ACA I

(Adaptive Cross Approximation)

18.06.04

Inhalt des Vortrages:

1. Einführung und Wiederholung
2. Partitionierung der Matrix
3. Unvollständige niedrigrangige Approximation

1. Einführung und Wiederholung

Wir wollen sogenannte Fredholm Integralgleichungen der Form

$$\lambda u(y) + \int_{\Gamma} \kappa(x,y)u(x)dx = f(y), y \in \Gamma \tag{1}$$

numerisch lösen, wobei Γ eine (d-1)-Mf in $\mathcal{R}^d$ ist.
3 häufig verwendete Methoden sind die Galerkin-, Kollokations- und Nyström-Methoden. Die Galerkin-Methode haben wir in frühere Vortäge bereits begegnet. Hier werden wir uns mit der Kollokationsmethode beschäftigen.

Idee: Wie bei der Galerkin-Methode wird die Lösung u approximiert durch

$$u_h = \sum_{j=1}^{N} u_j \varphi_j \in V_h,$$

wobei $\{\varphi_j\}_{j \in I}$, $I = \{1, ...N\}$ eine Basis vom Ansatzraum V_h ist.

Dazu definiert man die Träger $X_j = supp\varphi_j \subset \Gamma$

Wir reduzieren nun (1) zu ein lineares GLS der Form

$$(\lambda B + A)x = b, \quad A, B \in \mathcal{R}^{N \times N}, b \in \mathcal{R}^N, \tag{2}$$

wobei B dünn besetzt ist, A vollbesetzt und hat Einträge

$$a_{ij} = \int_\Gamma \kappa(x, y_i)\varphi_j(x)dx \tag{3}$$

wobei $y_i \in X_i$ Kollokationspunkte sind, κ asymptotisch glatt.

Wir wollen jetzt die exakte Lösung x von (2) mit $\tilde{x}$ approximieren, indem man ein gestörtes GLS löst wo A weniger aufwändig ist. Wie in früheren Vorträgen wollen wir den Kern durch Interpolation approximieren. Also werden wir ein degenerierter Kern $\tilde{\kappa}$ der Form:

$$\kappa(x,y) \approx \tilde{\kappa}(x,y) := \sum_{i=1}^{k} u_i(x) v_i(y) \tag{4}$$

konstruieren. Wie das Verfahren zur Konstruktion funktioniert wird man im Abschnitt 3 sehen.

Bei der Berechnung von a_{ij} ist der Kern nur auf $X_j \times X_i$ ausgewertet. Also folgt dass der Block $A_{t_1 t_2}$ auf $X_{t_1} \times X_{t_2}$ ausgewertet ist, wobei für $t \subset I$, $X_t = \cup_{j \in t} X_j$ definiert wird.

Falls t_1, t_2 die Verträglichkeitsbedingung

$$diam X_{t_2} \leq \eta \, dist(X_{t_1}, X_{t_2}), \eta < 1$$

erfüllt, folgt die Existenz von $\tilde{\kappa}$ aus der asymptotischen Glattheit von κ. Ersetze κ mit $\tilde{\kappa} \Rightarrow \tilde{A}_{t_1 t_2}$ ist eine Approximation von $A_{t_1 t_2}$, und $Rg(\tilde{A}_{t_1 t_2}) \leq k$, wegen $\tilde{\kappa} = \sum_{i=1}^{k} u_i(x) v_i(y)$. Man erzeugt also wiederum eine $\mathcal{H}$ -Matrix Struktur für unsere Matrix.

2. Partitionierung der Matrix

In diesem Teil werde ich Ihnen ein Algorithmus präsentieren, das die Indexmenge $I \times I$ in paarweise disjunkte Untermengen $P = \{t_1 \times t_2; t_1, t_2 \subset I\}$ unterteilt, s.d.

$$I \times I = \bigcup_{t_1 \times t_2 \in P} t_1 \times t_2$$

und für jedes Paar $t_1 \times t_2 \in P$ eine der zwei folgende Bedingungen gilt:

$$min\{\#t_1, \#t_2\} = 1 \; oder \qquad (Pa)$$

$$diam X_{t_2} \leq \eta \, dist(X_{t_1}, X_{t_2}). \qquad (Pb)$$

Um dieses Algorithmus zu verwenden müssen wir jedoch ein Paar Begriffe einführen: Sei 2^I die Potenzmenge von I. Wir betrachten eine Abbildung $S : 2^I \to 2^{2^I}, t \subset I \mapsto S(t)$, wobei $S(t)$ die Menge von paarweise disjunkten Untermengen s, s.d.

$$t = \bigcup_{s \in S(t)} s, \qquad if \, \#t > 1,$$

und $S(t) = \emptyset$ für $\#t \leq 1$. Mithilfe von S definieren wir ein Cluster Baum T indem wir als Wurzel I nehmen und rekursiv S anwenden.

Bsp: $I = \{a, b\}$
$2^I = \{\{a\}, \{b\}, \{a, b\}, \emptyset\}$,
$2^{2^I} =$
$\Big\{\{\{a\}\}, \{\{b\}\}, \{\{a, b\}\}, \{\{a\}, \{b\}\}, \{\{a\}, \{a, b\}\}, \{\{b\}, \{a, b\}\}, \emptyset\Big\}$.
Setze $S(\{a\}) = S(\{b\}) = \emptyset$ (da die Anzahl von jeweiligen Elementen ≤ 1 ist),
$S(\{a, b\}) = \big\{\{a\}, \{b\}\big\}$ (Kontrolle:$t = \{a, b\} = \{a\} \cup \{b\}$).
Bem: Hier (Fall $\#I = 2$) könnte man nur als andere Möglichkeit $S(\{a, b\}) = \big\{\{a, b\}, \emptyset\big\}$ wählen, die für die Konstruktion von einem Cluster Baum ungeeignet ist. Für $\#I \geq 3$ gibt es aber immer mehr mögliche Wähle von S. Obwohl algebraisch gesehen auch ein Element von 2^I, wird $\emptyset$ nicht als Element von I betrachtet.

Wir wollen nicht alle mögliche Partitionen P von $I \times I$ berechnen, die (P) erfüllen. Wir betrachten nur diejenige Partitionen mit Paare $t_1 \times t_2$, wobei $t_2 \in T$ liegt. Zur Konstruktion von P benötigen wir noch folgende Definition:
Für $t \in T$ sei $z_t \in X_t$ beliebig aber fest. Setze

$$\tilde{F}(t) := \left\{ i \in I : diam\, X_t \leq \frac{\eta}{1+\eta} dist(X_i, z_t) \right\},$$

$$F(t) := \{ i \in I : diam\, X_t \leq \eta\, dist(X_i, X_t) \}.$$

Es gilt: $\tilde{F}(t) \subset F(t)$.

Und nun zur Sache! Setze zuerst $P = \emptyset$ und jetzt geht los:

Algorithmus 1: *Partitioning*(t_1, t_2)

If $\min\{\#t_1, \#t_2\} = 1$ then $P := P \cup \{t_1 \times t_2\}$ // *Block* $t_1 \times t_2$ *erfüllt (Pa)*

else begin

$S_{t_2} := S(t_2)$;

for $s \in S_{t_2}$ begin

if $t_1 \cap \tilde{F}(s) \neq \emptyset$ then

$P := P \cup \{(t_1 \cap \tilde{F}(s)) \times s\}$ // *Block* $(t_1 \cap \tilde{F}(s)) \times s$ *erfüllt (Pb)*

end

if $t_1 \backslash \tilde{F}(s) \neq \emptyset$ then

Partitioning$(t_1 \backslash \tilde{F}(s), s)$

end

end

end

Bem: Der Cluster Baum muss nicht im voraus berechnet sein. S muss man aber vorgegeben sein (da nicht eindeutig).

Rechenaufwand von *Algorithmus 1*:

Die Anzahl Operationen und die Speicherplatz die benötigt sind um *Partitioning*(I, I) durchzuführen ist von der Ordnung $\eta^{-(d-1)} N \log N$. Die Anzahl von konstruierten Blöcken ist von der Ordnung N $(I = \{1, ..., N\})$.

Jetzt können wir mittels *Algorithmus 1* A in Blöcke $A_{t_1 t_2}$ für $t_1 \times t_2 \in P$ zerlegen, die die Verträglichkeitsbedingungen (P) erfüllen und dürfen uns auf ein einziges Block konzentrieren (und damit zur Abschnitt 3 gehen).

3. Unvollständige niedrigrangige Approximation

Ausgangspunkt: Wir betrachten ein Block B von unserer vollbesetzten Matrix A der die obige Verträglichkeitensbedingungen erfüllt (das können wir mittels unseres *Algorithmus 1* kriegen).
Ziel:Wir wollen ein Algorithmus entwickeln, das diesen Block durch eine Matrix vom Rang $\leq k$ approximiert. Dafür werden wir B auf folgender Art zerlegen:

$$B = U_k V_k^T + R_k,$$

wobei $U_k \in \mathcal{R}^{m \times k}$, $V_k \in \mathcal{R}^{n \times k}$ und $R_k \in \mathcal{R}^{m \times n}$. Falls die Norm R_k klein ist, ist B durch eine Matrix vom Rang $\leq k$ approximiert.

Unserer Block hat Einträge

$$b_{ij} = \int_{\Gamma} \kappa(x, y_i)\varphi_j(x)dx, \quad i = 1, ..., m, j = 1, ..., n. \tag{5}$$

Jetzt definieren wir die Funktionen $\mathcal{L}_j\kappa$ auf $\mathcal{R}^d \backslash \overline{X}_{t_1}$ als

$$(\mathcal{L}_j\kappa)(y) = \int_{\Gamma} \kappa(x, y)\varphi_j(x)dx.$$

Daraus folgt, dass $b_{ij} = (\mathcal{L}_j\kappa)(y_i)$.

Bem: Wegen asymptotischer Glattheit von κ ist $\mathcal{L}_j\kappa$ wohldefiniert.

Sei $m' \in \mathcal{N}$, $\zeta_i \in \mathcal{R}^d$, $i = 1, ...m'$. Wir definieren das $(m' + m)$-Tupel z auf folgender Weise:

$$z_i = \begin{cases} \zeta_i, & 1 \leq i \leq m' \\ y_{i-m'}, & m' < i \leq m' + m \end{cases}$$

und erweiteren B zu $\hat{B}$ durch

$$b_{ij} = (\mathcal{L}_j \kappa)(z_i), \qquad i = 1, ...m' + m, j = 1, ..., n.$$

Folgendes Algorithmus liefert uns nun Spalten von U_k und V_k.

Algorithmus 2: $k := 1; i_1 := 1$

repeat

$$(\tilde{v}_k)_j := (\mathcal{L}_j \kappa)(\zeta_{i_k}) - \sum_{l=1}^{k-1} (\hat{u}_l)_{i_k} (v_l)_j, \quad j = 1, ...n$$

if $\tilde{v}_k$ vanishes then $i_k := i_k + 1$

else begin

$j_k := argmax_{j=1,...,n} |(\tilde{v}_k)_j|; \gamma_k := (\tilde{v}_k)_{j_k}^{-1}; v_k := \gamma_k \tilde{v}_k;$

$$(\hat{u}_k)_i := (\mathcal{L}_{j_k} \kappa)(z_i) - \sum_{l=1}^{k-1} (\hat{u}_l)_i (v_l)_{j_k}, \quad i = 1, ..., m' + m$$

$i_{k+1} := i_k + 1; k := k + 1$

end

until $i_k > m'$

Bem: k hängt von m' ab. Von jetzt an ist mit k derjenige k für den $i_k > m'$ gilt gemeint. Dieses Algorithmus berechnet rekursiv folgende Spaltenvektoren: $v_1, \tilde{u}_1, v_2, \tilde{u}_2, ..., v_k, \tilde{u}_k$.
Man kann sich merken dass bei diesem Prozess $\hat{B}$ nur an den m' ersten Zeilen $\left((\tilde{v}_k)_j = \hat{b}_{i_k j} - \sum ..., i_k \leq m'\right)$ und den Spalten $j_1, ..., j_k$ $\left((\hat{u}_k)_i = \hat{b}_{i j_k} - \Sigma ...\right)$ ausgewertet wird. Man muss also nicht alle Einträge von $\hat{B}$ berechnen $\longrightarrow$ erklärt die Name *unvollständige Approximation*.

Setze

$$\hat{S}_k = \sum_{l=1}^{k} \hat{u}_l v_l^T, \hat{R}_k = \hat{B} - \hat{S}_k.$$

Nach Def. von $\hat{S}_k$ folgt sofort, $Rg(\hat{S}_k) \leq k$. Wir können sehen, dass

$$(\tilde{v}_k)_j = \hat{b}_{i_k j} - (\hat{S}_{k-1})_{i_k j} = (\hat{R}_{k-1})_{i_k j}, 1 \leq j \leq n$$
$$(\hat{u}_k)_i = \hat{b}_{i j_k} - (\hat{S}_{k-1})_{i j_k} = (\hat{R}_{k-1})_{i j_k}, 1 \leq i \leq m' + m$$

gilt.

Falls $i_l = j_l = l$ für $l = 1, ..., k$ ($i_l = l \, \forall l$ heisst, dass während der Rekursion kein $\tilde{v}_l = 0$ wird; $j_l = l \, \forall l$ heisst, dass $|(\tilde{v}_l)_l| \geq (\tilde{v}_l)_j, \forall j = 1, ...n$) gilt folgende Zerlegung:

$$\hat{R}_k = (I - \gamma_k \hat{R}_{k-1} e_k e_k^T) \hat{R}_{k-1} = L_k \hat{R}_{k-1}. \tag{6}$$

Ab jetzt wollen wir die Einträge von $\hat{R}_k$ schätzen (man will ja schliesslich zeigen, dass die Norm von $\hat{R}_k$ klein ist). Dafür konstruiert man die Funktionen $r_0(x,y) = \kappa(x,y)$, $s_0(x,y) = 0$ und für $k \geq 1$

$$r_k(x,y) = r_{k-1}(x,y) - \gamma_k(\mathcal{L}_{j_k} r_{k-1})(y) r_{k-1}(x, \zeta_{i_k}), \tag{7}$$

$$s_k(x,y) = s_{k-1}(x,y) + \gamma_k(\mathcal{L}_{j_k} r_{k-1})(y) r_{k-1}(x, \zeta_{i_k}). \tag{8}$$

Es gilt folgende Beziehung zwischen $\hat{R}_k$ und r_k:

Lemma 3: Für $1 \leq i \leq m' + m, 1 \leq j \leq n$ und $k \geq 1$ gilt $(\hat{R}_k)_{ij} = (\mathcal{L}_j r_k)(z_i)$.
Dazu gilt:
Lemma 4: Für $k \geq 0$ und $1 \leq l \leq k$ gilt

$$(\mathcal{L}_j r_k)(y) = 0, \forall y \in \mathcal{R}^d \backslash X_{t_1}.$$

Unseres nächstes Ziel ist zu zeigen, dass $\kappa = s_k + r_k \, \forall k$, und dass r_k in dieser Zerlegung nur ein Restterm ist. Dafür benötigen wir folgende Definitionen:

$$\hat{A}_k = \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & \cdots & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{bmatrix} = \begin{bmatrix} \hat{b}_{i_1 j_1} & \cdots & \hat{b}_{i_1 j_k} \\ \vdots & \cdots & \vdots \\ \hat{b}_{i_k j_1} & \cdots & \hat{b}_{i_k j_k} \end{bmatrix}.$$

Damit enthält $\hat{A}_k$ nur diejenige Elemente von $\hat{B}$, die im *Algorithmus 2* berechnet wurden.

$$\hat{A}_k(l,j) := \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_j\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_j\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{bmatrix} \in \mathcal{R}^{k\times k}$$

ist die Matrix die man erhaltet wenn man die l-te Spalte von $\hat{A}_k$ durch $(\mathcal{L}_j\kappa)([\zeta]_k)$ ersetzt, wobei

$$(\mathcal{L}_j\kappa)([\zeta]_k) := \begin{bmatrix} (\mathcal{L}_j\kappa)(\zeta_{i_1}) \\ \vdots \\ (\mathcal{L}_j\kappa)(\zeta_{i_k}) \end{bmatrix}.$$

$$M_k(l,y) := \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & \cdots & \vdots \\ (\mathcal{L}_{j_1}\kappa)(y) & \cdots & (\mathcal{L}_{j_k}\kappa)(y) \\ \vdots & \cdots & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \cdots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{bmatrix} \in \mathcal{R}^{k\times k}$$

ist die Matrix die man erhaltet wenn man die l-te Reihe von $\hat{A}_k$ durch $([\mathcal{L}]_k\kappa)(y)$ ersetzt, wobei

$$([\mathcal{L}]_k\kappa)(y) := \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(y) \\ \vdots \\ (\mathcal{L}_{j_k}\kappa)(y) \end{bmatrix}.$$

Insbesondere gilt $\hat{A}_k = \hat{A}_k(l, j_l) = M_k(l, \zeta_{i_l})$. Wir können folgende Aussagen über die Determinante von $\hat{A}_k(l, j)$ zeigen:

Lemma 5: Für $1 \leq l < k$ gilt

$$det\, \hat{A}_k(l, j) = \gamma_k^{-1} det\, \hat{A}_{k-1}(l, j) - (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\, \hat{A}_{k-1}(l, j_k),$$

$det\, \hat{A}_1(1, j) = (\mathcal{L}_j \kappa)(\zeta_{i_1})$ und $det\, \hat{A}_k(k, j) = (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) det\, \hat{A}_{k-1}$ für $k > 1$. Insbesondere gilt

$$det\, \hat{A}_k = \prod_{l=1}^{k} \gamma_l^{-1}.$$

Bem: Dieses Lemma impliziert insbesondere dass $\hat{A}_k$ regulär ist.

Jetzt können wir folgendes zeigen:

Lemma 6: Für alle $k \geq 0$ gilt

$$s_k(x,y) + r_k(x,y) = \kappa(x,y),$$

wobei für $k \geq 1$

$$s_k(x,y) = ([\mathcal{L}]_k \kappa)(y)^T \hat{A}_k^{-1} \kappa(x, [\zeta]_k).$$

Bem: Jetzt haben wir gezeigt dass $s_k + r_k = \kappa$. Aus $\hat{b}_{ij} = (\mathcal{L}_j \kappa)(z_i) = (\mathcal{L}_j s_k)(z_i) + (\mathcal{L}_j r_k)(z_i)$, folgt wegen $(\hat{R}_k)_{ij} = (\mathcal{L}_j r_k)(z_i)$ dass $(\hat{S}_k)_{ij} = (\mathcal{L}_j s_k)(z_i)$. Wie nah ist $(\mathcal{L}_j s_k)(y)$ aber zu $(\mathcal{L}_j \kappa)(y)$ (und damit $\hat{S}_k$ zu $\hat{B}$)?

Die Cramersche Regel

Die Cramersche Regel wird jetzt nützlich sein. Deshalb wiederhole ich sie kurz.

Betrachte ein lineares Gleichungssystem der Form

$$A\mathbf{x} = \mathbf{d},$$

oder

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ldots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}.$$

Setze

$$D := \begin{vmatrix} a_{11} \cdots a_{1n} \\ \vdots \quad \ldots \quad \vdots \\ a_{n1} \cdots a_{nn} \end{vmatrix}.$$

- Falls $\mathbf{d} = 0$, existiert es nichttriviale Lösungen nur wenn $D = 0$.
- Falls $\mathbf{d} \neq 0$ und $D = 0$ gibt es keine eindeutige Lösung.
- Für $\mathbf{d} \neq 0$ und $D \neq 0$ definiere nun

$$D_l = \begin{vmatrix} a_{11} \cdots a_{1(l-1)} \; d_1 \; a_{1(l+1)} \cdots a_{1n} \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{n1} \cdots a_{n(l-1)} \; d_n \; a_{n(l+1)} \cdots a_{nn} \end{vmatrix}.$$

Dann folgt:

$$x_l = \frac{D_l}{D}, 1 \leq l \leq n.$$

Begründung: Sei (a_i) die i-te Spalte. Dann gilt

$$\mathbf{d} = \sum_{i=1}^{n} (a_i) x_i.$$

Wir wissen, dass die Addition zu einer Spalte von einer Linearkombination von anderen Spalten ändert nicht die Determinante einer Matrix. Da die l-te Spalte von A aber ein Vielfaches x_l von (a_l) enthält, gilt also

$$D_l = x_l D \quad \square.$$

Mit Hilfe von der Cramerschen Regel (mit $\mathbf{d} = ([\mathcal{L}]_k \kappa)(y)$, $A = \hat{A}_k$, und betrachte das System $\mathbf{d}^T = \mathbf{x}^T A$) folgt

$$\left(([\mathcal{L}]_k \kappa)(y)^T \hat{A}_k^{-1} \right)_l = \frac{det\, M_k(l,y)}{det\, A_k},$$

und mit *Lemma 6* folgt nach Anwendung von $\mathcal{L}_j$

$$(\mathcal{L}_j s_k)(y) = \sum_{l=1}^{k} \frac{det\, M_k(l,y)}{det\, A_k} (\mathcal{L}_j \kappa)(\zeta_{i_l}).$$

Man kann noch zeigen, dass $\mathcal{L}_j s_k$ das eindeutig definierte Interpoland von $\mathcal{L}_j \kappa$ an den Stützstellen $(\zeta_{i_l})_{l=1,\ldots,k}$ im Raum aufgespannt von den Funktionen $\mathcal{L}_{j_l}\kappa$, $l = 1, ..., k$. Damit hätte man gezeigt, dass s_k eine gute Approximation für κ ist, und wird immer besser für k grösser. Also hat man die gewünschte Approximation $\hat{S}_k$ von $\hat{B}$ erhalten. Die genaue Fehlerberechnung sollte im nächste Vortrag kommen.

(happy) end