4. Direkte Lösung linearer Gleichungssysteme

Die warten an jeder Straßenecke ...

4.1. Vorbemerkungen

Lineare Gleichungssysteme

- Ein weiteres wichtiges Einsatzgebiet numerischer Verfahren ist die **numerische lineare Algebra**, die sich mit der numerischen Lösung von Aufgaben der linearen Algebra (Matrix-Vektor-Produkt, Bestimmung von Eigenwerten, Lösung linearer Gleichungssysteme) befasst.
- Von zentraler Bedeutung ist dabei die **Lösung von Systemen linearer Gleichungen**, d.h.

  zu $A = (a_{i,j})_{1 \leq i,j \leq n} \in \mathbb{R}^{n,n}\,, \quad b = (b_i)_{1 \leq i \leq n} \in \mathbb{R}^n\,,$
  finde $x \in \mathbb{R}^n$ mit $A \cdot x = b\,,$

  die in der Numerik allgegenwärtig sind:

  - Interpolation: Konstruktion des kubischen Spline-Interpolanten (vgl. Abschnitt 2.3)
  - Randwertprobleme (boundary value problems, BVP) gewöhnlicher Differentialgleichungen (ordinary differential equations, ODE, vgl. Kapitel 6)
  - partielle Differentialgleichungen (partial differential equations, PDE)
  - ...

- hinsichtlich der Problemstellung unterscheidet man
  - **voll besetzte** Matrizen: die Anzahl der Nicht-Nullen in A ist von der Ordnung der Anzahl der Matrixeinträge überhaupt, also $O(n^2)$
  - **dünn besetzte** Matrizen: hier dominieren die Nullen die Nicht-Nullen deutlich (typischerweise $O(n)$ oder $O(n \log(n))$ Nicht-Nullen); oft haben dünn besetzte Matrizen eine **Besetzungsstruktur** (Diagonalmatrizen, Tridiagonalmatrizen ($a_{i,j} = 0$ für $\mid i - j \mid > 1$), allgemeine Bandstruktur ($a_{i,j} = 0$ für $\mid i - j \mid > c$) etc.), die das Lösen des Systems vereinfacht

$$\begin{pmatrix} * &&&&&&&& \\ & * &&&&&&& \\ && * &&&&&& \\ &&& * &&&&& \\ &&&& * &&&& \\ &&&&& * &&& \\ &&&&&& * && \\ &&&&&&& * & \\ &&&&&&&& * \end{pmatrix}$$

diagonal

$$\begin{pmatrix} * & * &&&&&&& \\ * & * & * &&&&&& \\ & * & * & * &&&&& \\ && * & * & * &&&& \\ &&& * & * & * &&& \\ &&&& * & * & * && \\ &&&&& * & * & * & \\ &&&&&& * & * & * \\ &&&&&&& * & * \end{pmatrix}$$

tridiagonal

$$\begin{pmatrix} * & * & * &&&&&& \\ * &&& * &&&&& \\ * &&&& * &&&& \\ & * &&&& * &&& \\ && * && \overleftrightarrow{2c} && * && \\ &&& * &&&& * & \\ &&&& * &&&& * \\ &&&&& * &&& * \\ &&&&&& * & * & * \end{pmatrix}$$

Band (Brandbreite $2c$)

$$\begin{pmatrix} * & * & * &&&&&& \\ * & * & * &&&&&& \\ * & * & * &&&&&& \\ &&& * & * & * &&& \\ &&& * & * & * &&& \\ &&& * & * & * &&& \\ &&&&&& * & * & * \\ &&&&&& * & * & * \\ &&&&&& * & * & * \end{pmatrix} \ldots$$

Block-diagonal

Lineare Gleichungssysteme (2)

- Hinsichtlich der Lösungsverfahren unterscheidet man
  - **direkte** Löser: liefern die (modulo Rundungsfehler) exakte Lösung x (in diesem Kapitel behandelt);
  - **indirekte** Löser: berechnen, ausgehend von einer Startnäherung $x^{(0)}$, **iterativ** eine Folge von (hoffentlich immer besseren) Näherungen $x^{(i)}$, ohne x zu erreichen (im nächsten Kapitel behandelt).
- Sinnvollerweise gehen wir im Folgenden von einer *invertierbaren* bzw. *nicht-singulären* Matrix A aus, also det$(A) \neq 0$ bzw. Rang$(A) = n$ bzw. $Ax = 0 \Leftrightarrow x = 0$.
- Zwei zunächst nahe liegende Ansätze gelten aus Komplexitätsgründen als numerische Todsünden:
  - $x := A^{-1}b$, also explizite Berechnung der Inversen von A;
  - Verwendung der *Cramerschen Regel* (über die Determinanten von A bzw. der n Matrizen, die aus A durch Ersetzen einer Spalte durch die rechte Seite b entstehen).

- Überhaupt gilt natürlich auch in der numerischen linearen Algebra: Man schaue sich die Aufgabenstellung genau an, bevor man loslegt, denn schon den einfachen Ausdruck

$$y := A \cdot B \cdot C \cdot D \cdot x \,, \qquad A, B, C, D \in \mathbb{R}^n \,, \qquad x \in \mathbb{R}^n \,,$$

kann man dumm über

$$y := (((A \cdot B) \cdot C) \cdot D) \cdot x$$

mit $O(n^3)$ Operationen (Matrix-Matrix-Produkte!) oder schlau über

$$y := A \cdot (B \cdot (C \cdot (D \cdot x)))$$

mit $O(n^2)$ Operationen (nur Matrix-Vektor-Produkte!) berechnen!

- Dies kann man sich gleich für später merken: Eine lineare Abbildung in Form einer Matrix *anwenden* zu können (d.h., ihre Wirkung auf einen beliebigen Vektor im Griff zu haben), ist i.A. viel billiger möglich als über den expliziten Aufbau der Matrix!

Vektornormen

- Für die Untersuchung der Kondition des Problems der Lösung von Systemen linearer Gleichungen sowie für die Analyse des Konvergenzverhaltens iterativer Verfahren im folgenden Kapitel benötigen wir einen Normbegriff für Vektoren und Matrizen.
- Eine **Vektornorm** ist eine Abbildung $\|.\| : \mathbb{R}^n \to \mathbb{R}$ mit den drei Eigenschaften
  - *Positivität*: $\|x\| > 0 \;\; \forall x \neq 0$;
  - *Homogenität*: $\|\alpha x\| \;=\; \mid \alpha \mid \cdot \|x\|$ für beliebiges $\alpha \in \mathbb{R}$;
  - *Dreiecksungleichung*: $\|x + y\| \leq \|x\| + \|y\|$.
- Die Menge $\{x \in \mathbb{R}^n : \;\|x\| \leq 1\}$ heißt **Normkugel** bzgl. der Norm $\|.\|$.
- Beispiele für in unserem Zusammenhang relevante Vektornormen (man verifiziere jeweils die Normeigenschaften):
  - **Summennorm**: $\|x\|_1 \;:=\; \sum_{i=1}^{n} \mid x_i \mid$
  - **euklidische Norm**: $\|x\|_2 \;:=\; \sqrt{\sum_{i=1}^{n} \mid x_i \mid^2}$ (die übliche Vektorlänge)
  - **Maximumsnorm**: $\|x\|_\infty \;:=\; \max_{1 \leq i \leq n} \mid x_i \mid$

Matrixnormen

- Ausgehend von einer Vektornorm kann man eine **Matrixnorm** definieren bzw. aus ihr **induzieren** gemäß

$$\|A\| \;:=\; \max_{\|x\|=1} \|Ax\|\,.$$

Neben den (entsprechend umformulierten) drei obigen Eigenschaften einer Vektornorm, die auch hier gelten, ist eine solche Matrixnorm

  – **sub-multiplikativ**: $\|AB\| \leq \|A\| \cdot \|B\|$;
  – **konsistent**: $\|Ax\| \leq \|A\| \cdot \|x\|$.

- Als **Konditionszahl** $\kappa(A)$ definiert man

$$\kappa(A) \;:=\; \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|}\,.$$

  – Anschaulich gibt $\kappa(A)$ an, wie stark die Normkugel durch die Matrix A bzw. durch die zugehörige lineare Abbildung verzerrt wird.
  – Im Falle der Identitätsmatrix I (Einsen in der Diagonalen, Nullen sonst) und bei bestimmten Klassen von Matrizen gibt es gar keine Verzerrung – dort gilt $\kappa(A) = 1$.
  – Für nichtsinguläres A gilt

$$\kappa(A) \;=\; \|A\| \cdot \|A^{-1}\|\,.$$

Die Kondition der Lösung linearer Gleichungssysteme

- Jetzt können wir daran gehen, die Kondition des Problems der Lösung eines linearen Gleichungssystems zu bestimmen:
  - Dazu müssen wir in
    $$(A + \delta A)(x + \delta x) \;=\; b + \delta b$$
    von Störungen $\delta A, \delta b$ der Eingaben A, b auf die Verfälschung δx des Resultats x schließen. Sinnvollerweise muss δA so klein sein, dass die gestörte Matrix invertierbar bleibt (gilt z.B. für Änderungen mit $\|\delta A\| < \|A^{-1}\|^{-1}$).
  - Löse die obige Beziehung nach δx auf und schätze mit Hilfe der Sub-Multiplikativität und Konsistenz einer induzierten Matrixnorm (dafür haben wir sie gebraucht!) ab:
    $$\|\delta x\| \;\leq\; \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|} \cdot (\|\delta b\| + \|\delta A\| \cdot \|x\|) \;.$$
  - Dividiere nun beide Seiten durch $\|x\|$ und beachte, dass die relativen Eingabestörungen $\|\delta A\|/\|A\|$ sowie $\|\delta b\|/\|b\|$ durch ε beschränkt sein sollen (wir gehen bei Konditionsuntersuchungen ja von kleinen Eingabestörungen aus).
  - Es folgt
    $$\begin{aligned} \frac{\|\delta x\|}{\|x\|} &\leq \frac{\varepsilon\kappa(A)}{1 - \varepsilon\kappa(A)} \cdot \left(\frac{\|b\|}{\|A\| \cdot \|x\|} + 1 \right) \\ &\leq \frac{2\varepsilon\kappa(A)}{1 - \varepsilon\kappa(A)} \,, \end{aligned}$$
    da $\|b\| = \|Ax\| \leq \|A\| \cdot \|x\|$.

Die Kondition der Lösung linearer Gleichungssysteme (2)

- Weil's so schön und wichtig ist, nochmals unser erzieltes Resultat zur Kondition:
$$\frac{\|\delta x\|}{\|x\|} \leq \frac{2\varepsilon\kappa(A)}{1-\varepsilon\kappa(A)}\,.$$
  - Je größer die Konditionszahl $\kappa(A)$ ist, desto größer wird unsere obere Schranke rechts für die Auswirkungen auf das Resultat, desto schlechter ist das Problem „löse $Ax = b$" somit konditioniert.
  - Der Begriff „Konditionszahl" ist also sinnvoll gewählt – er stellt eine Maßzahl für die Kondition dar.
  - Nur wenn $\varepsilon\kappa(A) \ll 1$, was eine Einschränkung an die Größenordnung der zulässigen Eingabestörungen darstellt, macht eine numerische Lösung des Problems Sinn. Dann haben wir die Kondition allerdings im Griff.
  - Auch wenn's penetrant wirkt: Dies hat nur mit dem Problem (also mit der Matrix) und nichts mit Rundungsfehlern oder näherungsweiser Rechnung zu tun!
- **Hinweis**: Die Kondition eines Problems kann man oft durch geeignete Umformulierung verbessern. Ist die Systemmatrix A in $Ax = b$ schlecht konditioniert, so kann durch Wahl eines passenden Vorfaktors M die Kondition der neuen Systemmatrix MA in $MAx = Mb$ u.U. verbessert werden.

Das Residuum

- Eine wichtige Größe ist das **Residuum** r. Zu einer Näherung $\tilde{x}$ für x wird r definiert als
$$r \;:=\; b - A\tilde{x} \;=\; A(x - \tilde{x}) \;=:\; -Ae$$
mit dem **Fehler** $e \;:=\; \tilde{x} - x$.

- **Achtung**: Fehler und Residuum können von sehr unterschiedlicher Größenordnung sein. Insb. folgt aus $r = O(\bar{\varepsilon})$ keinesfalls $e = O(\bar{\varepsilon})$ – die Korrelation enthält vielmehr noch die Konditionszahl $\kappa(A)$.

- Dennoch ist das Residuum hilfreich:
$$r \;=\; b - A\tilde{x} \qquad \Leftrightarrow \qquad A\tilde{x} \;=\; b - r$$
zeigt nämlich, dass $\tilde{x}$ bei kleinem Residuum interpretiert werden kann als *exaktes* Resultat zu leicht gestörten Eingabedaten (A original, statt b jetzt $b - r$); somit ist $\tilde{x}$ in der Sprechweise des ersten Kapitels ein **akzeptables Resultat**!

- Wir werden das Residuum vor allem bei der Konstruktion iterativer Lösungsverfahren im folgenden Kapitel einsetzen: Im Gegensatz zum unbekannten Fehler kann das Residuum in jedem Iterationsschritt stets einfach bestimmt werden.

4.2. Die Gauß-Elimination

Das Prinzip der Gauß-Elimination

- Die klassische und aus der linearen Algebra bekannte Lösungsmethode für lineare Gleichungssysteme ist die **Gauß-Elimination**, die natürliche Verallgemeinerung des Auflösens zweier Gleichungen in zwei Unbekannten:
  - Löse eine der n Gleichungen (etwa die erste) nach einer Unbekannten (etwa x_1) auf.
  - Setze den resultierenden (von $x_2, \ldots, x_n$ abhängenden) Term für x_1 in die anderen $n-1$ Gleichungen ein – aus diesen ist x_1 somit **eliminiert**.
  - Löse das resultierende System von $n-1$ Gleichungen in $n-1$ Unbekannten entsprechend und fahre fort, bis in einer Gleichung nur x_n auftaucht, das somit explizit berechnet werden kann.
  - x_n wird nun eingesetzt in die Eliminationsgleichung für x_{n-1}, womit man x_{n-1} explizit erhält.
  - Man fährt fort, bis zuletzt die Eliminationsgleichung von x_1 durch Einsetzen der (inzwischen bekannten) Werte für $x_2, \ldots, x_n$ den Wert für x_1 liefert.
- Anschaulich bedeutet das Eliminieren, dass A und b so modifiziert werden, dass in der ersten Spalte unter $a_{1,1}$ nurmehr Nullen stehen, wobei das neue System (bestehend aus der ersten Gleichung und den von x_1 befreiten restlichen Gleichungen) natürlich von demselben Vektor x gelöst wird wie das alte!

Der Algorithmus

- Diese Überlegungen ergeben den folgenden Algorithmus:

  Gauß-Elimination:

```
for j from 1 to n do
   for k from j to n do r[j,k]:=a[j,k] od;
   y[j]:=b[j];
   for i from j+1 to n do
      l[i,j]:=a[i,j]/r[j,j];
      for k from j+1 to n do a[i,k]:=a[i,k]-l[i,j]*r[j,k] od;
      b[i]:=b[i]-l[i,j]*y[j]
   od
od;
for i from n downto 1 do
   x[i]:=y[i];
   for j from i+1 to n do x[i]:=x[i]-r[i,j]*x[j] od;
   x[i]:=x[i]/r[i,i]
od;
```

Ausgangssystem

$$\begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 2 & 3 & -3 & 2 & 3 \\ 1 & 2 & 5 & 3 & -2 \\ 3 & -3 & 2 & 1 & -2 \\ 1 & 2 & 3 & -1 & 4 \end{pmatrix}, \quad \begin{pmatrix} -8 \\ -34 \\ 43 \\ 19 \\ 57 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

erste Spalte eliminiert

$$\begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 0 & -1 & 1 & 4 & 1 \\ 0 & 0 & 7 & 4 & -3 \\ 0 & -9 & 8 & 4 & -5 \\ 0 & 0 & 5 & 0 & 3 \end{pmatrix}, \quad \begin{pmatrix} -8 \\ -18 \\ 51 \\ 43 \\ 65 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

zweite Spalte eliminiert

$$\begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 0 & -1 & 1 & 4 & 1 \\ 0 & 0 & 7 & 4 & -3 \\ 0 & 0 & -1 & -32 & -14 \\ 0 & 0 & 5 & 0 & 3 \end{pmatrix}, \quad \begin{pmatrix} -8 \\ -18 \\ 51 \\ 205 \\ 65 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 9 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

dritte Spalte eliminiert

$$\begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 0 & -1 & 1 & 4 & 1 \\ 0 & 0 & 7 & 4 & -3 \\ 0 & 0 & 0 & \frac{-220}{7} & \frac{-101}{7} \\ 0 & 0 & 0 & \frac{-20}{7} & \frac{36}{7} \end{pmatrix}, \quad \begin{pmatrix} -8 \\ -18 \\ 51 \\ \frac{1486}{7} \\ \frac{200}{7} \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 9 & \frac{-1}{7} & 1 & 0 \\ 1 & 0 & \frac{5}{7} & 0 & 1 \end{pmatrix}$$

letzte Spalte eliminiert

$$R := \begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 0 & -1 & 1 & 4 & 1 \\ 0 & 0 & 7 & 4 & -3 \\ 0 & 0 & 0 & \frac{-220}{7} & \frac{-101}{7} \\ 0 & 0 & 0 & 0 & \frac{497}{77} \end{pmatrix}, \quad \begin{pmatrix} -8 \\ -18 \\ 51 \\ \frac{1486}{7} \\ \frac{714}{77} \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 9 & \frac{-1}{7} & 1 & 0 \\ 1 & 0 & \frac{5}{7} & \frac{1}{11} & 1 \end{pmatrix} =: L$$

Faktoren L und R

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 3 & 9 & \frac{-1}{7} & 1 & 0 \\ 1 & 0 & \frac{5}{7} & \frac{1}{11} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 0 & -1 & 1 & 4 & 1 \\ 0 & 0 & 7 & 4 & -3 \\ 0 & 0 & 0 & \frac{-220}{7} & \frac{-101}{7} \\ 0 & 0 & 0 & 0 & \frac{497}{77} \end{pmatrix} = \begin{pmatrix} 1 & 2 & -2 & -1 & 1 \\ 2 & 3 & -3 & 2 & 3 \\ 1 & 2 & 5 & 3 & -2 \\ 3 & -3 & 2 & 1 & -2 \\ 1 & 2 & 3 & -1 & 4 \end{pmatrix}$$

es gilt $L \cdot R = A$

Diskussion der Gauß-Elimination

- **äußere j-Schleife**: eliminiere Variablen (d.h., annulliere Subdiagonalspalten) der Reihe nach
- **erste innere k-Schleife**: speichere den im oberen Dreieck liegenden Teil von Zeile j in die Hilfsmatrix R (eliminiert wird ja nur *unterhalb* der Diagonalen); speichere die (modifizierte) rechte Seite b_j nach y_j;
  die so abgelegten $r_{j,k}$ sowie y_j werden in der Folge nicht mehr verändert!
- **innere i-Schleife**:
  - bestimme die zur Annullierung der Einträge unter der Diagonalen in Spalte j erforderlichen Faktoren und speichere sie in der Hilfsmatrix L ab (hier wird zunächst $r_{j,j} \neq 0$ stillschweigend vorausgesetzt)
  - ziehe das entsprechende Vielfache der Zeile j von Zeile i ab
  - modifiziere auch die rechte Seite entsprechend (damit Lösung x nicht verändert wird)
- zuletzt: setze rückwärts (d.h. beginnend mit x_n) die ermittelten Komponenten von x in die in R und y abgelegten Gleichungen ein (dabei wird wieder $r_{i,i} \neq 0$ stillschweigend vorausgesetzt)
- Ein sorgfältiges Zählen der arithmetischen Operationen ergibt einen Gesamtaufwand von

$$\frac{2}{3}n^3 \; + \; O(n^2)$$

arithmetischen Grundoperationen.

Diskussion der Gauß-Elimination (2)

- Neben der Matrix A kommen im Algorithmus der Gauß-Elimination die Hilfsmatrizen L und R vor. Es gilt (vgl. Algorithmus):
  - in R wird nur der obere Dreiecksteil (einschließlich der Diagonalen) besetzt;
  - in L wird nur der strikte untere Dreiecksteil besetzt (ausschließlich der Diagonalen);
  - füllt man in L die Diagonale mit lauter Einsen, so gilt der fundamentale Zusammenhang

    $$A = L \cdot R .$$

    Eine derartige **Zerlegung** oder **Faktorisierung** einer gegebenen Matrix A in Faktoren bestimmter Eigenschaften (hier: Dreiecksgestalt) ist eine ganz fundamentale Technik in der numerischen linearen Algebra.
- Für uns bedeutet obige Erkenntnis: Statt über die klassische Gauß-Elimination können wir $Ax = b$ auch über die Dreieckszerlegung $A = LR$ lösen, und zwar mit dem sich aus

  $$Ax = LRx = L(Rx) = Ly = b$$

  ergebenden Algorithmus:

- Algorithmus der LR-Zerlegung:
  - 1. Schritt: **Dreieckszerlegung**: zerlege A in Faktoren L (untere Dreiecksmatrix mit Einsen in der Diagonalen) und R (obere Dreiecksmatrix); die Zerlegung ist – mit dieser Festlegung der jeweiligen Diagonalwerte – eindeutig!
  - 2. Schritt: **Vorwärts-Substitution**: löse $Ly = b$ (durch Einsetzen, von y_1 bis y_n)
  - 3. Schritt: **Rückwärts-Substitution**: löse $Rx = y$ (durch Einsetzen, von x_n bis x_1)

Die LR-Zerlegung

- Die vorigen Überlegungen ergeben den folgenden Algorithmus:

  LR-Zerlegung:

```
for i from 1 to n do
   for k from 1 to i-1 do
      l[i,k]:=a[i,k];
      for j from 1 to k-1 do l[i,k]:=l[i,k]-l[i,j]*r[j,k] od;
      l[i,k]:=l[i,k]/r[k,k]
   od;
   for k from i to n do
      r[i,k]:=a[i,k];
      for j from 1 to i-1 do r[i,k]:=r[i,k]-l[i,j]*r[j,k] od
   od
od;
for i from 1 to n do
   y[i]:=b[i];
   for j from 1 to i-1 do y[i]:=y[i]-l[i,j]*y[j] od
od;
for i from n downto 1 do
   x[i]:=y[i];
   for j from i+1 to n do x[i]:=x[i]-r[i,j]*x[j] od;
   x[i]:=x[i]/r[i,i]
od;
```

Diskussion der LR-Zerlegung

- Um den ersten Teil (die Zerlegung in die Faktoren L und R) nachvollziehen zu können, setze man die allg. Formel für das Matrixprodukt an:

$$a_{i,k} = \sum_{j=1}^{n} l_{i,j} \cdot r_{j,k} \,.$$

Allerdings ist hier ungewöhnlicherweise A bekannt, L sowie R sind dagegen zu bestimmen (dass dies funktioniert, liegt an der Dreiecksgestalt der Faktoren)!

  - Im Falle $i > k$ muss man nur bis $j = k$ summieren und kann $l_{i,k}$ bestimmen (Gleichung nach $l_{i,k}$ auflösen: alles auf der rechten Seite ist bereits bekannt):

$$a_{i,k} = \sum_{j=1}^{k-1} l_{i,j} \cdot r_{j,k} + l_{i,k} \cdot r_{k,k} \quad \text{und damit} \quad l_{i,k} := \left(a_{i,k} - \sum_{j=1}^{k-1} l_{i,j} \cdot r_{j,k} \right) / r_{k,k} \,.$$

  - Im Falle $i \leq k$ muss man nur bis $j = i$ summieren und kann $r_{i,k}$ bestimmen (Gleichung nach $r_{i,k}$ auflösen: wieder liegen alle Größen rechts bereits vor; beachte $l_{i,i} = 1$):

$$a_{i,k} = \sum_{j=1}^{i-1} l_{i,j} \cdot r_{j,k} + l_{i,i} \cdot r_{i,k} \quad \text{und damit} \quad r_{i,k} := a_{i,k} - \sum_{j=1}^{i-1} l_{i,j} \cdot r_{j,k} \,.$$

Diskussion der LR-Zerlegung (2)

- Man sieht: Wenn man sich mit wachsenden Zeilen- und Spaltenindizes durch die Unbekannten kämpft (wie das, startend bei $i = k = 1$, im Algorithmus geschieht), so kann man alle $l_{i,k}$ und $r_{i,k}$ der Reihe nach berechnen – rechts steht immer nur bereits Berechnetes!

- Es folgen die Vorwärts-Substitution (zweite i-Schleife) und – wie zuvor bei der Gauß-Elimination – die Rückwärts-Substitution (dritte und letzte i-Schleife).

- Man kann zeigen, dass die beiden Varianten „Gauß-Elimination“ und „LR-Zerlegung“ in dem Sinne identisch sind, dass sie die gleichen Operationen ausführen (d.h. insbesondere gleicher Aufwand); nur die Reihenfolge der Operationen ist anders!

- Im Spezialfall positiv definiter Matrizen A ($A = A^T$ und $x^T Ax > 0$ für alle $x \neq 0$) geht's auch billiger als mit dem soeben gezeigten Algorithmus:

  – zerlege den Faktor R in $A = LR$ in eine Diagonalmatrix D und eine obere Dreiecksmatrix $\tilde{R}$ mit Einsen in der Diagonale (geht immer):

$$A = L \cdot R = L \cdot D \cdot \tilde{R} \qquad \text{mit } D = \text{diag}(r_{1,1}, \ldots, r_{n,n})\,;$$

– dann folgt aus der Symmetrie von A

$$A^T \;=\; (L \cdot D \cdot \tilde{R})^T \;=\; \tilde{R}^T \cdot D \cdot L^T \;=\; L \cdot D \cdot \tilde{R} \;=\; A\,,$$

und die Eindeutigkeit der Zerlegung erzwingt $L \;=\; \tilde{R}^T$ und somit

$$A \;=\; L \cdot D \cdot L^T \;=:\; \tilde{L} \cdot \tilde{L}^T\,,$$

wenn man den Diagonalfaktor D zu gleichen Teilen auf die Dreiecksfaktoren aufteilt ($\sqrt{r_{i,i}}$ in beiden Diagonalen; die $r_{i,i}$ sind alle positiv, weil A positiv definit ist).

Die Cholesky-Zerlegung

- Die soeben beschriebene Vorgehensweise, bei der man sich in der LR-Zerlegung die Berechnung der $r_{i,k}$, $i \neq k$, und damit rund die Hälfte an Rechenzeit und Speicherplatz sparen kann, heißt **Cholesky-Zerlegung**. Wir schreiben $A = LL^T$.

  Cholesky-Zerlegung:

```
for k from 1 to n do
   l[k,k]:=a[k,k];
   for j from 1 to k-1 do l[k,k]:=l[k,k]-l[k,j]^2 od;
   l[k,k]:=(l[k,k])^0.5;
   for i from k+1 to n do
      l[i,k]:=a[i,k];
      for j from 1 to k-1 do l[i,k]:=l[i,k]-l[i,j]*l[k,j] od;
      l[i,k]:=l[i,k]/l[k,k]
   od
od;
```

- Obiger Algorithmus leistet natürlich nur die Dreieckszerlegung, es müssen noch wie zuvor die Vorwärts- und die Rückwärts-Substitution ausgeführt werden, um das lineare Gleichungssystem $Ax = b$ zu lösen.

Diskussion der Cholesky-Zerlegung

- Bei der Konstruktion des Cholesky-Algorithmus gehen wir wieder von der Formel für das Matrixprodukt aus:

$$a_{i,k} \;=\; \sum_{j=1}^{n} l_{i,j} \cdot l_{j,k}^{T} \;=\; \sum_{j=1}^{k} l_{i,j} \cdot l_{j,k}^{T} \;=\; \sum_{j=1}^{k} l_{i,j} \cdot l_{k,j}\,, \qquad i \geq k\,.$$

- Daraus berechnen wir L (also das untere Dreieck) spaltenweise, in jeder Spalte beim Diagonalelement beginnend,

$$a_{k,k} \;=\; \sum_{j=1}^{k} l_{k,j}^{2}\,, \qquad l_{k,k} \;:=\; \sqrt{a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}^{2}}\,,$$

  und dann die restlichen Zeilen $i > k$ abarbeitend:

$$a_{i,k} \;=\; \sum_{j=1}^{k} l_{i,j} \cdot l_{k,j}\,, \qquad l_{i,k} \;:=\; \left(a_{i,k} - \sum_{j=1}^{k-1} l_{i,j} \cdot l_{k,j} \right) / l_{k,k}\,.$$

- Wie bereits erwähnt, sinkt der Aufwand auf ca. die Hälfte, also

$$\frac{1}{3} n^{3} + O(n^{2})\,.$$

- Zum Abschluss dieses Kapitels widmen wir uns nun noch der bisher stets stillschweigend als möglich angenommenen Division durch das Diagonalelement.

4.3. Pivot-Wahl

Pivots

- In den soeben vorgestellten Algorithmen wurde angenommen, dass die Divisionen vom Typ $a_{i,j}/r_{j,j}$ bzw. $x_i/r_{i,i}$ keine Probleme verursachen, dass also insbesondere in der Diagonalen von R keine Nullen entstehen.
- Da sich alles um diese Werte in der Diagonalen dreht, werden sie **Pivots** (frz. für Angelpunkte) genannt.
- Im positiv-definiten Fall sind alle Eigenwerte positiv, weshalb Nullen in der Diagonalen ausgeschlossen sind – im Cholesky-Verfahren ist also alles in Ordnung.
- Im allgemeinen Fall dagegen ist die Bedingung $r_{j,j} \neq 0$ für alle j allerdings nicht selbstverständlich. Falls eine Null entsteht, muss man den Algorithmus modifizieren und durch Zeilen- oder Spaltentausch eine zulässige Situation, d.h. eine Nicht-Null auf der Diagonalen erzwingen (was natürlich geht, falls A nichtsingulär ist!). Betrachte als Beispiel die Matrix

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}.$$

- Mögliche Tauschpartner für eine Null $r_{i,i}$in der Diagonalen findet man entweder in der Spalte i unterhalb der Diagonalen (**Spalten-Pivotsuche**) oder in der gesamten Restmatrix (alles ab Zeile und Spalte $i+1$, **vollständige Pivotsuche**).

Pivotsuche

- **Partial pivoting** oder **Spalten-Pivotsuche**:
  Falls $r_{i,i} = 0$, so sucht man in Spalte i unterhalb von Zeile i nach einem Eintrag $a_{k,i} \neq 0$, $k = i+1, \ldots, n$:
  - Es gibt kein solches $a_{k,i} \neq 0$: Die Restmatrix hat eine komplett verschwindende erste Spalte, was $\det(A) = 0$ bedeutet (Fehlerfall).
  - Es gibt Nicht-Nullen: Dann wählt man üblicherweise das betragsgrößte Element (es sei $a_{k,i}$) als Pivot und vertauscht in der Matrix und in der rechten Seite die Zeilen k und i.
  - Große Pivots sind günstig, weil sie zu kleinen Eliminationsfaktoren $l_{k,i}$ führen und die Einträge in L und R nicht zu stark anwachsen lassen.
  - Ein solches Zeilenvertauschen ändert natürlich das Gleichungssystem und seine Lösung überhaupt nicht!
- **Total pivoting** oder **vollständige Pivotsuche**:
  Hier sucht man nicht nur in Spalte i der Restmatrix, sondern in der gesamten Restmatrix; neben einer Zeilenvertauschung ist hier auch eine Spaltenvertauschung (Umordnung der Unbekannten x_k) zu realisieren; die vollständige Pivotsuche ist somit teurer.
- Auch wenn keine Nullen auftreten – aus numerischen Grunden ist eine Pivotsuche stets ratsam!

4.4. Anwendungsbeispiele zur direkten Lösung linearer Gleichungssysteme

Lineare Gleichungssysteme in der Informatik

- Aus der Vielzahl der Aufgabenstellungen, die auf die Lösung linearer Gleichungssysteme führen, greifen wir eine heraus: das so genannte **Radiosity-Verfahren** in der Computergrafik, mit dem fotorealistische Bilder erzeugt werden können. Besonders geeignet ist Radiosity für ambientes Licht (Dunst, Nebel, Staub im Sonnenlicht ...).
  - Die Bildgenerierung erfolgt dabei in vier Schritten: Unterteilung der gesamten Szenenoberflächen in einzelne *Patches*, Berechnung der *Formfaktoren* zwischen den Patches (beschreiben den Lichtfluss), Aufstellen und Lösen des Radiosity-Gleichungssystems (aha!) in den Radiosity-Werten (Energiedichten) als Unbekannten, Rendering (Darstellen) der Patches mit den ermittelten Radiosity-Werten.
  - Für die **Radiosity** B_i des Patches i gilt dabei die Beziehung

$$B_i \;=\; E_i + \varrho_i \cdot \sum_{j=1}^{n} B_j F_{ij}\;,$$

    wobei E_i die Emissivität des Patches (was wird an Licht neu erzeugt und emittiert – wichtig insbesondere bei Lichtquellen), ϱ_i die Absorption und F_{ij} den Formfaktor beschreiben. Kurz: Auf Patch i ist an Licht vorhanden, was dort erzeugt (emittiert) wird oder von den anderen Patches ankommt.

– Für F_{ij} gelten die Beziehungen

$$F_{ij} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2} A_j V_{ij}\,, \qquad F_{ii} = 0\,, \qquad \sum_{j=1}^{n} F_{ij} = 1\,.$$

θ_i steht dabei für den Winkel zwischen der Normalen von Patch i und dem Vektor der Länge r, der die Mittelpunkte der Patches i und j verbindet. Der Term A_j bezeichnet die Fläche von Patch j, und V_{ij} gibt die Sichtbarkeit an (1 bei freier Sicht von Patch i zu Patch j, 0 sonst).

- Obige Beziehung stellt offensichtlich ein lineares Gleichungssystem dar – n Gleichungen in den n Unbekannten B_i.