Modellering en Simulatie

Oefenzitting 3: Eigenwaarden en iteratieve methodes

Academiejaar 2015–2016

1 Theorie

De eenvoudigste iteratieve methode voor het oplossen van een stelsel lineaire vergelijkingen $A\mathbf{x} = \mathbf{b}$ bestaat eruit om de methode van de steilste helling toe te passen op de gebruikelijke doelwaardefunctie

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}.$$

Voorwaar $A \in \mathbb{R}^{n\times n}$ een symmetrische positief definiete matrix is, i.e., $A = U\Lambda^2 U^T$ waarbij $U \in \mathbb{R}^{n\times n}$ een orthogonale matrix is en $\Lambda \in \mathbb{R}$ een diagonaalmatrix van rang n, dan heeft deze kwadratische vectorfunctie een unieke oplossing, namelijk de vector $\mathbf{x}^*$ waarvoor de gradiënt van deze functie $\phi(\mathbf{x})$ gelijk is aan nul.

Opgave 1. Schrijf $\phi(\mathbf{x})$ expliciet neer voor $n = 3$. Bepaal vervolgens de gradiënt

$$\nabla\phi(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}\phi(\mathbf{x}) \\ \frac{\partial}{\partial x_2}\phi(\mathbf{x}) \\ \frac{\partial}{\partial x_3}\phi(\mathbf{x}) \end{bmatrix}.$$

Schrijf het resultaat in matrix-vorm.

Vervolgens dienen we de optimale staplengte nog te bepalen. Dit is de waarde

$$\alpha = \arg\min_{\beta\in\mathbb{R}} \phi(\mathbf{x} - \beta\mathbf{r}).$$

Opgave 2. Bepaal de optimale staplengte α analytisch.

Opgave 3. Toon aan dat de methode van de steilste helling voor deze keuze van de staplengte convergeert.

2 Eigenwaarden

We berekenen de eigenwaarden van een matrix door middel van het QR-algoritme uit de cursus.

Opgave 4. Construeer twee testmatrices:

- $A_k = P_1 L_k P_1^T$ met P_1 een willekeurige orthogonale matrix en $L_k = \mathrm{diag}(1, 2, 3, \ldots, k)$ en
- $B_k = P_2 L_k P_2^{-1}$ met P_2 een matrix bestaande uit standaard normaalverdeelde willekeurige waarden met $L_k = \mathrm{diag}(1, 2, 3, \ldots, k)$.

Hint: zoek op waarvoor de functies `orth` en `randn` dienen.

*Begeleider: Nick Vannieuwenhoven (`nick.vannieuwenhoven@cs.kuleuven.be`).
†Opgesteld door: L. Vanherpe (2005–2010), N. Achtsis (2010–2014) en N. Vannieuwenhoven (2014–2016)

Opgave 5. Implementeer het QR-algoritme. Je dient het iteratieve proces te stoppen wanneer de Frobeniusnorm van de strikte benedendriehoeksmatrix kleiner is dan de invoerparameter `tol`; we noemen deze norm het *residu*. Als uitvoer geeft het algoritme de eigenwaarden van de invoermatrix, alsook het residu in elke iteratiestap. Hint: de `tril` functie kan handig zijn.

De convergentie-orde van een iteratieve methode wordt in de huidige context gedefiniëerd als de waarde p waarvoor

$$\lim_{k\to\infty} \frac{|r_{k+1}|}{|r_k|^p} \to \beta$$

met $\beta \neq 0$ een constante en r_j het residu in iteratie j.

Opgave 6. Wat is de convergentiesnelheid van dit QR-algoritme voor de matrix A_{20}? Voor B_{20}? Gebruik als tolerantie `tol = eps`.

Opgave 7. Na hoeveel iteraties werd het algoritme beëindigd omdat het residu voldoende klein was voor de matrix A_{20}? Hoeveel extra iteraties zou je dan ongeveer nodig hebben om het residu te verkleinen tot `tol = eps^20`?

De convergentiesnelheid van het QR-algoritme kan dramatisch versneld worden door gebruik te maken van *shifts*. Hiervoor is het wel cruciaal dat de matrix eerst wordt teruggebracht tot een Hessenbergmatrix. In Matlab kan dit eenvoudig met de functie `hess`. Het QR-algoritme met een shift toegepast op het laatste diagonaalelement van een Hessenbergmatrix $A \in \mathbb{R}^{n\times n}$ kunnen we dan als volgt beschrijven. Bereken iteratief

$$\begin{aligned} \kappa^{(k)} &= a^{(k)}_{n,n} \\ A^{(k)} - \kappa^{(k)} I_n &= Q^{(k)} R^{(k)} \\ A^{(k+1)} &= R^{(k)} Q^{(k)} + \kappa^{(k)} I_n \end{aligned}$$

zolang $|a^{(k)}_{n,n-1}| > \tau$ waarbij τ de gekozen tolerantie voor het residu is. Vervolgens stellen we $a^{(k)}_{n,n-1}$ gelijk aan nul. Nu zal $a_{n,n}$ een van de eigenwaarden van A zijn.

Opgave 8. Implementeer het QR-algoritme met een shift toegepast op het laatste diagonaalelement. Gebruik `hess` om de invoermatrix naar Hessenbergvorm om te zetten.

Opgave 9. Wat is de convergentiesnelheid van bovenstaand QR-algoritme met een shift toegepast op het laatste diagonaalelement voor A_{100}, i.e., hoe snel convergeert $|a^{(k)}_{n,n-1}|$ naar nul voor A_{100}? Voor B_{100}?

Om alle eigenwaarden van een Hessenbergmatrix $A \in \mathbb{R}^{n\times n}$ te berekenen, gaat men als volgt te werk. Eerst passen we bovenstaand algoritme met shift op het laatste diagonaalelement toe op A. Wanneer $|a^{(k)}_{n,n-1}|$ voldoende klein is dan stellen we het gelijk aan nul. Vervolgens kiezen we als shift $a^{(k)}_{n-1,n-1}$ en passen we dezelfde strategie toe om $|a^{(k)}_{n-1,n-2}|$ naar nul te laten streven. Wanneer deze waarde voldoende klein is dan stellen we ze gelijk aan nul. Deze strategie blijven we herhalen tot $a^{(k)}_{2,2}$ als shift gebruikt wordt en $|a^{(k)}_{2,1}|$ door de QR-iteraties voldoende klein gemaakt wordt.

Opgave 10. Implementeer een QR-algoritme dat shifts gebruikt om de convergentie van het standaard QR-algoritme te verbeteren. Gebruik $\tau = 10^{-64}$.

Opgave 11. Wat gebeurt er met de convergentie van het QR-algoritme met shifts wanneer we de elementen op de eerste onderdiagonaal, i.e., de elementen $a_{2,1}, a_{3,2}, \ldots, a_{n,n-1}$, niet expliciet op nul zetten wanneer een van de eigenwaarden geconvergeerd is? Helpt het verlagen van de tolerantie τ? Experimenteer met A_5.

Opgave 12. Pas het QR-algoritme met shifts en zonder shifts toe op de matrices A_{100} en B_{100}. Gebruik `tol=1e-5` voor het QR-algoritme zonder shifts. Hoeveel iteraties heeft het QR-algoritme zonder shifts nodig om te convergeren? Hoeveel iteraties heeft het verbeterde QR-algoritme nodig voor convergentie?

Opgave 13. Hoe groot is het verschil tussen de eigenwaarden die werden berekend met het QR-algoritme met en zonder shifts en de eigenwaarden die de `eig` functie berekent?

3 Stelsels vergelijkingen

Opgave 14. Implementeer de methode van de steilste helling. Zorg ervoor dat een van de uitvoerargumenten een vector is dewelke als k^{de} waarde het residu $\|A\mathbf{x}^{(k)} - \mathbf{b}\|_2$ bevat van iteratie k; hierin is $\mathbf{x}^{(k)}$ de benaderende oplossing in iteratiestap k.

We zullen de methode van de steilste helling en de methode van de toegevoegde gradiënten toepassen op de willekeurige symmetrisch positief definiete matrices van dimensie $n \times n$ met conditiegetal κ. Hiervoor kan je de volgende functie gebruiken:

```
function [A] = randSPD(n,kappa)
  U = orth(randn(n));
  A = U * diag(round(exp(linspace(0,log(kappa),n)))/kappa) * U';
end
```

De methode van de toegevoegde gradiënten kan in Matlab worden uitgevoerd door de `pcg` functie op te roepen.

Opgave 15. Vergelijk de convergentie van de methode van de toegevoegde gradiënten en de methode van de stijlste helling voor een matrix gegenereerd met `randSPD(500,10)`. Dit wil zeggen: plot het verloop van de residu's in functie van het aantal iteraties. Hint: gebruik `semilogy` voor betekenisvolle grafieken.

Opgave 16. Wat zijn de convergentie-ordes voor beide methoden uit de voorgaande opgave?

Opgave 17. Vergelijk het aantal iteraties van de methode van de toegevoegde gradiënten en de methode van de steilste helling in functie van het conditiegetal κ voor willekeurige 750×750 matrices gegenereerd met `randSPD`. Probeer de waarden $\kappa = 2^k$ met $k = 5, 6, \cdots, 20$.

Iteratieve methoden voor het oplossen van stelsels vergelijkingen zijn bijna uitsluitend gebaseerd op matrix-vector producten en vectoroperaties. De invoermatrix A wordt, in tegenstelling tot een directe methode zoals een LU-ontbinding of Gausseliminatie, nooit aangepast. Dit maakt dat iteratieve methodes voornamelijk gebruikt worden in gevallen waarbij een LU-ontbinding niet haalbaar is vanwege de grootte van de matrix of wanneer de matrix niet in expliciete vorm beschikbaar is.

In het volgende voorbeeld zullen we werken met een ijle matrix: dit is een matrix waarbij de meeste elementen gelijk zijn aan nul. De niet-nulwaarden van de matrix kunnen dan efficiënt voorgesteld worden door $a_{i,j}$ voor te stellen door de rij- en kolomindex en de waarde zelf: (i, j, a_{i_j}). Op deze manier is de benodigde opslagcapaciteit voor een ijle matrix gelijk aan drie keer het aantal niet-nulwaarden. Een voorbeeld van zo'n matrix kan in Matlab ingeladen worden met het volgende commando:

```
A = gallery('wathen',250,250,1);
A = A'*A;
```

Het laatste commando zorgt ervoor dat A een symmetrisch positief definiete matrix is, zodat we de methode van de toegevoegde gradiënten kunnen toepassen. Vervolgens genereren we een willekeurig rechterlid als volgt:

```
x = randn(size(A,1),1);
b = A*x;
```

Dit zorgt ervoor dat $A\mathbf{x} = \mathbf{b}$.

Opgave 18. Laad de matrix A in en genereer de vector $\mathbf{b}$. Wat is diens dimensie? Probeer de ijle matrix A eens voor te stellen met een volle matrix door `full(A)` uit te voeren. Hoeveel GB aan werkgeheugen zou je nodig hebben om deze matrix voor te stellen als een eenvoudige volle matrix?

Opgave 19. Gebruik de methode van de toegevoegde gradiënten om het stelsel $A\mathbf{x} = \mathbf{b}$ op te lossen. Hoeveel iteraties heb je nodig? Plot het verloop van de residu's. Hoeveel bedraagt de convergentie-orde?