MATHEMATISCHES INSTITUT
DER UNIVERSITÄT ZU KÖLN

Wintersemester 2018/2019

Prof. Dr. A. Klawonn
J. Knepper, M. Sc.
J. Weber, M. Sc.

09. Januar 2018

12. Übung zu Wissenschaftliches Rechnen I

Aufgabe 1 (Verallgemeinertes Eigenwertproblem): (3 + 3 = 6 Punkte)
Sei A eine symmetrisch positiv semi-definite Matrix und B symmetrisch positiv definit. Betrachte Sie das verallgemeinerte Eigenwertproblem

$$Ax = \lambda Bx$$

und zeigen Sie:

(a) Alle Eigenwerte sind nicht-negativ.

(b) Es gibt eine B-orthogonale Basis $\{x_1, \dots, x_n\}$ aus Eigenvektoren, sodass $(x_i, x_j)_B = \delta_{i,j}$, wobei $\delta_{i,j}$ das Kronecker-Delta ist.

Hinweis: Verwenden Sie, dass die Zerlegung $B = B^{1/2}B^{1/2}$ existiert.

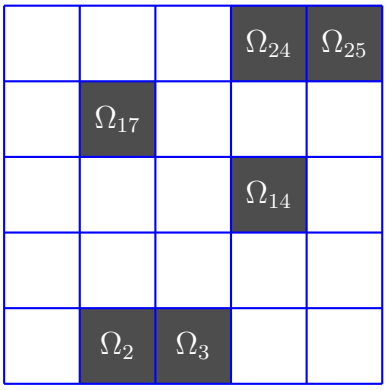
Programmieraufgabe (Adaptive FETI-DP): (35 Punkte) **Abgabe bis: 23. Jan.**
Im Folgenden betrachten wir auf $\Omega = (0,1)^2$ das Variationsproblem: Finde $u \in H_0^1(\Omega)$, sodass

$$\int_\Omega \rho \nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega v \, \mathrm{d}x \quad \forall v \in H_0^1(\Omega).$$

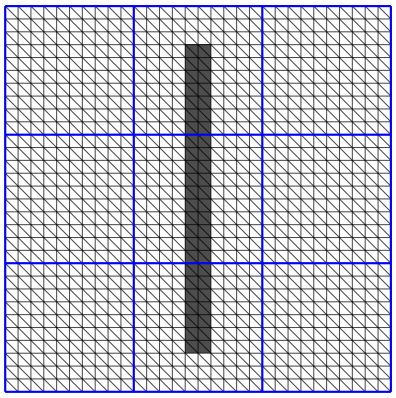
Sei Ω durch eine strukturierte Dreieckszerlegung ($P1$) diskretisiert und sei ρ konstant auf den Elementen. Wir partitionieren Ω in $N \times N$ quadratische Teilgebiete mit jeweils $2 \cdot 10^2$ Elementen.

Implementieren Sie die adaptive Berechnung der Nebenbedingungen nach Mandel und Sousedík mit dem Deflation-Ansatz in Ihr FETI-DP-Programm und testen Sie Ihre Implementierung an folgenden Problemen:

- Die Gebietszerlegungen und Koeffizientenfunktionen vom 9. Übungsblatt (inkl. der Zufallsfunktion):



Koeffizientenfunktion (i)
$N = 5$



Koeffizientenfunktion (ii)
$N = 3$

$$\rho = \begin{cases} 10^6, & \text{in } \Omega_i, i \in \{2,3,14,17,24,25\}, \\ 1, & \text{sonst.} \end{cases} \qquad \rho = \begin{cases} 10^6, & \frac{16}{30} \geq x \geq \frac{14}{30} \ \wedge \ \frac{27}{30} \geq y \geq \frac{3}{30}, \\ 1, & \text{sonst.} \end{cases}$$

Es ist $\rho = 10^6$ in dunkelgrau gefärbten Gebieten bzw. Elementen und sonst $\rho = 1$.

(iii) Zufalls-Koeffizientenfunktion mit $N = 5$, $\rho_{\min} = 1$ und $\rho_{\max} = 10^6$.

```
% Setze alle Koeffizienten der Elemente auf 'rhoMin'.
rho = rhoMin*ones(size(tri,1),1);

% Setze nun alle 'rhoMax' Koeffizienten.
rho(rand(length(rho),1) < 0.25) = rhoMax;
```

- Alle Eckknoten werden primal gewählt.
- Setzen Sie TOL $= 100$ zur Auswahl der Eigenwerte, d.h. konstruieren Sie zu jeder Eigenfunktion $w_{ij,k}$, die zu einem Eigenwert $\mu_{ij,k} \geq$ TOL gehört, den Vektor $u_{ij,k} := B_{D_{ij}} S_{ij} P_{D_{ij}} w_{ij,k}$. Diesen setzen Sie mit Null auf die restlichen Freiheitsgrade fort und schreiben ihn in eine Spalte der Matrix U, sodass in Ihrer Deflation-Implementierung die Nebenbedingungen $u_{ij,k}^T B_\Delta w = 0$ erfüllt wird.
- Wählen Sie als Abbruchkriterium das relative vorkonditionierte Residuum mit einer Toleranz von 10^{-8}.
- Startvektor $\lambda^{(0)} := 0$.

Vergleichen Sie die Anzahl Iterationen und die Konditionszahl für die folgenden Vorkonditionierer: $M^{-1} = M_D^{-1}$ (Dirichlet-Vorkonditionierer), $M^{-1} = M_{PP}^{-1}$ (Deflation mit adaptiven Nebenbedingungen). Nutzen Sie die Konditionszahlschätzung des Lanczos-Prozess im PCG-Verfahren. Geben Sie zusätzlich für die Berechnung mit M_{PP}^{-1} den Fehler $||u - u_{\text{FETI-DP}}||_2$ an, wobei Sie u über die direkte Lösung des Globalsystems, d.h. mit `u = K\b` bestimmen. Geben Sie für M_{PP}^{-1} jeweils die benötigte Anzahl an Nebenbedingungen an.

Hinweise: Siehe Implementierungshinweise im Anhang.

Allgemeine Hinweise zum Programmierteil

- Der Code **muss sinnvoll kommentiert** sein. Ein nicht kommentiertes Programm gilt als nicht erfolgreich bearbeitet.

- Das Programm muss ausführbar sein, ohne Änderungen am Code vornehmen zu müssen (d.h. ein Klick auf „Ausführen“ muss ausreichen). Schreiben Sie daher ein oder mehrere Skripte für die Teilaufgabe(n). Benennen Sie das Skript / die Skripte sinnvoll (z.B. `aufg1c.m`).
- Schreiben Sie bitte Funktionen in eigene Dateien und nicht in Skriptdateien (*Ausnahme:* anonyme Funktionen der Art `f = @(x) x.^2;`).
- Enthält Ihr Code mehrere Funktionen, so ist jede Funktion in eine eigene Datei zu schreiben. *Ausnahme:* Die Funktion wird ausschließlich von anderen Funktionen derselben Datei aufgerufen. In diesem Fall steht an oberster Stelle der Funktionsdatei die Funktion, welche von außerhalb (z.B. von einem Skript) aufgerufen wird.

Abgabe des Programmierteils

- Packen Sie Ihre Dateien in ein Archiv (Formate: .zip, oder .tar.gz) mit einem Dateinamen der Art:

  `ueb01_nachname_vorname.zip`

- Den Quellcode schicken Sie bitte an die E-Mail-Adresse Ihrer Übungsgruppenleiter / Übungsgruppenleiterinnen, mit einem Betreff der Art:

  `Betreff: Uebung1, Nachname, Vorname`

- Geben Sie bitte immer eine **ausgedruckte Version** Ihrer Programmcodes mit den schriftlichen Aufgaben ab ($\rightarrow$ Kasten), sofern dies in der Aufgabenstellung nicht eindeutig anders vermerkt wurde.
- Sofern es zur sinnvollen Lösung der Aufgabenstellung nötig ist, drucken Sie bitte auch die Ausgabe von Matlab aus. Dies sollte nicht zwei DIN-A4-Seiten überschreiten. Gleiches gilt für Grafiken.

Abgabe im entsprechenden Kasten in Raum 3.01 des Mathematischen Instituts.

- Theorie: Bis Mittwoch, 16. Januar 2019, 16:00 Uhr.
- Programmieraufgabe: Bis Mittwoch, 23. Januar 2018, 16:00 Uhr.

Anhang - Implementierungshinweise

Basierend auf dem Programm vom 11. Blatt muss „nur“ die Matrix U anders aufgestellt werden.

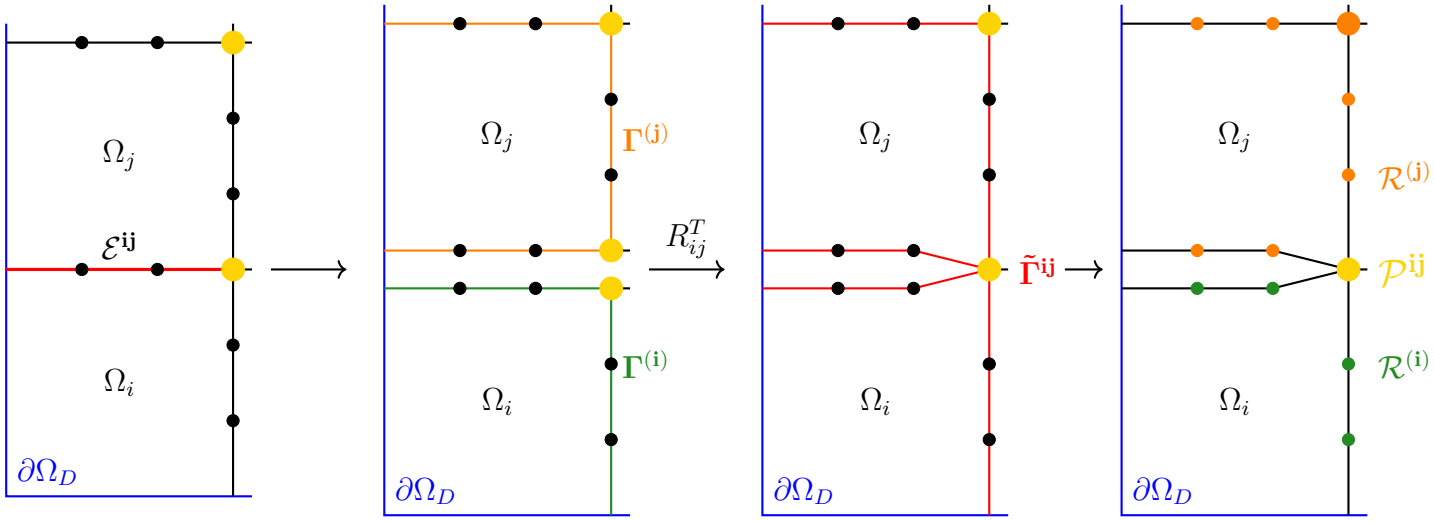
Online gibt es eine modifizierte Version der Funktion `findEdges`, die zusätzliche Informationen über die Kanten liefert.

```
[~,adjSd__e,cPrimalNodesOfEdges] =
   findEdges(cPrimal,primal,cDual,dual,l2g__sd);
```

Hierbei gibt `adjSd__e` zu jeder Kante die IDs der benachbarten Teilgebiete an und der Cell-Array `cPrimalNodesOfEdges` die primalen Knoten der Kanten in lokaler Nummerierung (bzgl. Ω_i bzw. Ω_j).

Iterieren Sie über alle Kanten, um die zugehörigen Eigenwertprobleme zu lösen.

Projektion auf $\widetilde{W}_{ij}$: Sei $\mathcal{E}^{ij}$ eine Kante zwischen den benachbarten Teilgebieten Ω_i und Ω_j. Seien zudem $\Gamma^{(i)}$ und $\Gamma^{(j)}$ die Interfaces der Teilgebiete. Mit $\tilde{\Gamma}^{ij}$ bezeichnen wir das zusammengesetzte Interface $\Gamma^{(i)} \times \Gamma^{(j)}$, welches in den primalen Knoten der Kante assembliert ist. Der Raum W_i ist auf $\Gamma^{(i)}$ definiert, W_j auf $\Gamma^{(j)}$ und $\widetilde{W}_{ij}$ auf $\tilde{\Gamma}^{ij}$. Der Operator R_{ij}^T assembliert $W_i \times W_j$ zu $\widetilde{W}_{ij}$. Im folgenden Bild gilt $R_{ij}^T \in \mathbb{R}^{12\times 13}$.



Grafiken 1–3: *Gelb: primale Knoten; Schwarz: duale Knoten*

Stellen Sie die Assemblierungsmatrizen $R_{ij}^{(i),T}$ und $R_{ij}^{(j),T}$ zur Kante auf. Dazu benötigen Sie keine Knotennummern, sondern nur die Anzahl der nicht assemblierten Interfaceknoten, die zum jeweiligen Teilgebiet gehören, und die Anzahl an assemblierten primalen Knoten. Im obigen Beispiel: $n_i = 4$, $n_j = 7$, $n_{\text{ass}} = 1$. Wir bezeichnen die zugehörigen Mengen mit $\mathcal{R}^{(i)}$, $\mathcal{R}^{(j)}$, $\mathcal{P}^{ij}$. Partitionieren Sie $R_{ij}^{(i),T}$ und $R_{ij}^{(i),T}$ nach

$$\begin{bmatrix} \mathcal{P}^{ij} \\ \mathcal{R}^{(i)} \\ \mathcal{R}^{(j)} \end{bmatrix}.$$

Sie müssen dann nur noch in die Assemblierungsmatrizen entsprechend große Einheitsmatrizen schreiben. Dann gilt

$$R_{ij} = \begin{pmatrix} R_{ij}^{(i)} \\ R_{ij}^{(j)} \end{pmatrix}$$

und

$$\Pi_{ij} = R_{ij}(R_{ij}^T R_{ij})^{-1} R_{ij}^T.$$

Listen: Sie benötigen für beide Teilgebiete noch die Listen $\mathcal{R}^{(i)}$ bzw. $\mathcal{R}^{(j)}$. Die Liste $\mathcal{P}^{ij}$ steht bereits mit `cPrimalNodesOfEdges` zur Verfügung. Stellen Sie $\mathcal{R}^{(i)}$ bzw. $\mathcal{R}^{(j)}$ auf, indem Sie einen logischen Vektor für die Knoten eines Teilgebiets erzeugen, der alle Knoten markiert, die auf dem Interface des Teilgebiets liegen, außer primale Knoten der Kante (letzeres gegeben durch `cPrimalNodesOfEdges`).

Sprung- und skalierte Sprungoperatoren: Entsprechend der Partitionierung

$$\begin{bmatrix} \mathcal{P}^{ij} \\ \mathcal{R}^{(i)} \\ \mathcal{R}^{(j)} \end{bmatrix}$$

sind die ersten n_{ass} Spalten von $B_{ij}^{(k)}$ Nullspalten. Stellen Sie daher die Sprungoperatoren auf, indem Sie $B_{\mathcal{R}^{(k)}}^{(k)}$ extrahieren und vorne die entsprechende Anzahl an Nullspalten anfügen; $\tilde{B}_{ij}^{(k)} := \left(0, B_{\mathcal{R}^{(k)}}^{(k)}\right)$.

Achtung: Sie können dies nicht über einen Befehl der Art `cB{k}(:,Gamma_k)` abkürzen, da dann die Sortierung „*assembliert / nicht-assembliert*“ nicht mehr übereinstimmt.

Fügen Sie nun die lokalen Matrizen $\tilde{B}_{ij}^{(k)}$ zusammen,

$$\tilde{B}_{ij} := \begin{bmatrix} \tilde{B}_{ij}^{(i)} & \tilde{B}_{ij}^{(j)} \end{bmatrix},$$

und löschen Sie anschließend alle Zeilen, in denen nicht genau zwei Einträge stehen. Dafür können Sie folgenden Befehl nutzen:

```
restr = (full(sum(abs(tildeB_ij),2)) == 2);
```

Dieser logische Vektor markiert nun alle Zeilen, die behalten werden müssen. Damit erhalten Sie B_{ij} bzw. vollkommen analog $B_{D,ij}$.

Sie können nun den P_D-Operator aufstellen:

$$P_{D,ij} := B_{D,ij}^T B_{ij}.$$

Schurkomplement: *Vorweg:* Der Einfachheit halber differenzieren wir nicht zwischen inneren und Dirichletrandknoten, d.h. Dirichletrandknoten sind innere Knoten.

Im Folgenden stellen Sie für beide Teilgebiete die Schurkomplemente

$$S_{\Gamma^{(k)},\Gamma^{(k)}} = K_{\Gamma^{(k)},\Gamma^{(k)}}^{(k)} - K_{\Gamma^{(k)},I^{(k)}}^{(k)} \left(K_{I^{(k)},I^{(k)}}^{(k)}\right)^{-1} K_{I^{(k)},\Gamma^{(k)}}^{(k)}, \quad k \in \{i,j\}, \tag{1}$$

auf, wobei Sie auf die Sortierung der Knoten achten müssen: Dazu stellen Sie die Schurkomplemente nicht so auf wie in (1), sondern partitioniert bzgl. $\mathcal{R}^{(k)}$ und $\mathcal{P}^{ij}$:

$$S_{\Gamma^{(k)},\Gamma^{(k)}} = \begin{pmatrix} S_{\mathcal{P}^{ij,(k)},\mathcal{P}^{ij,(k)}} & S_{\mathcal{P}^{ij,(k)},\mathcal{R}^{(k)}} \\ S_{\mathcal{P}^{ij,(k)},\mathcal{R}^{(k)}}^T & S_{\mathcal{R}^{(k)},\mathcal{R}^{(k)}} \end{pmatrix}$$

Nutzen Sie für $K^{(i)}$ die lokalen Steifigkeitsmatrizen mit den bereits eliminierten Randwerten.

Sofern Sie die zwei Schurkomplemente in einem Cell-Array gespeichert haben, können Sie mit

```
S_ij = blkdiag(S{:});
```

das zusammengesetzte Schurkomplement aufstellen. Wählen Sie

```
sigma = max(diag(S_ij));
```

für den Stabilisierungsfaktor σ.

Verallg. Eigenwertproblem und Projektion $\overline{\Pi}_{ij}$: Wir betrachten das verallg. Eigenwertproblem

$$Ax = \lambda Bx.$$

Setzen Sie

$$\begin{aligned} A &:= \Pi_{ij} P_{D,ij}^T S_{ij} P_{D,ij} \Pi_{ij}, \\ \tilde{B} &:= \Pi_{ij} S_{ij} \Pi_{ij} + \sigma(I - \Pi_{ij}). \end{aligned}$$

Sofern ein Teilgebiet am Dirichletrand liegt, ist die zugehörige lokale Steifigkeitsmatrix (mit eliminierten Randwerten) invertierbar. Folglich ist auch das zugehörige Schurkomplement invertierbar. Liegt das Teilgebiet nicht am Dirichletrand, so ist die lokale Steifigkeitsmatrix singulär und ebenso das Schurkomplement. Sofern also beide Teilgebiete Ω_i und Ω_j nicht den Dirichletrand berühren, sind die konstanten Funktionen im Kern von $S^{(i)}$ und $S^{(j)}$, d.h. der Kern von S_{ij} hat die Dimension zwei. Liegt genau ein Teilgebiet am Dirichletrand, so hat S_{ij} einen Kern der Dimension 1 und liegen beide Teilgebiete am Dirichletrand, so ist S_{ij} invertierbar. Nach Vorlesung gilt

$$\ker(\tilde{B}) = \ker(\Pi_{ij} S_{ij} \Pi_{ij} + \sigma(I - \Pi_{ij})) = \operatorname{range}(\Pi_{ij}) \cap \ker(S_{ij}).$$

Wir wissen, dass Π_{ij} eine Projektion auf $\widetilde{W}_{ij}$ ist, also den Funktionen aus $W_i \times W_j$, die stetig in den primalen Knoten der Kante sind. Damit gilt

$$\operatorname{range}(\Pi_{ij}) \cap \ker(S_{ij}) = \operatorname{span}(\{\mathbb{1}\})$$

genau dann, wenn beide Teilgebiete nicht den Dirichletrand berühren und sonst

$$\operatorname{range}(\Pi_{ij}) \cap \ker(S_{ij}) = \{0\}.$$

Für den Fall $\big(\partial\Omega_i \cup \partial\Omega_j\big) \cap \partial\Omega_D = \emptyset$ ist also $\tilde{B}$ nicht invertierbar. Damit ist das verallg. EWP $Ax = \lambda \tilde{B} x$ nicht mehr wohldefiniert. Um dieses Problem zu lösen, führen wir eine weitere Projektion $\overline{\Pi}_{ij}$ ein, um dem Kern von $\tilde{B}$ rauszuprojizieren. Diese Projektion wird also nur benötigt, wenn beide Teilgebiete nicht am Dirichletrand liegen; in allen anderen Fällen ist nichts zu tun.

Wir definieren die Projektion $\overline{\Pi}_{ij}$, sodass

$$\operatorname{range}(I - \overline{\Pi}_{ij}) = \ker(\tilde{B}) = \ker(\Pi_{ij} S_{ij} \Pi_{ij} + \sigma(I - \Pi_{ij})) = \operatorname{range}(\Pi_{ij}) \cap \ker(S_{ij}).$$

Sofern mindestens ein Teilgebiet am Dirichletrand liegt, gilt $\operatorname{range}(I - \overline{\Pi}_{ij}) = \{0\}$ und somit $\overline{\Pi}_{ij} = I$. Andernfalls gilt $\operatorname{range}(I - \overline{\Pi}_{ij}) = \operatorname{span}(\{\mathbb{1}\}) = \{s \cdot c, s \in \mathbb{R}\}$, wobei c der normierte Einsvektor ist.

Implementierung: Überprüfen Sie, ob die konstanten Funktionen im Kern von beiden Schurkomplementen liegen. Definieren Sie dazu, wie in Aufgabe 3 auf dem 11. Übungsblatt, den normierten Vektor c, der eine Basis der konstanten Funktionen ist. Überprüfen Sie nun, ob c im Kern von S_{ij} liegt, d.h. ob $||S_{ij}c||_2 < 10^{-8}$ erfüllt ist.

1. Fall: range$(I - \overline{\Pi}_{ij}) = \text{span}(\{\mathbb{1}\})$, d.h. c ist im Kern von S_{ij}:

$$B := \overline{\Pi}_{ij}\tilde{B}\overline{\Pi}_{ij} + \sigma(I - \overline{\Pi}_{ij}),$$

   wobei

$$\overline{\Pi}_{ij} := I - cc^T.$$

2. Fall: range$(I - \overline{\Pi}_{ij}) = \{0\}$:

$$B := \tilde{B}.$$

Aufgrund von Rundungsfehlern werden die Matrizen A und B nicht mehr exakt symmetrisch sein. Dies kann in Matlab dazu führen, dass der Eigenwertlöser einen anderen Algorithmus wählt und komplexe Eigenwerte berechnet. Um dies zu verhindern, re-symmetrisieren Sie die Matrizen A und B:

$$A = \tfrac{1}{2}(A + A^T)$$
$$B = \tfrac{1}{2}(B + B^T).$$

Berechnung der Eigenfunktionen und Constraint-Vektoren: Mit

```
[W_ij,D_ij] = eig(A,B);
```

können Sie die Eigenfunktion berechnen, welche auf $\tilde{\Gamma}_{ij}$ definiert sind. Die Eigenwerte stehen auf der Diagonalen von D_{ij}, d.h. `d_ij = diag(D_ij);`.

Extrahieren Sie alle Eigenfunktionen, die zu Eigenwerten gehören, die größer als die vorgegebene Toleranz `TOL=100` sind. Stellen Sie anschließend die Matrix U_{ij} (für die Constraint-Vektoren) auf, welche so viele Zeilen hat wie es Lagrangesche Multiplikatoren gibt. Mit dem oben bestimmten Vektor `restr` können Sie in `U_ij(restr,:)` die Constraint-Vektoren

$$B_{D,ij}S_{ij}P_{D,ij}W_{ij},$$

schreiben, wobei W_{ij} die ausgewählten Eigenfunktionen enthält.