block
% MATLAB's 'setdiff'-command finds the indices of nodes NOT on the
% boundary (interior nodes)
inner = setdiff( 1:N, getBdNodes(T) )';

% These column vectors contain the row and column indices of the
% nonzero entries of the third block of the matrix. They are used to
% build this block using MATLAB's 'sparse' command.
I_idx = []; J_idx = [];
for k=1:M
  % For the k-th triangle the row indices are just the numbers of
  % its vertices
  I_idx = [I_idx;T(k,:)'];
  % The column indices are the numbers of its angles
  J_idx = [J_idx; 3*(k-1)+[1;2;3]];
end
% Build third block. Note that nodes on the boundary are still
% taken into account
B3 = sparse(I_idx,J_idx,ones(3*M,1),N,3*M);
% Now drop all rows corresponding to nodes on the boundary
B3 = B3(inner,:);

% Assembling the whole matrix
A = [speye(3*M); B2; B3];
```

6. In der Datei `samplemesh.dat` (einzulesen mit dem `load`-Kommando von MATLAB) stehen die Daten für ein planares Dreiecksnetz in Form des Tripels `(x,y,T)` (vgl. Teilaufgabe 1) zur Verfügung. Berechnen Sie für dieses Netz die Koeffizentenmatrix **A** des überbestimmten linearen Gleichungssystems aus Teilaufgabe 3 und die Koeffizientenmatrix der zugehörigen Normalengleichungen, siehe Formel (3.9.2.B) aus der Vorlesung. Visualisieren Sie mit Hilfe des MATLAB-Kommandos `spy()` die Struktur der beiden Matrizen, also die Lage der von Null verschiedenen Einträge. Was beobachten Sie?

Hinweis. Die Ausgabe der `spy`-Kommando für die beiden Matrizen ist in Abbildung 2

7. (Schwierig!) Wenn man sämtliche Winkel eines Dreiecksnetzes kennt, dann kann man im Prinzip dessen Geometrie bis auf Ähnlichkeit rekonstruieren. Algorithmisch kann man so vor-

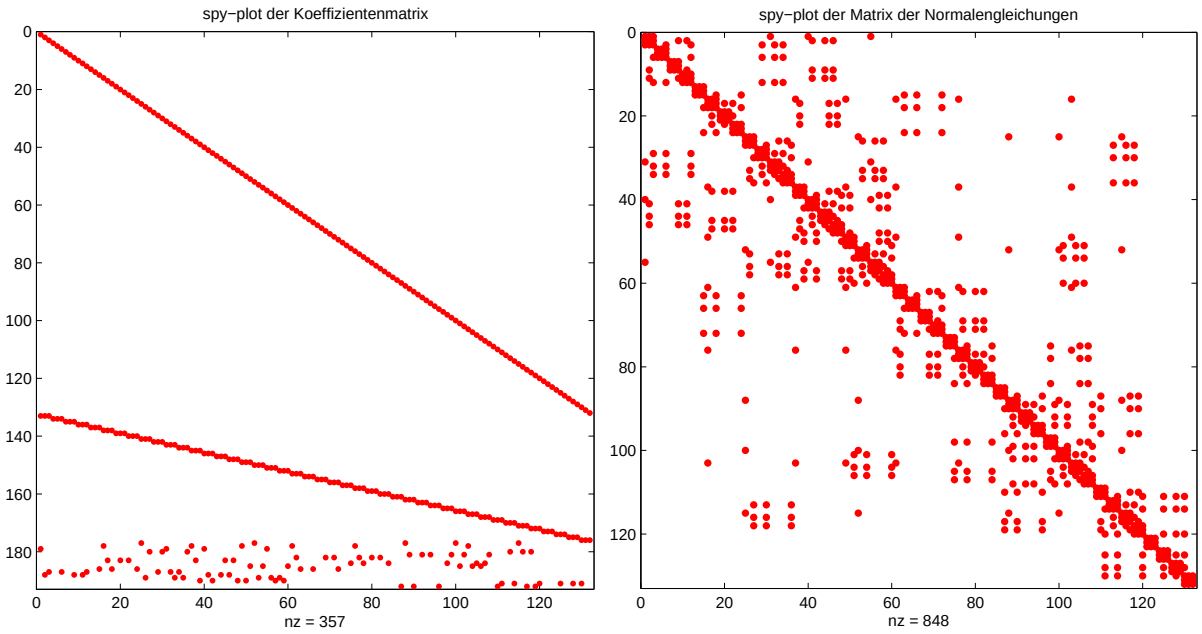

Abbildung 2: Teilaufgabe 6: Besetzungsmuster der Matrizen des unterbestimmten Gleichungssystems (links) und der zugehörigen Normalengleichungen (rechts) für das Beispielnetz aus `samplemesh.dat`

gehen, dass man mit dem ersten Dreieck beginnt, die Positionen von zwei Eckpunkten als $\binom{0}{0}$ und $\binom{1}{0}$ fixiert und dann den die Lage des dritten Eckpunkts ausrechnet. Anschliessend besucht man alle Kanten, und, wenn eine von diesen nicht auf dem Rand liegt, dann, überprüft man, ob das Nachbardreieck bereits konstruiert ist. Wenn nicht, dann wird dessen Lage bestimmt und die Position der entsprechenden Knoten gesetzt. Dann besucht man alle Kanten des neuen Dreiecks, usw. . Der Algorithmus beruht also auf **Rekursion**.

Implementieren Sie eine MATLAB-Funktion

```
function [x,y] = constructmesh(T,W),
```

die die Geometrie eines planaren Dreiecksnetzes (bis auf Ähnlichkeit) aus der Inzidenzinformation in `T` und der Winkelinformation in `W` bestimmt. Die berechneten Koordinaten der Knoten sollen in `x` und `y` zurückgegeben werden.

Hinweis. Zur Implementierung der Rekursion in MATLAB benötigen Sie noch eine Hilfsfunktion. Ausserdem ist auch hier die Funktion `processmesh` aus [1, Teilaufgabe 1] wieder sehr nützlich, denn sie liefert auch Information darüber welche Kanten auf dem Rand liegen und welche Dreiecke einer Kante benachbart sind.

8. Zu dem Dreiecksnetz aus `samplemesh.dat` enthält `W.dat` gemessene Winkel und die Datei `W_exact.dat` die (bis auf Rundungsfehler) exakten Winkel. Führen Sie für beide Datensätze die Rekonstruktion der Geometrie des Netzes mit Hilfe von `constructmesh` durch und visualisieren Sie das Ergebnis mit Hilfe der MATLAB-Funktion `triplot`, wie in [1, Listing 1]. Implementieren Sie dazu ein MATLAB-Skript **vissamplemeshes.m**

9. Benutzen Sie nun die Methode der kleinsten Quadrate, um aus der in `W.dat` enthaltenen

Information die Winkel zu schätzen. Dazu können Sie natürlich die Funktion `angleestmat` aus Teilaufgabe 5 verwenden.

Sobald Sie über die geschätzten Winkel verfügen, können Sie auch mit deren Hilfe die Geometrie des Netzes rekonstruieren. Beispiele für drei rekonstruierte Dreiecksgitter sind in Abbildung 3 gegeben.

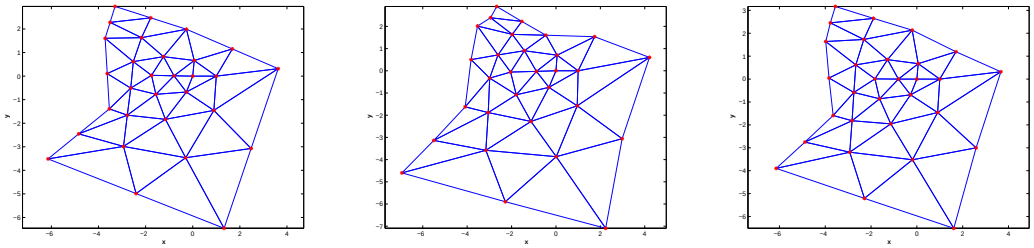

Abbildung 3: Rekonstruierte Dreiecksgitter: aus exakten Winkeln (links), aus den gemessenen Winkeln direkt (Mitte), aus der Kleinste-Quadrate-Schätzung für die gemessenen Winkel (rechts).

Erweiterung: Kleinste-Quadrate-Lösung mit exakten Nebenbedingungen

Es stellt sich heraus, dass eine gute Rekonstruktion des Dreiecksgitters eher dann möglich ist, wenn die Bedingungen **(W1)** und **(W2)** für die verwendeten Winkel mit hoher Genauigkeit erfüllt sind. Daher wollen wir jetzt diese Bedingungen erzwingen und gleichzeitig die gemessenen Winkel berücksichtigen.

Wir verwenden dazu die spezielle MATLAB-Funktion `null` deren Funktion in der MATLAB-Hilfe wie folgt beschrieben wird:

> The columns of `Z = null(A)` form an orthonormal basis for the null space of `A` obtained from the singular value decomposition. That is, `A*Z` has negligible elements, `size(Z,2)` is the nullity of `A`, and `Z'*Z = I`.

Dabei wird unter "nullity" die Dimension des Kerns von $\mathbf{A}$ verstanden. "Singular value decomposition" ist ein komplexes numerisches Verfahren, dessen Besprechung über eine Grundvorlesung weit hinausgeht. Die Funktion `null` kann natürlich wegen der unvermeidlichen Rundungsfehler den Kern nur in einer Näherung bestimmen.

Nun wollen wir erarbeiten, wie man eine Kleinste-Quadrate-Lösung für die Winkel findet, die **(W1)** und **(W2)** exakt (im Rahmen der Computerarithmetik) erfüllt.

10. Zunächst überlegen wir uns, ob die Nebenbedingungen **(W1)** und **(W2)** überhaupt noch Freiheitsgrade übriglassen, um die gemessenen Winkel zu berücksichtigen. Dazu erinnern wir uns an den *Eulerschen Polyedersatz* für planare Dreiecksnetze, deren Rand ein einziger geschlossener Polygonzug ist:

$$\sharp\{\text{Knoten}\} - \sharp\{\text{Kanten}\} + \sharp\{\text{Dreiecke}\} = 1\ . \tag{1}$$

Hier bedeutet das Symbol $\sharp$ die Anzahl der Elemente einer Menge. Sie sind eingeladen diese überraschende Formel einmal durch Nachzählen für einfache planare Dreiecksnetze zu verifizieren.

Schliessen Sie nun aus (1), dass die Anzahl K der Gleichungen für die Innenwinkel, die durch **(W1)** und **(W2)** impliziert wird, kleiner ist als die Gesamtzahl der Innenwinkel. Drücken Sie dazu zuerst K durch geeignete Anzahlen geometrischer Objekte im Dreiecksnetz aus.

11. Wir schreiben $\mathbf{w} \in \mathbb{R}^{3M}$ für einen Spaltenvektor, der die Innenwinkel des Dreiecksnetzes (in einer geeigneten Anordnung) enthält. Dann lassen sich die Bedingungen **(W1)** und **(W2)** formal ausdrücken durch

$$\mathbf{C}\mathbf{w} = Vc \quad \text{mit geeigneter Matrix/Vektor} \quad \mathbf{C} \in \mathbb{R}^{K,3M},\ \mathbf{c} \in \mathbb{R}^{K}\ . \tag{2}$$

Übringens, diese Matrix $\mathbf{C}$ war auch schon ein Teil der in Teilaufgabe 5 berechneten Matrix des überbestimmten linearen Gleichungssystems.

Wir nehmen nun an, dass uns eine Orthonormalbasis von $\mathrm{Kern}(\mathbf{C})$ zur Verfügung steht, beschafft etwa durch die MATLAB-Funktion `null`. Die Vektoren der Orthonormalbasis von $\mathrm{Kern}(\mathbf{C})$ bilden die Spalten der Matrix $\mathbf{Z} \in \mathbb{R}^{3M,L}$ mit geeignetem $L \leq K$, $L := \dim \mathrm{Kern}(\mathbf{C})$.

Begründen Sie, warum sich jeder "Winkelvektor" $\mathbf{w} \in \mathbb{R}^{3M}$, der **(W1)** und **(W2)** genügt, schreiben lässt als $\mathbf{w} = \mathbf{w}_0 + \mathbf{Z}\mathbf{v}$ mit einem Vektor $\mathbf{v} \in \mathbb{R}^{L}$, wenn $\mathbf{C}\mathbf{w}_0 = \mathbf{c}$.

11a. Wie kann man mit Hilfe der MATLAB-Funktion `qr` eine geeigneten Vektor $\mathbf{w}_0$, wie in der vorherigen Teilaufgabe definiert, berechnen?

12. Der Vektor $\mathbf{b} \in \mathbb{R}^{3M}$ enthalte nun die gemessenen Winkel. Dann ist die *Kleinste-Quadrate-Lösung* $\mathbf{w}^* \in \mathbb{R}^{3M}$ für das Problem der Winkelrekonstruktion mit exakter Berücksichtigung von **(W1)** und **(W2)** gegeben durch den Vektor

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{3M},\,\mathbf{C}\mathbf{w}=\mathbf{c}} \{\|\mathbf{w} - \mathbf{b}\|\}\ .$$

Dabei bedeutet die Notation "argmin" die Auswahl desjenigen Vektors, der den Ausdruck in den geschweiften Klammern minimal werden lässt.

Zeigen Sie, dass $\mathbf{w}^*$ erhalten werden kann als $\mathbf{w}^* = \mathbf{Z}\mathbf{v}^*$, wobei

$$\mathbf{v}^* = \operatorname*{argmin}_{\mathbf{v}\in\mathbb{R}^{L}} \{\|\mathbf{Z}\mathbf{v} + \mathbf{w}_0 - \mathbf{b}\|\}\ . \tag{3}$$

13. Aus (3) lesen wir ab, dass $\mathbf{v}^*$ die Kleinste-Quadrate-Lösung eines überbestimmten linearen Gleichungssystems ist. Was sind die Matrix und der Rechte-Seite-Vektor dieses Gleichungssystems?

Hinweis: Repetieren Sie Satz 3.9.3.B aus der Vorlesung.

14. Schreiben Sie nun eine MATLAB-Funktion

```
function W_star = angleestconstrained(T,W)
```

die die Inzidenzinformation zu dem planaren Dreiecksnetz in `T` bekommt und die gemessenen Winkel in `W`. Die Datenformate sind wie in den Teilaufgaben 5 bis 7. Die Funktion soll die geschätzten Winkel, die **(W1)** und **(W2)** exakt erfüllen, in der $M \times 3$-Matrix `W_star` zurückgeben.

Hinweis: Man erinnere sich daran, dass in MATLAB der \-Operator auch Kleinste-Quadrate-Lösungen berechnet. Natürlich kann (und soll) die Funktion `null` verwendet werden.

15. Bearbeiten Sie den Auftrag aus Teilaufgabe 8 nun für die geschätzten Winkel, die **(W1)** und **(W2)** exakt erfüllen.

Literatur

[1] Projekt *Netzglättung in MATLAB*, Vorlesung “Lineare Algebra und Numerische Mathematik”, Herbstsemester 2013.