Otto-von-Guericke-Universität Magdeburg
Max-Planck-Institut für Dynamik komplexer technischer Systeme
Dr. Patrick Kürschner, Janna Puderbach

Wintersemester 2017/18

Krylov-Unterraum-Verfahren für große, sparse Eigenwertprobleme

Algorithmus X.1 Lanczos-Verfahren für symmetrische EWP (SEP)

Input: $A = A^T \in \mathbb{R}^{n\times n}$, Startvektor $w \in \mathbb{R}^n$.
Output: Approximation an Eigenwert und ggf. Eigenvektor von A

1: $\beta_0 = \|v\|_2$, $v_0 := 0$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad v_k = w/\beta_{k-1}$
4: $\quad w = Av_k$
5: $\quad \alpha_k = v_k^T w (= v_k^T A v_k)$
6: $\quad w = w - \alpha_k v_k - \beta_{k-1} v_{k-1}$
7: $\quad$ Reorthogonalisiere wenn nötig.
8: $\quad \beta_k = \|w\|_2$
9: $\quad$ Löse SEP: $S_k D_k S_k^T = T_k = \begin{bmatrix} \ddots & \ddots & \\ \ddots & \ddots & \beta_k \\ & \beta_k & \end{bmatrix} = \text{tridiag}(\beta_i, \alpha_i, \beta_i)$,
$\quad D_k = \text{diag}(\theta_1, \ldots, \theta_k)$, $S_k = [s_1, \ldots, s_k]$ orthonormal.
10: $\quad$ Test auf Konvergenz, z.B. mit Residuen $\|AV_k s_{i_*} - \theta_{i_*} V_k s_{i_*}\|$, für $i_* \in \{1, \ldots, k\}$.
11: **end for**

Algorithmus X.2 Arnoldi-Verfahren für unsymmetrische EWP

Input: $A \neq A^T \in \mathbb{R}^{n\times n}$, Startvektor $w \in \mathbb{R}^n$.
Output: Approximation an Eigenwert und ggf. Eigenvektor von A.

1: $v_1 = w/\|w\|_2$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad w = Av_k$
4: $\quad$ **for** $j = 1, 2, \ldots, k$ **do** {MGS}
5: $\quad\quad h_{j,k} = v_j^T w$
6: $\quad\quad w = w - h_{j,k} v_j$
7: $\quad$ **end for**
8: $\quad$ Reorthogonalisiere wenn nötig (d.h., führe MGS erneut aus).
9: $\quad h_{k+1,k} = \|w\|$, $v_{k+1} = w/h_{k+1,k}$.
10: $\quad$ Löse EWP: $H_k = \begin{bmatrix} \ddots & \ddots & \\ \ddots & \ddots & h_{k,k-1} \\ & h_{k,k-1} & \end{bmatrix} = (h_{i,j})_{i,j=1}^k = S_k D_k S_k^{-1}$.
11: $\quad$ Test auf Konvergenz, z.B. mit Residuen $\|AV_k s_{i_*} - \theta_{i_*} V_k s_{i_*}\|$, für $i_* \in \{1, \ldots, k\}$.
12: **end for**

- $\text{span}\{V_k\} = \mathcal{K}(A, v_1, k)$ mit $AV_k = \begin{cases} V_k T_k + \beta_k v_{k+1} e_k^T & = [V_k, v_{k+1}] \begin{bmatrix} T_k \\ \beta_k e_k^T \end{bmatrix} \ : \text{Lanczos} \\ V_k H_k + h_{k+1,k} v_{k+1} e_k^T & = [V_k, v_{k+1}] \begin{bmatrix} H_k \\ h_{k+1,k} e_k^T \end{bmatrix} \ : \text{Arnoldi} \end{cases}$
- Die Eigenwerte und -vektoren der kleinen Matrizen T_k bzw. H_k bilden *Ritzwerte* θ_i und *-vektoren* $y_i := V_k s_i$. Diese werden als Approximation an Eigenpaare von A verwendet: $(\theta_i, y_i) \approx (\lambda, x)$. Hier werden meist spezielle EW gesucht, z.B., größter/kleinster Betrag, Realteil, ...

Algorithmus X.3 IRA: Implicitly Restarted Arnoldi

Input: $A \in \mathbb{R}^{n\times n}$, $v_1 \in \mathbb{R}^n$ mit $\|v_1\|_2 = 1$, $k, p \in \mathbb{N}$, $\tau > 0$.

Output: $V_k = [\, v_1, \ldots, v_k \,] \in \mathbb{R}^{n\times k}$ mit $V_k^T V_k = I_k$, so dass $\{\, v_1, \ldots, v_k \,\}$ ONB für approximativen A-invarianten Unterraum ist.

1: Führe $m = k + p$ Schritte des Arnoldi-Algorithmus (Alg. X.2) mit Startvektor v_1 durch, so dass

$$AV_m = V_m H_m + v_{m+1} e_m^T.$$

2: $\beta \leftarrow \|v_{m+1}\|_2$.

3: **while** $\beta > \tau$ **do**

4: Wähle p Shifts $\theta_1, \ldots, \theta_p \subset \Lambda(H_m)$.

5: **for** $j = 1, \ldots, p$ **do** {geshiftete QR-Iteration bzlg. H_m (Kap. VIII)}

6: $U_j R_j = H_m - \theta_j I$,

7: $H_m \leftarrow U_j^T H_m U_j$,

8: $U_j \leftarrow U_{j-1} U_j$, $(U_0 = I)$

9: **end for**

10: $\beta \leftarrow H_m(k+1, k)$, $v_{k+1} \leftarrow v_{k+1}\beta + v_{m+1} U_p(m, k)$,

11: $V_k \leftarrow V_{k+p} U_p(:, 1:k)$, $H_k := H_m(1:k, 1:k)$.

12: Erhalte neue Arnoldi-Rekursion der Länge k

$$AV_k = V_k H_k + v_{k+1} e_k^T. \tag{1}$$

13: Ausgehend von (1), führe weitere p Schritte des Arnoldi-Algorithmus (Alg. X.2) durch, so dass

$$AV_m = V_m H_m + v_{m+1} e_m^T, \quad m = k + p.$$

14: **end while**

- Die geshiftete QR-Iteration (Zeile 5-9) dient, kurz gesagt, dazu einen impliziten Neustart mit Startvektor

$$q_1 \leftarrow \rho(A - \theta_1 I)(A - \theta_2 I) \cdots (A - \theta_p I) e_1 = P(A) e_1, \ (\text{“Filter-Polynom”})$$

  effizient durch zu führen.

- Wahl der Shifts, z.B., s.d. $\{\theta_1, \ldots, \theta_p\} \subset \Lambda(H_m)$ der “unerwünschte Teil” von $\Lambda(H_m)$ ist und “herausgefiltert“ wird:

$$\Lambda(H_m) = \{\underbrace{\theta_1, \ldots, \theta_p}_{unerwünscht}, \underbrace{\theta_{p+1}, \ldots, \theta_{m+p}}_{erwünscht}\}$$

  Beispiele um die *erwünschte* Menge zu spezifizieren: k EW mit größtem/kleinstem Realteil/Betrag, ...

- IRA Konvergenz oft langsamer als Standard Arnoldi, d.h. nach q Restarts hat man $k + pq$ Arnoldivektoren berechnet, die Standard-Arnoldirekursion der Länge $k + pq$ hat aberoft bessere Ritzwete.

- $A = A^T \Rightarrow$ implicitly restarted Lanczos.

Algorithmus X.4 Golub-Kahan-Lanczos-Verfahren für SVD (vervollständigen Sie)

Input: $A \in \mathbb{R}^{n\times m}$, $n \geq m$ Startvektor $q \in \mathbb{R}^m$.
Output: Approximation an Singulärwert und ggf.-vektoren von A

$\beta_0 = 0$, $q_1 := q/\|q\|$, $q_0 = 0$.
for $k = 1, 2, \dots$ **do**
  $v_k =$
  $\alpha_k =$
  $v_k = v_k/\alpha_k$
  $q_{k+1} =$
  $\beta_k =$
  $q_{k+1} = q_{k+1}/\beta_k$
  Löse SVD: $S_k \Sigma_k T_k^T = B_k = \left[\ \diagdown\!\diagdown\ \right] = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ 0 & \ddots & \ddots & \\ & \ddots & \alpha_{k-1} & \beta_{k-1} \\ & & 0 & \alpha_k \end{bmatrix}$,
  $\Sigma_k = \operatorname{diag}(\sigma_1, \dots, \sigma_k)$, $S_k = [s_1, \dots, s_k]$, $T_k = [t_1, \dots, t_k]$ orthonormal.
  Test auf Konvergenz, z.B. mit Residuen __________________.
end for