(Eigen)aardigheden van MATLAB

SCIENTIFIC COMPUTING, 10 SEPTEMBER 2012

1 Introductie

MATLAB is een programmeertaal (programmeeromgeving) voor het *numeriek* oplossen van wiskundige problemen (uit de techniek en de exacte wetenschap) met een bijzonder krachtige collectie standaard commando's[1] en subroutines. MATLAB is gebruikersvriendelijk en daardoor uitermate geschikt om 'snel' een programmaatje in te schrijven of om 'eventjes' een idee voor een algorithme uit te testen. Hoewel MATLAB gebruikersvriendelijk is, levert het betrouwbare resultaten: de 'kern' van MATLAB is gebaseerd op routines (uit LAPACK) die geschreven zijn door vooraanstaande computational scientists en bevat de Computational Science 'state of the art'. Als betere algorithmes beschikbaar komen, dan worden die verwerkt in nieuwe updates van MATLAB. Hetzelfde geldt (zij het in wat mindere mate) voor de overige subroutines (functie subroutines) die door MATLAB geleveerd worden.

Gebruikersvriendelijkheid gaat enigsinds ten koste van efficiëntie. Een algorithme geïmplementeerd in C++ of FORTRAN is sneller dan een implementatie in MATLAB. Bovendien kan de factor waarmee MATLAB versie trager is per type commando verschillen. Dat neemt niet weg dat MATLAB behoorlijk efficiënt is. Er zijn diverse onderzoeksinstituten waar veel grootschaligrekenwerk verricht wordt die al hun rekenwerk in MATLAB doen.

In tegenstelling tot programmeertalen als C++ en FORTRAN hoef je in MATLAB niet te vertellen met welke type grootheden je werkt en van je array's hoef je niet te vertellen hoe groot ze zijn: dat zoekt MATLAB zelf uit. Dat is een van de redenen waarom MATLAB zo gebruikersvriendelijk is, maar ook waarom MATLAB minder efficiënt is.

Matlab is gebaseerd op manipulaties met array's van getallen. Een $n \times k$ matrix is een `n` bij `k` array, een n-vector is een `n` bij `1` array. Het product $\mathbf{AB}$ van een $n \times k$ matrix $\mathbf{A}$ en een $k \times \ell$ matrix $\mathbf{B}$ is een $n \times \ell$ matrix en krijg je in MATLAB met het commando `A*B`.[2] `A'` is de matrix $\mathbf{A}^* \equiv \mathbf{A}^{\mathrm{H}} \equiv \bar{\mathbf{A}}^{\mathrm{T}}$, de $k \times n$ complex geconjugeerde getransponeerde van $\mathbf{A}$. Als $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ en $\mathbf{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ kolom vectoren zijn van dimensie n en $(\mathbf{x}, \mathbf{y}) \equiv \sum_{j=1}^{n} x_j \bar{y}_j$ is het (complex) inproduct van $\mathbf{x}$ en $\mathbf{y}$, en $\mathbf{x}$ en $\mathbf{y}$ zijn in MATLAB ingevoerd als `x`, respectievelijk `y` (als `n` bij `1` array) dan kan je in MATLAB het inproduct uitrekenen als `y'*x`. Zie in dat je het inproduct inderdaad kunt krijgen als een product van matrices en dat in het commando `y'*x` de afmetingen kloppen.[3]

Onderstaande opdrachten zijn bedoeld om een beetje te oefenen met MATLAB en om bovenstaande opmerkingen te illustreren.

Als je nog geen ervaring hebt, met MATLAB, lees dan eerst de tutorial. Schroom niet om de `help` faciliteit en eventueel de `type` faciliteit van MATLAB te gebruiken. Met, bijvoorbeeld `help lu` vertelt MATLAB je hoe je de LU-decompositie kunt berekenen en wat dat is, terwijl `type pcg` niet alleen vertelt wat pcg is en doet, maar laat het ook de code zien. Voer de commando's `help lu`, `help pcg` en `type pcg` eens in. Wat levert `type lu`?

Opdracht 1 (P).

Voer de matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix}$ en de vector $\mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ in. Los met MATLAB x op uit de vergelijking $\mathbf{Ax} = \mathbf{b}$. Vervang de $(3,3)$ coëfficiënt van $\mathbf{A}$ door 9 en los $\mathbf{x}$ weer op uit $\mathbf{Ax} = \mathbf{b}$. Kan je de resultaten en de 'melding' van MATLAB begrijpen? Voer het commando `lu(A)` uit en de eerste drie commando's die besproken worden in "`help lu`". Kan je de resultaten begrijpen? Vervang nu $\mathbf{A}$ door de 3×2 matrix die bestaat uit de eerste en tweede kolom van $\mathbf{A}$. Wat verwacht je van het commando `x=A\b`? Komen je verwachtingen uit? Wat verwacht je van het commando `x=A*b`?

Voor de volgende opdrachten kan je de M-files Opdracht2.m, ..., Opdracht4.m en PartSVD.m gebruiken.[4]

Opdracht 2 (P).

In MATLAB hoef je geen geheugen te reserveren. Er is zelfs geen officiële manier om dat te doen.[5] In de M-files Opdracht2.m kan je zien dat het reserveren van

[1] bijvoorbeeld als `A` een $n \times n$ matrix is en `b` is een n-vector dan levert het MATLAB commando `x=A\b` de oplossing van de matrix-vector vergelijking $\mathbf{Ax} = \mathbf{b}$

[2] Hierbij zijn $\mathbf{A}$ en $\mathbf{B}$ in MATLAB ingevoerd is als `A`, respectievelijk `B`

[3] In een taal als C++ is 0 de laagste array-index, in MATLAB is dat 1. Met bijvoorbeeld het commando `x=0:0.1:1`, wordt `x` een 1 by 11 array met `x(1)=0, x(2)=0.1, x(3)=0.2, ...`. Dus `x(i)=(i-1)*0.1` voor `i=1,2,...,11`.

[4] M-files als Opdracht2.m kan je in MATLAB 'runnen' met het commando `Opdracht2` (dus zonder de extensie .m).

[5] Met `x=zero(200,300);` reserveer je een array van 200 bij 300 nullen.

geheugen voor een array een dramatisch verschil maakt in de benodigde tijd. Neem voor `N` achtereenvolgens de waarde `10^4`, `5*10^4`, `10^6`, etc.. Wat is de functie van het commando `clear x` in deze file? Maakt het nog verschil of $\mathbf{y}$ een vector is van integers, van reële getallen (vermenigvuldig $\mathbf{y}$ eens met π, `y=y*pi;`) of van complexe getallen (tel het complexe getal i eens op bij $\mathbf{y}$: `ii=sqrt(-1); y=y+ii;`)?

Opdracht 3 (P).

Gebruik Opdracht3.m om te kijken of de bewering dat MATLAB sneller wordt door vector operaties als $\mathbf{x} = \mathbf{y} + \mathbf{z}$ daadwerkelijk als vectoroperatie uit te voeren, als

```
x=y+z;
```

in plaats van met een loop:

```
for j=1:n, x(j)=y(j)+z(j); end
```

Opdracht 4 (P).

In MATLAB kan je op verschillende manieren een functie definiëren om bv. een array met functiewaarden te vullen. Wil je bijvoorbeeld een plot van $\sin^2(x)$ voor $x \in [0, 2\pi]$, dan kan je een array

```
x=0:0.01:1; x=2*x*pi;
```

van x-waarden aanmaken en het bijhorende array `f` van sinus^2-waarden laten uitrekenen met een loop

```
for j=1:length(x)
     y(j)=sin(x(j)); f(j)=y(j).*y(j);
end
```

of middels array manipulaties als

```
y=sin(x); f=y.*y;
```

of met een *function* subroutine (die array vermenigvuldigingen accepteerd).

Voor functies van twee veranderlijken is het nuttig vaak om eerst een twee dimensionaal array van x-waarden en een twee dimensionaal array van y-waarden aan te maken. Je kunt hiervoor de MATLAB commando's `meshgrid` en `ndgrid` gebruiken.

- Onderzoek de efficiëntie van het uitrekenen van een array van functie-waarden voor een functie van twee veranderlijken (bv. $f(x, y) = \sin(x)\sin(2y)$ voor $(x, y) \in [0, \pi] \times [0, \pi]$) voor verschillende manieren om een functie te definiëren.
- Bespreek de voordelen en nadelen van de verschillende manieren.
- Waarom is het nuttig om een functie als function subroutine (of inline function of string) te definiëren en niet pas "ad hoc" in de commando regel waarin je functiewaarden wilt uitrekenen?

In deze cursus kiezen we ervoor om functies te definiëren middels een function subroutine (waarin array's als input variabelen opgegeven kunnen worden).

Matrices kan je zien als two dimensionale array's van getallen. MATLAB heeft commando's voor het grafisch weergeven van array's van een, twee en drie dimensies. Hierbij kent MATLAB aan iedere getal (in het bereik van het array) een kleurwaarde toe (die bepaald wordt door de maximale en minimale waarde van het te-plotten array en door de gebruikte `colormap`). Een digitaal 'zwart-wit' plaatje kan je zien als de grafische weergave van een tweedimensionaal array: een 'zwart-wit' plaatje bestaat uit pixels met een zekere grijs tint. Grijstinten in een zwart-wit plaatje corresponderen met (integer) getallen tussen 0 (zwart) en 255 (wit).[6] Met andere woorden, een digitale zwart-wit foto kan je zien als een matrix (en een reële matrix kan je, eventueel na schaling, zien als een digitaal plaatje).

Opdracht 5 (P).

In Opdracht5.m wordt een plaatje ingelezen en geïdentificeerd met een matrix. Zie dit in.

Uit een $n \times k$ matrix $\mathbf{A}$ kan je een nk vector maken door de kolommen van de matrix onder elkaar te zetten. Dat kan in MATLAB op verschillende manieren. Onderzoek de efficiëntie ervan.

Opdracht 6 (P).

De SVD (Singuliere Waarde Decompositie, svd in MATLAB) van een $n \times k$ matrix $\mathbf{A}$ bestaat uit een unitaire $n \times n$ matrix $\mathbf{U}$ (d.w.z. $\mathbf{U}^{\mathrm{H}}\mathbf{U} = \mathbf{I}_n$, waarbij $\mathbf{I}_n$ de $n \times n$ eenheids matrix is), een unitaire $k \times k$ matrix $\mathbf{V}$ en een $n \times k$ diagonaal matrix Σ zodat $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{H}}$. De diagonaal elementen Σ_{ii} van Σ zijn ≥ 0 en geordend van groot naar klein.

- Lees een zwart-wit plaatje in als matrix,
- bepaal de SVD,
- plot de diagonaal elementen van Σ (op log10-schaal),
- bereken voor $j = 1 : k$ de matrix $\mathbf{A}_j$ die ontstaat door $\mathbf{U}_j$, Σ_j en $\mathbf{V}_j^{\mathrm{H}}$ te vermenigvuldigen. Hierbij is $\mathbf{U}_j$ de $n \times j$ matrix die bestaat uit de eerste j kolommen van $\mathbf{U}$, $\mathbf{V}_j$ bestaat uit de eerste j kolommen van $\mathbf{V}$ en Σ_j is de linker $j \times j$ bovenblok van Σ.
- Representeer $\mathbf{A}_j$ als een plaatje.

[6] of, afhankelijk van de gehanteerde conventie met reële getallen tussen 0 en 1.