Wiederholung: Krylov Unterraum:

$$K_m(\underline{\underline{A}}, \underline{x}_1) = \text{span}\left(\underline{x}_1, \underline{\underline{A}} \cdot \underline{x}_1, \underline{\underline{A}}^2 \cdot \underline{x}_1, \ldots, \underline{\underline{A}}^n \cdot \underline{x}_1\right)$$

Basis für viele Verfahren.

(Achtung: Wenn sich $A(\lambda)$ kontinuierlich ändert, dann ist es besser Davidson methoden zu verwenden)

Arnoldi – Methode

Hessenbergmatrix

$$H = \begin{pmatrix} h_{11} & \cdots & \cdots & h_{1n} \\ h_{21} & h_{22} & & \vdots \\ & \ddots & \ddots & \vdots \\ 0 & & h_{n\,n-1} & h_{nn} \end{pmatrix}$$

Einfachster Arnoldi

Ziel ist es eine Matrix A auf die Hessenberg-Form zu bringen:

$AV = VH$ ← Hessenbergmatrix

A, V: Orthogonal Matrix

V wird durch die erste Spalte $v_1 = \underline{\underline{V}}\, \underline{e}_1$

Teilweise schlägt der Algorithmus fehl, dann gibt es eine $n \times m$ Matrix V_m

$$A V_m - V_m H_m = 0$$

Es gibt das Residual nach m Durchgängen

$$\underline{\underline{A}} \cdot \underline{\underline{V}}_m - \underline{\underline{V}}_m \cdot \underline{\underline{H}}_m = f\, \underline{e}_m'$$

Algorithmus (analog zu Slepc technical report)

Ausgangspunkte Matrix A, iterationszahl und Ausgangsvektor v_1 (normiert!)

For $j = 1, 2, \ldots, m-1$
$w = A v_j$
w bzgl V_j orthogonalisieren Resultat $h_{1,j,j}$
$h_{j+1,j} = \|w\|_2$
If $h_{j+1,j} = 0$ stop
$v_{j+1} = w / h_{j+1,j}$
end

$f = A v_m$
f bzgl. V_m orthogonalisiert (Resultat $h_{1,j,j}$)

$\Rightarrow$ Resultat V_m, H_m und f und $\beta = \|f\|_2$
mit $A V_m - V_m H_m = f e_m^\times$

Schauen wir uns den Algorithmus an:

- Das ist im Prinzip die Konstruktion eines Krylov-Unterraums! (Iterativ Orthogonalisieren)
- Die Spalten von V_m heißen Arnoldi Vektor
- Im Falle von $h_{j+1,j} = 0$ bekommen wir einen exakten invarianten Unterraum von A (A verändert den Unterraum nicht. passiert in der Praxis nicht, endliche Genauigkeit)
- Da f per Konstruktion $\perp$ zu V_m ist, gilt $A V_m - V_m H_m = f e_m^\prime$
$\Rightarrow V_m^\times A V_m = H_m$

Erlaubt die Berechnung der Eigenpaare über Rayleigh-Ritz Approximation.
Also falls (λ_i, y_i) ein Eigenpaar von H_m ist, so ist der Ritzwert $\lambda_i = \frac{\underline{x}_i^\times \cdot A \, \underline{x}_i}{\underline{x}_i^\times \cdot \underline{x}_i}$ und $\underline{x}_i = V_m y_i$ eine Approximation des Eigenpaars.

Meist sind nur sehr wenige Approximationen brauchbar (geringe Prozentzahl).

Die Qualität kann mit der Residual Norm überprüft werden:

$$\| \underline{\underline{A}} \cdot \underline{x}_i - \lambda_i \underline{x}_i \|_2 = \| A V_m y_i - \lambda_i V_m y_i \|_2$$
$$= \| (A V_m - V_m H_m) y_i \|_2 = \beta | e_m^{\times} y_i |$$

Achtung, die Speicheranforderungen wachsen mit der Anzahl der Schritte.

Meist ist die Schrittzahl zu hoch für eine gute Konvergenz.

$\Longrightarrow$ Lösung Algorithmus neustarten.

Idee: Berechne die Arnoldi m Schritt Faktorisierung mit immer besseren initial Vektor!

Ausführung: Wir verwenden den Ritz vektor des dominanten Eigenvektor.

Wichtig: Um nicht nur einen Eigenvektor zu bekommen müssen die schon gut approximierten EV gelockt werden!

(Werden nicht mehr modifiziert!)

Also $$V_m = [\underbrace{V_{1,k}^{(l)}}_{\substack{\text{schon}\\ \text{konvergierte}\\ \text{Eigenvektoren}\\ \text{(gelockt)}}} | \underbrace{V_{k+1,m}^{(n)}}_{\substack{\text{aktiven}\\ \text{Vektoren}\\ \text{nur diese werden}\\ \text{aktualisiert.}}}]$$

Orthogonalisierung erfolgt bzgl. beider Vektorgruppen.

Arnoldi mit Deflation

Matrix A, Anzahl der Schritte m, $V_{1:k}$, H_{1k} mit $k<m$ und v_{k+1} mit Norm 1

For $j = k+1, \ldots, m-1$

$w = A v_j$

Orthogonalisierung von w bzgl. V_j (ergibt $h_{1:j,j}$)

$h_{j+1,j} = \|w\|_2$

falls $h_{j+1,j} = 0$ stopp

$v_{j+1} = w / h_{j+1,j}$

end

$f = A v_m$

f bzgl. V_m orthogonalisiert (ergibt $h_{1:m,m}$)

$\beta = \|f\|_2$

(Explizit neustartender Arnoldi)

Eingang: Matrix A, Initialvektor v_1

Dimension des Unterraums m

Normalisiere v_1

Setze $V_m = [v_1]$, $k = 0$

Restart Schleife

$m-k$ Schritte Arnoldi mit Deflation

Reduziere H_m auf (quasi) Dreiecksform $H_m \leftarrow U_1^{\dagger} H_m U_1$ (~~z.B. QR~~ z.B. QR)

Bei Entartung: Sortiere 1×1, 2×2 diagonal Blöcke: $H_m \leftarrow U_2^{\dagger} H_m U_2$

$U = U_1 U_2$

Bestimme Eigenvektoren von H_m, $H_m y_i = y_i \theta_i$

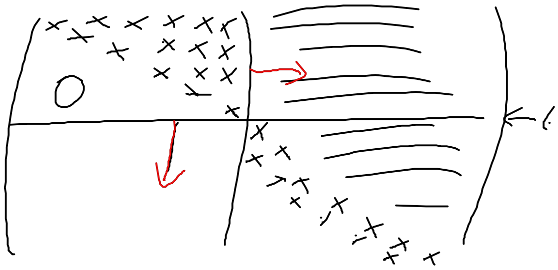
Berechne Residuum $r_i = \beta |e_m^{\dagger} y_i|$

Locke konvergierte Eigenvektoren

...

$V_m \leftrightarrow V_m U$

und A

$H_m =$ 

Wir sehen die Verfahren sind sehr kompliziert.

$\Rightarrow$ Wichtiges Verfahren Krylov Schur (konvergiert schneller), ähnlich Arnoldi.

Grundidee: Diese Verfahren bauen für große Matrizen kleine Unterräume, die die Eigenvektoren gut approximieren.

$\Rightarrow$ Wichtiges Resultat, Forschungsfeld für sich, benutze fertige Solver für eigene wissenschaftliche Projekte!

II. 6) Preconditioning von Matrizen

Problem bei iterativen Verfahren

Die Konvergenz kann sehr langsam sein.

Meist hängt die Konvergenz von der Konditionszahl ab.

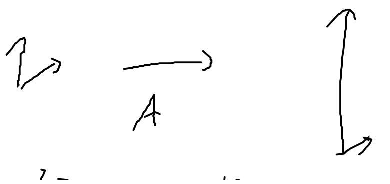
$$\kappa(A) = \left|\frac{\lambda_{max}(A)}{\lambda_{min}(A)}\right|$$

$\leftarrow$ größter Eigenwert

$\leftarrow$ kleinster Eigenwert

Falls $\kappa(A) \gg 1$, schlecht konditioniertes Problem (Kontext Gleichungssystem)

Problem: schlecht konditionierte Systeme konvergieren schlecht.



hohe numerische Ungenauigkeit.

$\Rightarrow$: Lösung: Preconditioning (Vorkonditionierung) geschieht jeweils Problemabhängig und verändert Spektrum.

Die Idee ist es, dass z.B. ein Gleichungssystem

$\underline{\underline{A}} \cdot \underline{x} = \underline{b}$ mit einer regulären Matrix (invertierbar) $\underline{\underline{M}} \cdot \underline{\underline{A}} \cdot \underline{x} = \underline{\underline{M}} \cdot \underline{b}$

umgeformt werden, bzw.

$$\underline{\underline{A}} \cdot \underline{\underline{M}} \cdot \underline{y} = \underline{b} \quad \text{mit} \quad \underline{y} = \underline{\underline{M}}^{-1} \cdot \underline{x}$$

Wichtig: M wird so konstruiert, dass sich ihr Inverses gut approximieren lässt.

$\Rightarrow$ Ziel schnelle Konvergenz

Beispiele

- Jacobi Preconditioner

· Falls die Diagonale dominant ist (z.B. part DGL)

$$M = \mathrm{diag}(A) \quad \text{und} \quad M_{ij}^{-1} = \frac{\delta_{ij}}{A_{ij}} \qquad (A_{ij} \neq 0!)$$

- Unvollständige Cholesky Faktorisierung
- Unvollständige LU Faktorisierung ...

$\Rightarrow$ Wenn Konvergenz zu langsam $\Rightarrow$ Schaut nach dem Preconditioner.