Iteratives Lösen linearer Gleichungssysteme
Übung 3 - Beispiel 1

Michael Neunteufel

7. Mai 2017

1. Beispiel

Geg.: Sei $A > 0$ und symmetrisch, $\phi(x) = \frac{1}{2}(Ax, x) - (b, x)$ die zu minimierende quadratische Form des CG-Verfahrens und x_* die exakte Lösung von $Ax = b$.

a)

Z.z.:

$$\phi(x) - \phi(x_*) = \frac{1}{2}||x - x_*||_A^2$$

Beweis:

$$\begin{aligned}
||x - x_*||_A^2 &= \langle A(x - x_*), (x - x_*)\rangle \\
&= \langle Ax, x\rangle - \langle Ax, x_*\rangle - \langle Ax_*, x\rangle + \langle Ax_*, x_*\rangle \\
&\overset{A \text{ sym.}, \langle\cdot,\cdot\rangle \text{ sym.}}{=} \langle Ax, x\rangle - 2\langle Ax_*, x\rangle + \langle Ax_*, x_*\rangle \\
&\overset{Ax_*=b}{=} \langle Ax, x\rangle - 2\langle b, x\rangle + \langle Ax_*, x_*\rangle \\
&= \langle Ax, x\rangle - 2\langle b, x\rangle - \langle Ax_*, x_*\rangle + 2\langle \underbrace{Ax_*}_{b}, x_*\rangle \\
&= 2(\phi(x) - \phi(x_*))
\end{aligned}$$

b)

Sei $(x_0, x_1, .., x_k)$ Folge von CG-Iterierten, x_{k+1} die nächste CG-Iterierte und $\hat{x}_k$ SD-Iterierte von x_k.
Z.z.:

$$||x_{k+1} - x_*||_A \leq ||\hat{x}_k - x_*||_A$$

Beweis: Nach (8.17a) gilt für die CG-Iterierten:

$$x_{k+1} = \underset{x \in x_0 + \mathcal{K}_{k+1}}{\arg\min} ||x - x_*||_A$$

mit $\mathcal{K}_{k+1} = \text{span}\{r_0, Ar_0, A^2r_0, .., A^kr_0\} = \text{span}\{d_0, d_1, .., d_k\} = \text{span}\{r_0, r_1, .., r_k\}$ und außerdem $\hat{x}_k = x_k + \alpha r_k \in x_0 + \mathcal{K}_{k+1}$, da $x_k \in x_0 + \mathcal{K}_k$. Damit folgt sofort die Behauptung.

3. Beispiel (D. Herold)

Z.z.: Für $r_0 \in \mathbb{R}^n$ und $p \in \mathcal{P}_{m-1}$ gilt

$$p(A)r_0 = V_m p(H_m) V_m^T r_0 = ||r_0||_2 V_m p(H_m) e_1$$

Beweis: Induktion nach Polynombasis für $\mathcal{P}_{m-1}$
Induktionsanfang $p(x) = x^0$: Mit dem Orthogonalprojektor P_m auf $\mathcal{K}_m$ gilt für $r_0 \in \mathcal{K}_m$

$$\mathcal{I} r_0 = r_0 = P_m r_0 \overset{\text{Lemma 9.1}}{=} V_m V_m^T r_0 = V_m \mathcal{I} V_m^T r_0$$

sowie

$$\mathcal{I} r_0 = r_0 = ||r_0||_2 \frac{r_0}{||r_0||_2} = ||r_0||_2 v_1 = ||r_0||_2 V_m e_1 = ||r_0||_2 V_m \mathcal{I} e_1$$

Induktionsvorraussetzung: $A^n r_0 = V_m H_m^n V_m^T r_0 = ||r_0||_2 V_m H_m^n e_1$

Induktionsschritt $n \to n+1$: Mit $H_m = V_m^T A V_m$ (Theorem 9.1) sowie $V_m V_m^T = P_m$ (Lemma 9.1) gilt

$$\begin{aligned} V_m H_m^{n+1} V_m^T r_0 &= V_m H_m H_m^n V_m^T r_0 \\ &= V_m (V_m^T A V_m) H_m^n V_m^T r_0 \\ &\overset{\text{I.V.}}{=} P_m A A^n r_0 \\ &= A^{n+1} r_0 \end{aligned}$$

da $A^{n+1} r_0 \in \mathcal{K}_m$ für $n+1 \leq m-1$; sowie

$$\begin{aligned} ||r_0||_2 V_m H_m^{n+1} e_1 &= ||r_0||_2 V_m (V_m^T A V_m) H_m^n e_1 \\ &= ||r_0||_2 P_m A V_m H_m^n e_1 \\ &\overset{\text{I.V.}}{=} P_m A A^n r_0 \\ &= A^{n+1} r_0 \end{aligned}$$

womit die Behauptung für alle $p \in \mathcal{P}_{m-1}$ gezeigt ist.

Iterative Lösung großer Gleichungssysteme,
Übung 3, Aufgabe 4
Lukas Kogler

(a)
Siehe das angehängte MATLAB-file.
(b)
Als erstes muss gezeit werden, dass H_m invertierbar, oder equivalent injektiv ist.
Für SPD-Matrizen A ist dies trivial, denn für $x_m \neq 0$ gilt dann:

$$\langle H_m x_m, x_m \rangle = \left\langle V_m^T A V_m x_m, x_m \right\rangle = \langle A V_m x, V_m x \rangle > 0$$

Für **allgemeine** (aber reguläre) Matrizen muss dies jedoch **nicht** unbedingt gelten.
Wir können davon ausgehen, dass $\langle A v_k, v_{k+1} \rangle \neq 0 \quad \forall k = 1 \dots m-1$, da sonst die Lanczos-Iteration schon in einem Schritt $j < m$ abgebrochen wäre. Wir wissen also, dass $V_m^T A v_k = V_m^T A V_m e_k \neq 0 \quad \forall k = 1 \dots m-1$. Für die Injektivität von H_m bliebe also noch $H_m e_m = V_m^T A v_m \neq 0$ zu zeigen. Falls wir die Lanczos-Iteration noch weiterführen würden, wäre unser nächster Basisvector (bis auf Normierung) $v_{m+1} = (I - V_m V_m^T) A v_m$. $V_m^T A v_m = 0$ ist also dazu äquivalent, dass

$$A v_m = (I - V_m V_m^T) A v_m = v_{m+1} \tag{1}$$

Zur Erinnerung: Das Abbruchkriterium des Lanczos-Verfahrens ist

$$h_{m+1,m} = \langle A v_m, v_{m+1} \rangle = 0 \tag{2}$$

Wir konstruieren nun ein reguläres A, sodass 1, aber nicht 2 erfüllt ist (dh dass H_m nicht injektiv ist und das Lanczos-Verfahren nicht abbricht):

$$A := \begin{pmatrix} 0 & \dots & \dots & 0 & 1 \\ 1 & 0 & & & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}$$

Es gilt also $A e_k = e_{k+1} \quad k < b$ und $A e_n = e_1$. Wir zeigen per Induktion, dass $V_m = (e_1, e_2 \dots e_m)$ und dass das Abbruchkriterium nie erfüllt ist.
IA:
Wir starten das Verfahren mit $v_1 := e_1$. Dann ist $v_2 = A v_1 = A e_1 = e_2$ und

das Abbruchkriterium ist nicht erfüllt: $\langle Av_1, v_2\rangle = \langle e_2, e_2\rangle = 1 \neq 0$.
Sei also $V_m = (e_1, e_2 \dots e_m)$. Es ist $v_{m+1} = (I - V_m V_m^T)Av_m = (I - V_m V_m^T)e_{m+1} = e_{m+1}$, was die Identität für V_{m+1} zeigt. Das Abbruchkriterium ist wiederrum nicht erfüllt: $h_{m+1,m} = \langle Av_m, v_{m+1}\rangle = \langle e_{m+1}, e_{m+1}\rangle = 1 \neq 0$. Nun gilt (für $m < n$):

$$H_m = V_m^T A V_m = \begin{pmatrix} 0 & \dots & \dots & 0 \\ 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{pmatrix}$$

$H_m e_m = 0$, H_m ist also nicht injektiv, daher nicht invertierbar!!
Als nächstes soll eine explizite Darstellung für das Residuum $r_m = b - Ax_m$

gefunden werden, wobei $x_m = x_0 + V_m H_m^{-1} e_1$ ist, und gezeigt werden dass es orthogonal auf $\mathcal{K}_m(A, r_0)$ steht. Dafür machen wir uns die Identität $AV_m = V_m H_m + w_m e_m^T$, mit $w_m = h_{m+1,m} v_m$, zu Nutze.

$$\begin{aligned} r_m &= b - Ax_m = b - Ax_0 - AV_m H_m^{-1} e_1 = \\ &= r_0 - (V_m H_m + w_m e_m^T) H_m^{-1} e_1 = \\ &= r_0 - \underbrace{V_m e_1}_{=r_0} - h_{m+1,m} \underbrace{e_m^T H_m^{-1} e_1}_{\in \mathbb{R}} v_{m+1} = \\ &= -(h_{m+1,m} e_m^T H_m^{-1} e_1) v_{m+1} \quad \in \operatorname{span}\{v_{m+1}\} \subseteq \mathcal{K}_m(A, r_0)^\perp \end{aligned}$$

(c)
Falls A SPD ist, (die Lanczos-Iteration daher in eine 3-Term-Rekursion zerfällt) hat man die Option, die (voll besetzten) Matrizen V_m nicht zu speichern. Falls man die Approximation x_m wie in (b) berechnen möchte, bazahlt man aber unter Umständen, in Abhängigkeit von der Besetzungsstruktur von A, mit erhöhtem Rechenaufwand.
Für SPD A kann man also H_m ohne zusätzlichen Aufwand berechnen ohne V_m komplett zu speichern - man braucht in der Iteration nur die jeweils neuesten 2 Basisvektoren. $s_m := H_m^{-1} e_1$ lässt sich dann natürlich auch einfach berechnen. Um $x_m = r_0 + V_m s_m$ zu erhalten, können wir einfach die gesamte Lanczos-Iteration noch einmal durchführen und jedesmal, wenn wir einen neuen Basisvector v_k ausgerechnet haben, x_m um $x_{m,k} \cdot v_k$ updaten. Ein MATLAB-File dazu befindet sich ebenfalls im Anhang.

Iteratives Lösen linearer Gleichungssysteme - Übung 3

Tobias Danczul

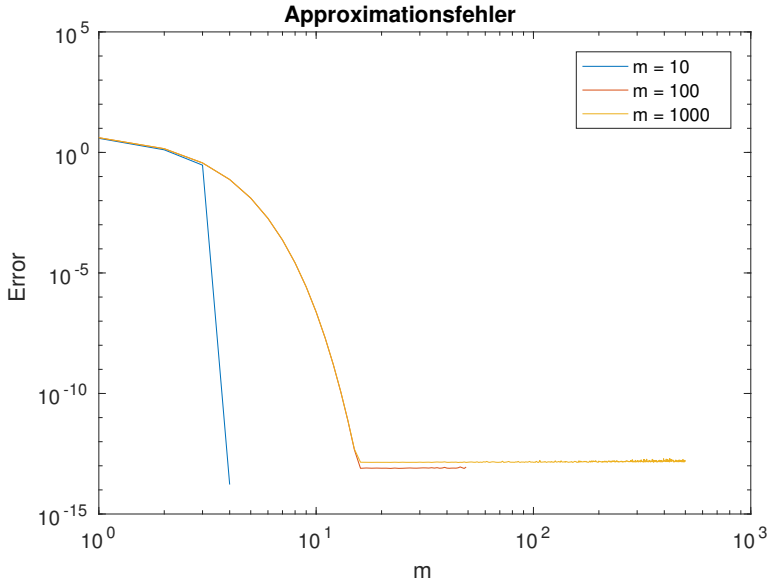
3. Mai 2017

5. Beispiel

Im Folgenden betrachten wir die *Krylov-Approximation* für die Matrixexponentialfunktion. Für große Matrizen $A \in \mathbb{R}^{n\times n}$ kann die direkte Berechnung von $\exp(A)y_0$ empfindlich teuer werden. Deshalb bedienen wir uns der aus der Arnoldi-Iteration hervorgehenden Hessenbergmatrix H_m als Annäherung für A und motivieren daraus

$$\exp(A)y_0 \approx y_m := \|y_0\|_2 V_m \exp(H_m)e_1$$

wobei die kleinere Matrix $\exp(H_m)$ direkt berechnet wird.
In folgendem numerischen Beispiel wählen wir jenes symmetrische und positiv definite A, das aus der FD-Approximation hervorgeht. Die Ergebnisse der Approximation werden mittels direkter Spektralzerlegung und anschließender (inversen) Sinustransformation verglichen. Wir erhalten für wachsende Dimension von m folgenden Plot.



Während sich der Fall $m = 10$ deutlich von den beiden anderen Fällen abhabt, indem der Approximationsfehler bereits in weniger als 10 Schritten Maschinengenauigkeit erreicht, lässt sich zwischen $m = 100$ und $m = 1000$ kaum noch ein Unterschied erkennen.

```
l = (10.^3)/2;
err = zeros(l,3);

for k = 1:3
    %Dmiension A
    n = 10.^k;

    r0 = ones(n,1);
    %Poisson Matrix
    e = ones(n,1);
    A = spdiags([-e 2*e -e], -1:1, n, n);

    % exakt:
    h = 1/(1+n);
    xi = (h:h:1-h)';
    D = spdiags((4*sin(pi*xi/2).^2),0,n,n);
    x = dst(r0);
    x = expm(D)*x;
    x = idst(x);


    for m = 2:n/2

        [T,step] = lanczos_H(r0,A,m);

        e1 = zeros(m,1);
        e1(1) = 1;

        H = T+tril(T,-1)';
        exp_H_e1 = expm(H)*e1;

        d = zeros(n,1);

        v0 = zeros(n,1);
        v1 = r0/norm(r0);

        %Multiplikation mit V_m
        for j = 1:m
            d = d + v1*exp_H_e1(j);
            if j == m
                break;
            end
            v2 = lanczos_V(v0, v1,A, T, j);
            v0 = v1;
            v1 = v2;
        end

        d = norm(r0)*d;

        err(m-1,k) = norm(d - x);
    end

end

loglog(err(:,1));
hold on
loglog(err(:,2));
loglog(err(:,3));
title('Approximationsfehler')
xlabel('m')
ylabel('Error')
legend('m = 10','m = 100','m = 1000')
hold off
```