Oefeningen Numerieke Wiskunde

Oefenzitting 6 (PC): Het oplossen van stelsels lineaire vergelijkingen

In deze oefenzittingen ga je m.b.v. Matlab aspecten zoals conditie, stabiliteit en complexiteit onderzoeken voor stelsels vergelijkingen. Het relevante deel van de cursus is *Hoofdstuk 3 'Stelsels lineaire vergelijkingen'*, de slides over QR, SVD en kleinste kwadraten benaderingen en de slides over het creëren van nullen in een matrix. Voor het uitwerken van de opgaven gebruik je de `.m`-bestanden die je kan vinden op Toledo.

1 MATLAB–functies voor deze oefenzitting

Genereren van matrices

De volgende functies genereren een stelsel $Ax = b$, waarvan de oplossingsvector x bestaat uit natuurlijke getallen. Ieder commando genereert een matrix $M = [A\ b]$ met A een $n \times n$–matrix en b een $n \times 1$–vector.

- `M = genmatrix1(n)` : De matrix A is een goed geconditioneerde random matrix.
- `M = genmatrix2(n)` : De matrix A is een goed geconditioneerde random matrix met een specifiek permutatiegedrag.
- `M = genmatrixc(n)` : De matrix A is een slecht geconditioneerde random matrix.

Stelsels oplossen

- `G = gauss1(M)` waarbij $M = [A\ b]$ met A een $n \times n$–matrix en b een $n \times 1$–vector. Deze functie maakt de matrix

$$\left(\begin{array}{c|c|c} & & 1 \\ A & b & \vdots \\ & & n \end{array}\right)$$

  aan en past Gauss–eliminatie toe (Cursusboek, Algoritme 3.4).

- `G = gauss2(M)` is analoog aan `gauss1`, maar met optimale rij–pivotering (Cursusboek, Algoritme 3.5). Merk op dat dit equivalent is met `A\b` in Matlab voor vierkante matrices A.

- `[Q,R] = qr(M)` berekent een QR-factorisatie van de matrix M.
- `x = asubst(G)` voert achterwaartse substitutie uit op het resultaat van `gauss1` of `gauss2` (`x = asubst(R)` resp. op het resultaat van `qr`), en geeft de oplossing x van het stelsel $Ax = b$.

2 Conditie en achterwaartse stabiliteit

Stel dat je een algoritme $\hat{F}(x)$ hebt om $F(x)$ te berekenen. De **voorwaartse absolute fout** is gedefinieerd als $\Delta F(x) = \hat{F}(x) - F(x)$, dit is dus de fout op het eindresultaat omwille van afrondingsfouten. De **achterwaartse absolute fout** Δx is gedefinieerd als $\hat{F}(x) = F(x + \Delta x)$, dit is dus de fout op het gegeven waarvoor een exact algoritme $F(x)$ hetzelfde resultaat geeft als het algoritme $\hat{F}(x)$ op het juiste gegeven. Het verband tussen de corresponderende relatieve fouten wordt gegeven door het (relatief) conditiegetal:

$$\frac{||\hat{F}(x) - F(x)||}{||F(x)||} = \frac{||F(x + \Delta x) - F(x)||}{||F(x)||} \leq \kappa_F \frac{||\Delta x||}{||x||}.$$

Een algoritme $\hat{F}(x)$ is **achterwaarts stabiel** als de achterwaartse relatieve fout klein is, d.w.z., van de grootte-orde van de machine-precisie: voor een niet al te grote constante C:

$$\frac{||\Delta x||}{||x||} \leq C \cdot \epsilon_{mach} \quad \text{en bijgevolg} \quad \frac{||\hat{F}(x) - F(x)||}{||F(x)||} \leq C \cdot \kappa_F \cdot \epsilon_{mach}.$$

Veronderstel nu dat je een algoritme $\hat{x} = \hat{x}(A, b)$ hebt om het stelsel $Ax = b$ op te lossen. De voorwaartse absolute fout is gelijk aan $\hat{x} - x$. De achterwaartse absolute fout is gelijk aan het **residu**

$$r = A\hat{x} - b,$$

want $A\hat{x} = b + r$, dus r is de fout op het gegeven b waarvoor $\hat{x}$ de exacte oplossing is van het gewijzigde stelsel.[1] Uit Sectie 9.2 van het handboek volgt dat

$$\frac{||\hat{x} - x||}{||x||} \leq \kappa_A \frac{||r||}{||b||}, \tag{1}$$

met κ_A het conditiegetal van de matrix A. Zie ook Sectie 10.4.

Een algoritme om stelsels op te lossen is achterwaarts stabiel als het residu klein is t.o.v. het rechterlid b: voor een niet al te grote constante C:

$$\frac{||r||}{||b||} \leq C \cdot \epsilon_{mach} \quad \text{en bijgevolg} \quad \frac{||\hat{x} - x||}{||x||} \leq C \cdot \kappa_A \cdot \epsilon_{mach}. \tag{2}$$

[1] Merk op dat we geen fout op de matrix A moeten aanbrengen opdat $\hat{x}$ een oplossing zou zijn van het gewijzigde stelsel.

3 Oefeningen

Probleem 1. (De LU ontbinding)

(a) Definieer de matrix

$$A = \begin{pmatrix} 7 & 8 & 0 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

met

```
>> A = [ 7 8 0 ; 1 2 3 ; 4 5 6 ]
```

Bereken de LU-ontbinding van A met `[L,U] = lu(A)`. Wat is $L*U$? Geef de Matlab-bevelen om uit L en U de determinant van A te berekenen. (Hint: gebruik de functies `prod` en `diag` en controleer met de functie `det`.)

(b) Definieer

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 4 & 3 & 5 \\ 9 & 8 & 0 \end{pmatrix}.$$

en bereken opnieuw de LU-ontbinding van A. Wat merk je op als je de matrices L en U bekijkt? Geef een verklaring voor wat er gebeurd is. (Hint: `doc lu`.)

Probleem 2. (Gauss-eliminatie) In deze oefening bestuderen we de stabiliteit en de conditie bij het oplossen van stelsels. We observeren het belang van optimale rijpivotering bij Gauss eliminatie en het effect van de conditie op het eindresultaat.

Open een nieuw script en sla het op als `probleem2.m`. Gebruik de functie `genmatrix1` om een 6×7–matrix $M = [A\ b]$ te genereren. Los vervolgens het stelsel $Ax = b$ op m.b.v. de functies `gauss1`, `gauss2` en `qr`, gevolgd door de functie `asubst`. Je bekomt dus drie oplossingen voor hetzelfde stelsel in de respectievelijke variabelen `x1`, `x2` en `x3`.[2] Je kan het verschil zien tussen deze oplossingen met `format long`.

(a) Bereken de relatieve fout van de oplossingen en de residu's, indien je weet dat de exacte oplossing een vector met natuurlijke getallen is. Vul in de onderstaande tabel de grootte-orde van de fouten en de residu's in. De relatieve fout kan je berekenen met

```
fout1 = norm(x1-x)/norm(x)
```

[2] De namen van de variabelen zijn bij wijze van voorbeeld, om verder in de opgave naar te kunnen verwijzen.

waarbij `x` de exacte oplossing is, die je kan vinden uit een van de berekende oplossingen m.b.v. de functie `round`. De norm van het residu kan je berekenen met

```
residu1 = norm(A*x1-b)
```

waarbij je A en b nog uit M moet halen met `A = M(:,1:6)` en `b = M(:,7)`. Let erop dat je dit nog deelt door $||b||_2$ voor in de tabel. Doe eerst

```
format short e
```

zodat je de grootte-orde goed kan zien (bvb. $10^{-5}, 10^{-9}, \ldots$).

(b) Wat kan je zeggen over de stabiliteit van de methodes? (Zie vergelijking (2).)

Doe nu hetzelfde maar met de functie `genmatrix2`. Kopieer hiervoor de code die je al hebt en pas ze aan. (De bedoeling is dat als je het script opnieuw uitvoert je de resultaten krijgt voor alle verschillende soorten matrices.)

(c) Wat is het verschil t.o.v. je resultaten met het eerste stelsel? Verklaar.

(d) Wat kan je nu zeggen over de stabiliteit van de methodes?

Merk op dat als je het script meermaals uitvoert, je resultaten telkens anders zullen zijn, want we werken met random matrices. De grootte-ordes zullen echter redelijk gelijk blijven.

Doe nogmaals hetzelfde, maar nu voor de functie `genmatrixc`. Vul de tabel verder aan.

(e) Bereken het 2-norm conditiegetal van de matrix A voor de drie verschillende stelsels m.b.v. de functie `cond`. Vul telkens de grootte-orde in de tabel in.

(f) Ga na dat de relatieve fouten van de oplossingen en de residu's voor alle verschillende matrices voldoen aan formule (1).

(g) Wat is je algemene besluit over de stabiliteit van de methodes?

	$\kappa_2(A)$		gauss1 gauss2 qr
genmatrix1		$\|\hat{x} - x\|_2 \,/\, \|x\|_2$ $\|\|r\|\|_2/\|\|b\|\|_2$	
genmatrix2		$\|\hat{x} - x\|_2 \,/\, \|x\|_2$ $\|\|r\|\|_2/\|\|b\|\|_2$	
genmatrixc		$\|\hat{x} - x\|_2 \,/\, \|x\|_2$ $\|\|r\|\|_2/\|\|b\|\|_2$	

Probleem 3. (Implementatie-opdracht)

Vervolledig de functie

```
[Q,R]=qrtridiag(M)
```

die een QR-factorisatie berekent van een tridiagonale matrix $M = [A\ B]$:

$$M = \left[\begin{array}{ccccc|ccc} a_1 & e_1 & & & 0 & b_{1,1} & \cdots & b_{1,p} \\ f_1 & a_2 & e_2 & & & b_{2,2} & & b_{2,p} \\ & \ddots & \ddots & \ddots & & \vdots & & \vdots \\ & & f_{n-2} & a_{n-1} & e_{n-1} & b_{n-1,1} & \cdots & b_{n-1,m} \\ 0 & & & f_{n-1} & a_n & b_{n,1} & \cdots & b_{n,p} \end{array}\right].$$

De implementatie moet gebeuren met maximaal $n-1$ Givens rotaties. Een Givens rotatie kan berekend worden met de standaard Matlab functie `planerot` (zie `help planerot`).

Test je algoritme: is Q orthogonaal, R bovendriehoeks en $M = Q*R$? Vergelijk met het resultaat van de matlab functie `qr`. Merk op dat dit niet noodzakelijk hetzelfde is, waaruit nogmaals blijkt dat de QR-factorisatie essentieel uniek is. D.w.z. dat de kolommen van Q uniek zijn op het teken na. Wat betekent dit voor de rijen van R?

Hints:

- Een matrix M met de bovenstaande structuur kan je bv. genereren met

```
M = [diag(rand(n,1)) + diag(rand(n-1,1),-1)
        + diag(rand(n-1,1),1), rand(n,p)];
```

  waarbij je alles op één lijn zet.