NMB oefenzitting 1

Nico Vervliet, Michiel Vandecappelle, *KU Leuven* *20 februari 2018*

1 Pen-en-papieroefeningen

1. Gegeven een matrix $A \in \mathbb{R}^{10\times 7}$ met rang 5. Stel de singuliere waardenontbinding $A = USV^*$ visueel voor en duid de kolomruimte (bereik), de rijruimte, de nulruimte en nulruimte van A^* aan.

2. Gegeven de matrices

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 3 & 4 & 1 \\ 2 & 3 & 4 & 5 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & & & \\ & 1 & & 2 \\ & & 1 & \\ & -2 & & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 0 & 1 & 3 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix}.$$

   Bereken AB en CA (zonder computer).

2 Matlaboefeningen

3. Creëer twee willekeurige matrices `A` en `B` met grootte 1000×2000 met normaal verdeelde elementen (hint: `randn`). Het doel is om beide matrices elementsgewijs met elkaar te vermenigvuldigen, i.e., `C(i,j) = A(i,j)*B(i,j)`. Implementeer de bewerkingen op de volgende manieren:

   a) Met een geneste `for`-lus waarvan de buitenste over de kolommen loopt, en de binnenste over de rijen.

   b) Met een geneste `for`-lus waarvan de buitenste over de rijen loopt, en de binnenste over de kolommen.

   c) Prealloceer `C` in de vorige twee methoden door eerst een nulmatrix van de juiste grootte aan te maken.

d) Zonder `for`-lussen, met `.*`.

Maak voor de verschillende gevallen een functie die `A` en `B` als invoer gebruikt en `C` als uitvoer geeft. Schrijf een script dat de verschillende functies aanroept. Meet de tijd nodig voor de verschillende gevallen. Gebruik hiervoor de `timeit` functie: `timeit(@() jouwfunctie(A,B))`, met `jouwfunctie` de functie waarvan je de tijd wil meten, of gebruik `tic; jouwfunctie(A,B); toc`. Wat besluit je?

4. Creëer een willekeurige matrix `A` met grootte 1000×2000 en een willekeurige vector `x` met grootte 1000×1. Gebruik de uniforme verdeling om de willekeurige data te genereren (hint: `rand`) Vermenigvuldig elke kolom van `A` elementsgewijs met `x` en bereken vervolgens de som van elke kolom:

   a) Met een `for`-lus om het elementsgewijze product te berekenen en `sum` om de som te berekenen (hint `.*`).

   b) Zonder `for`-lus met `bsxfun` (hint: `@times = .*`) en met `sum`.

   c) Met een matrix-vectorvermenigvuldiging.

   Meet de tijd opnieuw voor de verschillende functies. Wat kan je besluiten?

5. Creëer een willekeurige matrix `A` met grootte 2000×1000 en met gehele getallen tussen -5 en 5 (hint: `randi`). Bereken de matrix `B` waarvan kolom `j` het product is van kolommen `1` t.e.m. `j` van `A`

   a) Met twee `for`-lussen.

   b) Met een `for`-lus, preallocatie van `B` en hergebruiken van de resultaten van de vorige kolom.

   c) Met een geschikte ingebouwde Matlabfunctie.

6. Implementeer het klassieke Gram–Schmidt algoritme:

   for $j = 1, \ldots, n$ **do**
   $v_j = a_j$
   for $i = 1, \ldots, j-1$ **do**
   $r_{ij} = q_i^* a_j$
   $v_j = v_j - r_{ij} q_i$
   $r_{jj} = \|v_j\|_2$
   $q_j = v_j / r_{jj}$

   Maak twee versies: een met twee `for`-lussen en een met slechts een `for`-lus (hints: `'`, `norm`). Test je algoritme voor een matrix `A = rand(10,5)`. Vergelijk je resultaat met de Matlabimplementatie van `qr`. Krijg je een gelijkaardig resultaat? Hoe kan je met `qr` een gereduceerde QR ontbinding krijgen (hint: `help qr`)?