1 Kurveninterpolation

1. **Wie kann eine Wurfparabel approximiert werden?**

   LU-Faktorisierung, Cholesky-Faktorisierung,..

2. **Wie werden A und b aufgestellt?**

   b beinhaltet die Werte y_i der gegebenen Punkte. In A werden die Basisfunktionen gespeichert, also zB. Monome (wie Übung) werden in der ersten Zeile die $x_1^{0\ldots n}$ und in den Spalten $x_{1..n}$ gespeichert. Somit steht in der i-ten Zeile und j-ten Spalte der Eintrag x_{i+1}^j.

   $$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^{n-1} \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

3. **Was ist eine Basisfunktion?**

   Eine Basisfunktion beschreibt eine Kurve mit einer endliche Anzahl von Parametern. Die von uns verwendete Basisfunktion ist die Monom-Basis x^i.

4. **Wie funktioniert LU-Faktorisierung und wie löst man das Gleichungssystem?**

   Faktorisierung von A in untere (L) und obere Dreiecksmatrix (U) mittels Gauß.
   $$\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \rightarrow \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \rightarrow \ldots$$
   Aus dem Gauß ergibt sich: $(L_1 L_2 \ldots)A := U$.
   L ist dann : $(L_1^{-1} L_2^{-1} \ldots) = (L_1 L_2 \ldots)^{-1}$
   Inverse zu bilden ist hier einfach da einfach nur alle Einträge die nicht auf der Haupt-diogonalen liegen negiert werden müssen.
   Gleichungssystem ist dann: $LUx = b$
   Zum Lösen dann (1) $Ly = b$ und (2) $Ux = y$ lösen.
   Vorwärtssubstitution für (1):
   Erstes Element: $y_1 = \frac{b_1}{l_{11}}$
   $y_i = \frac{1}{l_{ii}}(b_i - \sum_{j=1}^{i-1} l_{ij} y_j)$ für $i = 2, \ldots, m$.
   Rückwärts für (2):
   $x_1 = \frac{y_m}{u_{mm}}$
   $x_i = \frac{1}{u_{ii}}(y_i - \sum_{j=i+1}^{m} u_{ij} x_j)$ für $i = m-1, \ldots, 1$.

5. **Wie kann man LU-Faktorisierung stabiler machen?**

   Pivotisierung der Matrix, mittels Permutationsmatrix. Vertausche den maximalen Eintrag a_{ik}^k der Spalte mit dem Element auf der Diagonalen a_{kk}^k.

6. **Was ist der Aufwand?**

   $PA = LU$: $\frac{2}{3}m^3$ FLOPs(Floating Point Operations).

7. **Wann geht der LU-Faktorisierung kaputt?**

   Die LU-Faktorisierung scheitert sobald der Algorithmus versucht eine Division durch 0 durchzuführen. Bsp. Matrix: $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$

2 Kurvenapproximation

8. **Was ist der Unterschied zwischen Approximation und Interpolation?**

   Polynomgrad bzw. Dimensionen der Matrix $A^{m \times n}$.

   - $m = n$: A quadratisch, voller Rang, existiert Inverse, dann eindeutige Lösung. $\Rightarrow$ Interpolation
   - $m > n$: System überbestimmt, nicht exakt lösbar. $\Rightarrow$ Approximation.

9. **Wie kann ein überbestimmtes Gleichungssystem approximiert werden?**

   Normalengleichung mit folgender LU-Faktorisierung, Cholesky-Faktorisierung, SVD,..

10. **Warum sollte man manchmal ein Polynom kleineren Grades finden?**

    Durch ein Polynom mit einem kleineren Grad können verrauschte Daten besser approximiert werden.

11. **Was ist der Grad eines Polynomes?**

    Der Grad eines Polynoms entspricht dessen höchster Potenz.

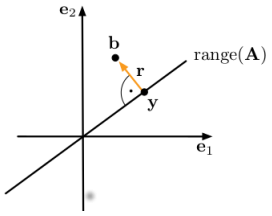
12. **Was sind Normalengleichungen und wie sind sie definiert? Welche Dimension hat A^TA?**

Erster Teil siehe nächste Frage/Antwort.
$A^TA \in (n \times n)$.

13. **Wie leitet man die Normalengleichung her?**

3 Herleitungen, geometrische:

$$
\begin{aligned}
Ax = b \text{ nicht lösbar, da } b \text{ nicht in } range(A) & \\
\Rightarrow\ & y \text{ ist orthognoale Projektion von } b \text{ auf } range(A) \\
\Rightarrow\ & r = b - Ax \\
\|r\| \to \min \Leftrightarrow\ & r \perp range(A) = \langle a_1, .., a_n \rangle \\
\Leftrightarrow\ & r \perp a_j, j = 1, .., n \\
\Leftrightarrow\ & a_j^T r = \|a_j\| \cdot \|r\| \cdot cos(a_j, r) = 0 \\
\Leftrightarrow\ & A^T r = 0 \\
\Leftrightarrow\ & A^T (b - Ax) = 0 \\
\Leftrightarrow\ & A^T b - A^T Ax = 0 \\
\Leftrightarrow\ & A^T Ax = A^T b
\end{aligned}
$$



14. **Was ist die 2-Norm?**

$||x||_2 = \sqrt{\sum_i x_i^2}$

15. **Wann ist eine Matrix symmetrisch positiv definit?**

Eine Matrix is symmetrisch wenn: $A^T = A$ und sie ist positiv definit wenn gilt: $\forall x \neq 0;\ x^T \cdot A \cdot x > 0$.

16. **Welcher Löser muss verwendet werden, wenn A nicht quadratisch ist?**

Cholesky, QR oder SVD. Je nach Eigenschaften von A.

17. **Was ist der Unterschied zwischen Cholesky-Faktorisierung und QR-Faktorisierung?**

QR-Faktorisierung	Cholesky-Faktorisierung
eher geometrisch (Basiswechsel)	eher analytisch

Die QR-Faktorisierung ist besser konditioniert, da sie auf dem geometrischen Ansatz basiert (Konditionierung: $\kappa(A)$). Die Cholesky-Faktorisierung verwendet hingegen die Normalgleichungen, welche schlecht konditioniert sind ($\kappa(A^2)$).

18. **Was ist Cholesky-Faktorisierung? Wie funktioniert sie?**

Robuster, effizienter Löser für spd Matrizen. Das Verfahren basiert auf der symmetrischen Gauß-Elimination.

$$\mathbf{A} = \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{w} & \mathbf{K} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{w} & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K} - \mathbf{w}\mathbf{w}^T \end{pmatrix} \cdot \begin{pmatrix} 1 & \mathbf{w}^T \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Dabei wird die Matrix A in die Faktoren L und L^T zerlegt. $A = \begin{pmatrix} a_{11} & w^T \\ w & K \end{pmatrix}$

$$A = \begin{pmatrix} \sqrt{a_{11}} & 0 \\ \frac{w}{\sqrt{a_{11}}} & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \tilde{L}\tilde{L}^T \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & \frac{w^T}{\sqrt{a_{11}}} \\ 0 & I \end{pmatrix} = \begin{pmatrix} \sqrt{a_{11}} & 0 \\ \frac{w}{\sqrt{a_{11}}} & \tilde{L} \end{pmatrix} \begin{pmatrix} \sqrt{a_{11}} & \frac{w^T}{\sqrt{a_{11}}} \\ 0 & \tilde{L}^T \end{pmatrix} = LL^T$$

Im Algorithmus wird $L = A$ initialisiert und das obere Dreieck auf 0 gesetzt. Mit 3 for-Schleifen (aufgrund MatLab Schreibweise) werden dann die Elemente der Spalte wie folgt berechnet: $L(i,j) = L(i,j) - L(i,k) * (L(j,k)/L(k,k))$. Die Schleifen laufen über k und j und für die Spalte i. Der Wert $x = 1/sqrt(L(k,k))$ wird separat berechnet und zum Schluss werden die Elemente der Spalte durch $L(i,k) = L(i,k) * x$ endgültig berechnet.
Zum Schluss werden die folgenden zwei Gleichungen gelöst: $Ly = b$ und $L^T x = y$.

19. **Was ist fill-in und wie kommt es zustande?**

Fill-in beschreibt, dass eine $a_{ij} = 0$ in A ist, aber nach der Faktorisierung mit Cholesky $l_{ij} \neq 0$. Fill-in tritt nur innerhalb des Bandes auf alle Nullen außerhalb des Bandes bleiben erhalten. Es ensteht durch die gebildete Differenz $K - ww^T$.

20. **Wie funktioniert QR-Faktorisierung?**

- Erst orthonormale Basis $\{q_1,\ldots,q_n\}$ von range(A) konstruieren (Gram-Schmidt)
- Dann sind Vektoren aus A darstellbar in der gefundenen Basis $\{q_1,\ldots,q_n\}$
- $a_1 = r_{11}q_1,\ldots,a_n = r_{1n}q_1 + \cdots + r_{nn}q_n$ (Gram-Schmidt liefert auch Koeffizienten r)
- In Matrixschreibweise zusammengefasst ergibt sich: $A = QR$
- Daraus ergibt sich das GL: $QRx = QQ^T b \Rightarrow Rx = Q^T b$

Für die Lösung eines Least-Squares Problems:

- QR-Faktorisierung $A = QR$ (siehe oben)
- Berechne $b' = Q^T b$
- Löse $Rx = b'$ durch Rückwärtssubstitution

21. **Was ist eine orthogonale Projektion?**

Entspricht der Projektion eines Punktes b auf das Bild einer Matrix A.
Beginne mit Projektion von b auf $\langle A\rangle$, mit $\|A\| = 1$,
$r = b - b' = (I - AA^T)b$, mit $b' = \left(\sum_{i=1}^{k} A_i A_i^T\right) b = \hat{A}\hat{A}^T b$ und $\hat{A}\hat{A}^T$ als orthogonale Projektion.

22. **Wann ist A von vollem Rang?**

Die Matrix hat den vollen Rang, wenn: $rank(A) = n \Leftrightarrow dim(range(A)) = n \Leftrightarrow dim < a_1,...,a_n > i = n \Leftrightarrow$ Spalten a_j linear unabhängig $\Leftrightarrow null(A) = 0$

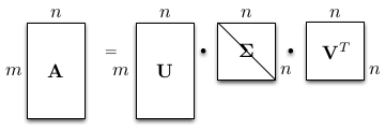
23. **Wie ist die Pseudoinverse definiert?**

$A^+ = (A^T A)^{-1} A^T$

24. **Übersichtstabelle der LGS-Löser:**

Faktorisierung	Matrix	Aufwand	Stabilität
LU	$m = n$	$\frac{2}{3}m^3$	mit Pivotisierung ok
LL^T	$m = n$, **A** s.p.d	$\frac{1}{3}m^3$	+
A^{T}Ax = A^{T}b mit LL^T	$m \geq n$, **A** voller Rang	$\frac{1}{3}n^3 + mn^2$	−
QR	$m \geq n$, **A** voller Rang	$2mn^2 - \frac{2}{3}n^3$	++
SVD	$m \geq n$	$2mn^2 + 11n^3$	++

25. **Wie sehen die SVDs aus?**



Die Matrizen U und V^T beschreiben dabei die Reflexion und Rotation von A. Σ beinhaltet nur auf der Diagonalen Elemente, die Singulärwerte, welche absteigend sortiert sind. Generell wird durch Σ die Skalierung definiert.

3 Konditionierung und finite Differenzen

26. **Was ist die Idee der Konditionierung?**

Das Problem als Funktion modellieren, welche bei Eingabe x Ausgabe $f(x)$ liefert. Ein Problem ist gut konditioniert, wenn eine kleine Pertubation (Veränderung) in x auch nur eine kleine Pertubation der Ausgabe $f(x)$ bewirkt. Umgekehrt sollte klar sein.

27. **Was bedeutete es, wenn ein Problem gut konditioniert ist?**

Ein Problem ist gut konditioniert, wenn eine Störung (Perturbation) in **x** auch nur eine kleine Störung in **f(x)** bewirkt. Dabei werden Perturbationen in relativen Änderungsraten gemessen mit: $\frac{\|\delta x\|}{\|x\|}$, $\frac{\|f(x+\delta x)-f(x)\|}{\|f(x)\|}$.

28. **Was sind finite Differenzen?**

Diskretisierung auf einem regulärem Gitter, wobei die Ableitungen approximiert werden.

(a) Vorwärtsdiff.: $f'[i] \approx \frac{f[i+1]-f[i]}{h}$

(b) Rückwärtsdiff.: $f'[i] \approx \frac{f[i]-f[i-1]}{h}$

(c) Zentrale Diff.: $f'[i] \approx \frac{f[i+1]-f[i-1]}{2h}$

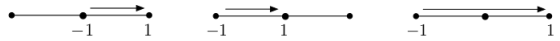


Abbildung 8.2: Approximation der ersten Ableitung mittels Vorwärtsdifferenz (links), Rückwärtsdifferenz (Mitte) und zentrale Differenz (rechts).

$\triangle u[i,j,t] = \frac{u[i-1,j,t]+u[i+1,j,t]+u[i,j-1,t]+u[i,j+1.t]-4u[i,j,t]}{h^2}$

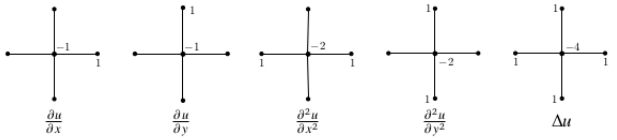


Abbildung 8.3: Schematische Darstellung der partiellen Ableitungen.

29. **Was bedeuten parabolisch, hyperbolisch und elliptisch?**

Parabolische PDEs beschreiben dynamische Gleichungen die gegen ein statisches Equilibrium konvergieren (z.B. Heat). Elliptische PDEs beschreiben statische Gleichgewichtszustände (z.B. Laplace). Hyperbolisches PDEs modellieren dynamische Systeme ohne Equilibrium (z.B. Wave).

30. **Was besagt Diskretisierung?**

Grundlegende Idee: Kurve (kontinuierlich) mit einer endlichen Zahl von Parametern zu beschreiben.
Dazu Kurve mittels Basisfunktionen φ_j und Koeffizienten f_j darstellen: $f(x) = \sum_{j=1}^{n} f_j \varphi_j(x)$.
Daraus ein LGS zusammenbasteln.

31. **Wie lässt sich die erste Ableitung mittels finiter Differenzen approximieren?**

Taylor-Entwicklung erster Ordnung $f(x) + hf'(x) + O(h^2)$ nach $f'(x)$ auflösen
$\Rightarrow f'(x) \approx \frac{f(x+h)-f(x)}{h}$
Wird $f(x)$ diskretisiert mit $f[i] = f(ih)$

$$\xRightarrow{\text{Vorwärtsdifferenz}} f'[i] = \frac{f[i+1]-f[i]}{h}$$
$$\xRightarrow{\text{Rückwärtsdifferenz}} f'[i] = \frac{f[i]-f[i-1]}{h}$$
$$\xRightarrow{\text{Zentraldifferenz}} f'[i] = \frac{f[i+1]-f[i-1]}{2h}$$

32. **Was ist der Unterschied zwischen $-\triangle u$ und $\triangle u$?**

$\triangle u[i,j,t] = \frac{u[i-1,j,t]+u[i+1,j,t]+u[i,j-1,t]+u[i,j+1.t]-4u[i,j,t]}{h^2}$: Ausgehend von den Punkten um die Mitte wird davon der mittlere Punkt 4mal abgezogen.
$-\triangle u[i,j,t] = \frac{4u[i,j,t]-u[i-1,j,t]-u[i+1,j,t]-u[i,j-1,t]-u[i,j+1.t]}{h^2}$: Ausgehen von dem mittleren Punkt, werden die umliegenden Nachbarn abgezogen.
vgl. Abb. Frage 26

4 Heat-Equation und Laplace

33. **Wie sieht die Heat-Equation aus?**

$\dot{u} = \triangle u$

34. **Was besagt $\dot{u} = \triangle u$?**

Siehe vorige Frage. Bedeutung: Fluss der Wärme in Richtung Nachbarn wird berechnet.

35. **Was ist der explizite/implizit Euler?** (Definition und Vorteile/Nachteile sowie Unterschiede (Heat))

explizit	implizit
Explizit: Berechne $v[i,j] = \triangle u = \dot{u}$ und dann $u[i,j,t+1] = u[i,j,t] + \delta t \cdot v[i,j]$ **Vorteile:** einfach zu verstehen einfach zu implementieren **Nachteile:** Instabilität bei großen Zeitsprüngen, nicht mehr δt konvergiert Fehler schaukeln stark $\rightarrow$ keine Lösung mehr möglich	Implizit: $u[i,j,t+1] = u[i,j,t] + \delta t \cdot \triangle u[i,j,t+1]$ erfordert also das Lösen des GLs:$(I - \delta t)\triangle u[t+1] = u[t]$ **Vorteile:** verwendet Zeitableitung $\dot{u}$ am nächsten Zeitpunkt $u[t+1]$ stabil für jeden Zeitschritt **Nachteile:** höherer Rechenaufwand

36. **Was passiert bei zu hohen Schritten im explizit Euler?**

Es konvergiert nicht mehr, da die Nachbarn einen zu hohen Einfluss auf die Lösung haben. Um das zu berechnen müsste man mehr Nachbarn in Betracht ziehen, da bei zu hohen Zeitschritten auch weit entfernte Nachbarn einen Einfluss auf den betrachteten Punkt haben.

37. **Was ist die CFL-Bedingung?**)

$\delta t \leq \frac{h}{||v||}$ mit $||v||$ = Informationsgeschwindigkeit. Bei der Heat-Equation: $\delta t \leq \frac{1}{4}$

38. **Wann ist der Gleichgewichtszustand erreicht?**

Der implizite Euler konvergiert immer gegen den statischen Gleichgewichtszustand der Heat-Equation, der durch die Laplace-Equation $\triangle u = 0$ beschrieben wird. Während der explizite nur bei ausreichend kleinen Zeitschritten konvergiert.

39. **Was beschreiben die Matrizen U und V in der Heat-Equation?**

In U werden die Höhen der Funktion an jedem Gridpunkt gespeichert. In V wird die Ableitung der Zeit in Bezug auf eine bestimmte Höhe U, also die Geschwindigkeit, gespeichert.

40. **Was passiert mit den Randpunkten?**

Die Randpunkte in der Heat-Equation werden nicht verändert.

41. **Wie stellt man das lineare Gleichungssystem für die Laplace-Gleichung auf?**

Jeder Punkt im Gitter ist eine Unbekannte, die berechnet werden muss. Wir haben also bei einem $(N$ x $N)$ Gitter $(N-2)$x$(N-2)$ Unbekannte (Randpunkte verändern sich nicht). Dies führt zu einem $(N-2)^2$ x $(N-2)^2$ Gleichungssystem $-Lu = b$. u ist ein Spaltenvektor in dem die neuen Werte u für das Gitter stehen. Die Reihenfolge, in der die Gitterpunkte in u stehen, führt zu einer Index Abbildung: $u_k = u(i,j)$ mit $k = idx(i,j) = (i-1) + (j-2) \cdot (N-2)$. Für die Matrix L gilt: $l_{kk} = 4, l_{kl} = -1/h^2$, wenn ($\bar{\text{i}}$, $\bar{\text{j}}$) Nachbarn von i, j und l=idx($\bar{\text{i}}$,$\bar{\text{j}}$) sonst $l_{kl} = 0$.

42. **Warum stellt man das lineare Gleichungssystem für den Laplace auf?**

Wir betrachten nur das Gleichgewicht, welches nicht mit expliziter Integration lösbar ist, da es unabhängig von der Zeit ist.

43. **Wie unterscheidet sich das Lösen des Gleichungssystems beim Laplace zu den konjugierten?**

Die konjugierten Verfahren lösen dieses iterativ, während bei der Laplace Gleichung genau eine Lösung gefunden wird. Da er unabhängig vom der Zeit ist, kann die explizite Euler Integration nicht angewendet werden.

44. **Wie löst man dieses Gl?**

Dieses Gl ist mit der Cholesky-Faktorisierung lösbar. Die Zeitkomplexität ist allerdings in diesem Fall mit $O(m^3)$ hoch, wodurch die Lösungweise durch iterative Lösungverfahren besser ist.

45. **Welche Eigenschaft hat die Laplace-Matrix?**

$-L$ hat folgende Eigenschaften:

- Die Matrix ist Sparse: Maximal 5 Einträge pro Zeile, die ungleich null sind.
- $-L$ ist symmetrisch.
- $-L$ ist positiv definit (L negativ definit).

5 Konjugierte Gradienten & Gradientenabstieg

46. **Wie funktionieren konjugierte Gradienten?**

Beliebige Suchrichtung p_n für die Minimierung von $f(x)$ zulassen:

$$\Rightarrow x_{n+1} = x_n + \alpha_n p_n,$$

wobei α_n ähnlich zum Gradientenabstieg bestimmt werden kann:

$$\frac{\delta}{\delta \alpha_n} f(x_n + \alpha_n p_n) = 0$$

$$\Rightarrow \alpha_n = \frac{p_n^T r_n}{p_n^T A p_n},$$

wenn $p_n = -\nabla f(x_n) = r_n$ äquiv. zu Gradientenabstieg

47. **Wann werden konjugierte Gradienten angewendet?**

Für große Matrizen (m>10000), da sie die Struktur ausnutzen.

48. **Was ist α?**

α ist die Schrittweite in der wir uns bei einem iterativem Löser Richtung Lösung bewegen. Im Gradientenabstieg ist $\alpha = \frac{r_n^T r_n}{r_n^T A r_n}$; in konjugierten Gradienten ist $\alpha = \frac{r_n^T r_n}{p_n^T A p_n}$.

49. **Was ist die Konvergenz des Gradientenabstiegs?**

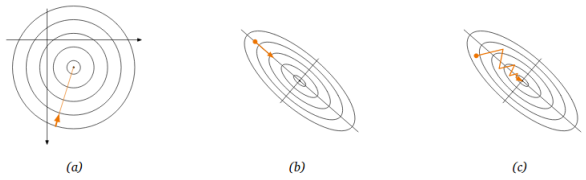


Abbildung 11.3: Konvergenz des Gradientenabstiegs. (a) Sphärische Konturlinien: Konvergenz in einem Schritt. (b) Elliptische Konturlinien, Startpunkt auf Hauptachse: Konvergenz in einem Schritt. (c) Elliptische Konturlinien, beliebiger Startpunkt: Langsame Konvergenz!

50. **Wie ist der Gradientenabstieg definiert?**

Löse $Ax = b$, mit A $s.p.d.$ formuliert als Minimierung $f(x) = \frac{1}{2}x^T Ax - b^T x \rightarrow \min$.
$f(x)$ hat ein Minimum, wenn gilt: $0 = \nabla f(x) = Ax - b$.
Minimiere $f(x)$ iterativ, indem bei jedem Schritt von x_n ein Stückchen im Paraboloid heruntergelaufen wird. Dies entspricht dem negativen Gradienten $-\nabla f(x)$.
Da $r_n = b - Ax_n = -\nabla f(x)$ folgt r_n als Abstiegsrichtung der n-ten Iteration und $x_{n+1} = x_n + \alpha_n r_n$, mit $\alpha_n = \arg\min_\alpha f(x_{n+1})$.

51. **Was ist der Unterschied zwischen dem Gradientenabstieg und konjugierten Gradienten?**

Bei dem Gradientenabstieg ist die Schrittweite abhängig von dem Resduum r_n. Die konjugierten Gradienten ersetzen r_n durch die Suchrichtung p_n, wobei jedes p_n einmalig vorkommt.

52. **Was ist eine Datenstruktur für eine dünnbesetzte Matrix?**

Eine einfache Datenstruktur für eine dünnbesetzte Matrix währe das Triple-Format wo alle Einträge $a_{ij} \neq 0$ in einem Tripel (i,j,v) gespeichert werden wobei i der Zeilenindex, j der Spaltenindex und v der Wert von a_{ij} ist. Überlicherweise gespeichert als 3 Arrays für jeweils i,j,v.

53. **Was sind die Vorteile von konjugierten Gradienten?**

- Fehler $\|e_n\|_A$ nimmt monoton ab.
- Der Algorithmus terminiert nach maximal m Iterationen.
- Aufwand $O(m^2)$, typischerweise $O(m^{1.5})$.
- Speicheraufwand $O(m)$.
- Algorithmus muss nicht direkt auf Matrix-Einträge zugreifen. Daher Nutzung einer effizienten Datenstruktur für das Matrix-Vektorprodukt Ap möglich.

54. **Was ist ein Stopkriterium von konjugierten Gradienten?**

Wenn $A^{n \times n}$ kann man nach maximal n-Schritte aufhören, ebenso wenn $\beta < \varepsilon$n mit β= der gemachte Fehler und ε der maximale Fehler.

6 Wave-Equation

55. **Wie ist die Wave-Equation definiert? Wie kann man sie als PDE erster Ordnung schreiben?**

$\ddot{u} = c^2 \triangle u$. Durch eine Hilfsfunktion $v := \dot{u}$ kann die Gleichung wie folgt modelliert werden: $\dot{u} = v\ \dot{v} = c^2 \triangle u$.

56. **Was erfordert eine doppelt so große Wellengeschwindigkeit?**

Eine doppelt so hohe Wellenggeschwindigkeit erfordert einen halb so großem Zeitschritt, damit die explizite Integration nicht instabil wird und weiterhin die CFL Bedingung erfüllt.

57. **Wie diskretisiert man die PDE's mit finiten Differenzen im Falle der Wave-Equation?**

$u[i,j,t] = u(ih, jh, t\delta t)$, $v[i,j,t] = v(ih, jh, t\delta t)$. Die PDEs hängen nun im 2-dim Gitter von Raum und Zeit ab.

58. **Wieso müssen Zeitschritte manchmal gesplittet werden?**

Um im expliziten Euler eine große Wellengeschwindigkeit zu berechnen, müssen Zeitschritte gesplittet werden.

59. **Welche Randbedingungen sind möglich?**

(a) periodisch

(b) gespiegelt

(c) Null

60. **Was ist der Unterschied zwischen implizit und explizit?** (Wave)

Explizite Integration berechnet die Werte des nächsten Zeitpunktes $u[t+1]$ aus dem momentanen Werten am Punkt, sowie dessen Nachbarn: $u_{t+1} = u_t + \delta t v_{t+1}$ und $v_{t+1} = v_t + \delta t c^2 L u_t$ (L als Laplace Matrix). Implizite Integration berechnet die Werte des nächsten Zeitpunktes $u[t+1]$ mithilfe des nächsten Zeitpunktes: $u_{t+1} = u_t + \delta t v_{t+1}$ und $v_{t+1} = v_t + \delta t c^2 L u_{t+1}$. Zum berechnen von $u[t+1]$ müssen wir das Gleichungssystem: $(I - \delta t^2 c^2 L) v_{t+1} = v_t + \delta t c^2 L u_t$ ($\Leftrightarrow$ (A)x=b) lösen.

61. **Wie sieht das lineare Gleichungssystem im impliziten Fall im Vergleich zum expliziten (Wave Equation) aus?**

Im expliziten Fall kann das Gleichungssystem in einem Schritt gelöst werden, da die Gleichungen nicht voneinander abhängen. Im impliziten Fall hängt v_{t+1} aber von u_{t+1} ab und umgekehrt, und muss daher als Gleichungssystem gelöst werden:

- $\left(I - \delta t^2 c^2 L\right) v_{t+1} = v_t + \delta t c^2 L u_t$
- Dabei ist $A = \left(I - \delta t^2 c^2 L\right)$, $x = v_{t+1}$ (gesucht) und $b = v_t \delta t c^2 L u_t$

7 Sparse Cholesky-Faktorisierung

62. **Was ist der Unterschied von Sparse Cholesky-Faktorisierung und normaler Cholesky-Faktorisierung?**

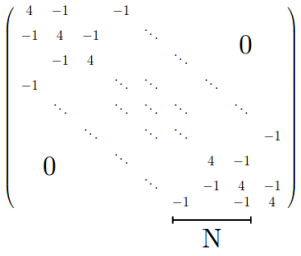
Für die Sparse Matrix A wird die Spalte j nicht mehr durch alle Spalten $k < j$ aus L modifiziert. Die j-Spalte wird nur noch durch die Spalten k modifiziert, für die $l_{jk} \neq 0$ ist.

63. **Was sind fill-ins?**

Fill-in beschreibt, dass eine $a_{ij} = 0$ in A ist, aber nach der Faktorisierung mit Cholesky $l_{ij} \neq 0$. Fill-in tritt nur innerhalb des Bandes auf alle Nullen außerhalb des Bandes bleiben erhalten. Es ensteht durch die gebildete Differenz $K - ww^T$.

64. **Was ist eine Bandmatrix?**

Eine Bandmatrix ist eine dünnbesetzte Matrix deren Nicht-Null-Einträge sich auf ein „Band" um die Diagonale der Matrix beschränken, wobei die Bandbreite $\beta(A)$ definiert ist als die maximale Distanz einer Nicht-Null von der Diagonalen der Matrix A.



65. **Kommt in der Sparse Cholesky-Faktorisierung fill-in vor?**

Ja, aber es wird durch die Permutation verhindert, da L weniger Nullen enthält.

66. **Welche Schritte muss man im Sparse Cholesky-Faktorisierung machen?**

Ändere die Reihenfolge der x_i mittels Permutationsmatrix P , so dass $\tilde{x} = Px$.
Sortiere analog A um; also $\hat{A} = AP^T$. Da A nun nicht mehr symmetrisch ist, permutiere nun auch die Zeilen von A und b respektive $\tilde{A} = P\hat{A} = PAP^T$, $\tilde{b} = Pb$.
Wende nun Cholesky auf $\tilde{A}\tilde{x} = \tilde{b}$ an.

67. **Welche Vorteile hat der Sparse Cholesky-Faktorisierung?**

Effizientes, robustes Verfahren für dünn-besetzte Matrizen. Durch eine Umsortierung der Matrix kann es keine fill-ins geben. Durch die Konstruktion von $\tilde{L}$ enthält sie weniger nicht 0en.

68. **Was ist die Band-Cholesky-Faktorisierung?**

Zunächst lege eine Band-Matrix-Datenstruktur für L an. Der Speicherverbrauch ist nun $O(mb)$ anstatt $O(m^2)$, wobei b die Bandbreite von der Bandmatrix ist. Bei der Berechnung der l_{ij} von L müssen nur Einträge $l_{ij} \in Band(L)$ berücksichtigt werden, da alle anderen Einträge null sind. Dies reduziert den Rechenaufwand von $O(m^3)$ auf $O(mb^2)$. Löse nun $Ly = b$ und $L^T x = y$ unter Ausnutzung der Bandbegrenzung von L. Dies reduziert den Aufwand von $O(m^2)$ auf $O(mb)$.

69. **Wieso macht man den Band-Cholesky-Faktorisierung und was ist der Vorteil davon?**

Geringerer Speicherplatzverbrauch und schneller (vgl. Frage davor).

70. **Was macht der Minimum degree-Algorithmus?**

Ziel: Permutation finden, sodass $LL^T = PAP^T$ möglichst wenig fill-in besitzt.
Dazu suche den Knoten v mit den wenigsten Kanten und nummeriere diesen. Des Weiteren verbinde die Nachbarn von v miteinander.

71. **Was ist der Aufwand von Sparse-Cholesky?**

Insgesamt ist der Aufwand O(m), wobei er hier stark von der Anzahl der Nicht-Nullen abhängt.

8 Optimierung

72. **Wie parallelisiert man?**

Auf einem Prozess mittels SIMD und SSE. Auf mehreren Prozessoren mittels OpenMP (1 Masterthread). Der Code von OpenMP ist:
# pragma omp parallel
id = omp_get_thread_num (); Welcher Thread ?
int nthreads = omp_get_num_threads () Wieviele?

73. **Welche Kommandozeilen Argumente?**

- -O: schaltet einfache Optimierungen ein
- -O3: mehr Optimierungen, insbesondere function inlining
- -Ofast: noch mehr Optimierungen kann aber Ergebnisse verändern!
- –march=native: Aktiviert die passenden AVX/SSE/SSE3 Instruktionen
- -fexpensive-optimizations: Noch mehr Optimierungen
- -funroll-loops Schreibt Schleifen aus.
- -ftree-vectorize: Automatische SSE-Vektorisierung

74. **Worauf muss man bei Parallelisierung achten?**

Zu beachten ist, dass manche Variablen private gesetzt werden müssenalso der Speicherzugriff eindeutig sein muss.

75. **Wie kann man Code optimieren?**

Bsp.: Call by Reference, Return by Reference, Inlinen von Funktionen.

76. **Wie hat man in den Übungen (Bildglättung) SSE sinnvoll zur Optimierung benutzt?**

SSE: Streaming SIMD Extension: SIMD Operation auf Gleitkommazahlen.
SIMD: Single Instruction Multiple Data.

Die RGBA (A für α) Werte wurden mittels SSE Vektorisierung in einen Vektor der Dimension 4 geschrieben (Datentyp 4 single precision floats, 128 Bit Register). Dadurch konnte auf ihnen, mit Hilfe von OpenMp, parallel gerechnet werden (pragma). Die einzelnen Vektoren mussten dafür private gesetzt werden. Private bedeutet, dass jeder Thread über eine eigene Kopie des Vektors (Variable) verfügt. Zum Schluss wurden die Ergebnisse in ein Ergebnisvektor reduziert.