FACULTEIT
BIO-INGENIEURSWETENSCHAPPEN

WETENSCHAPPELIJK PROGRAMMEREN: PRACTICUMNOTA'S 1ste SEMESTER

Prof. dr. ir. Jan Verwaeren
Dr. Demir Ali Köse
Dr. Gauthier Vanhaelewyn

Prof. dr. Bernard De Baets

Eerste Bachelor in de Bio-ingenieurswetenschappen
Academiejaar 2020–2021

UNIVERSITEIT
GENT

DEEL I

PROGRAMMEREN IN MATLAB

1

REKENEN MET MATLAB

1.1 Praktische richtlijnen

Om in te loggen op computers in de PC-zalen van UGent gebruik je je UGentNet gebruikersnaam en paswoord. Na het inloggen wordt een connectie gemaakt met je persoonlijke **H-schijf**. Het is belangrijk al je bestanden op te slaan op deze schijf (en bijvoorbeeld niet onder 'Mijn documenten'). Wanneer je via een andere computer connectie maakt met het UGentNet, heb je op deze manier steeds toegang tot je bestanden.

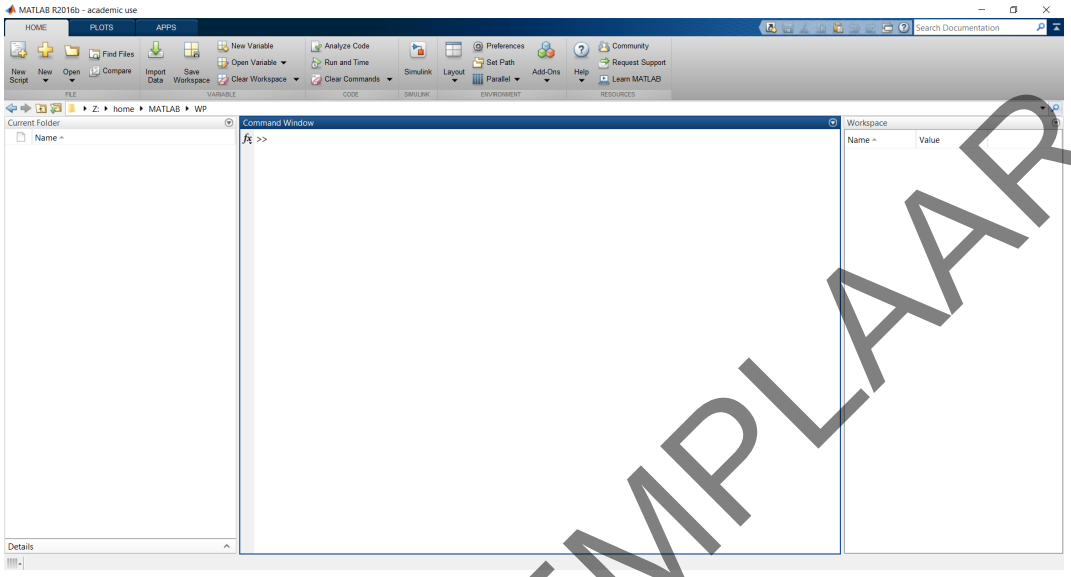
Het is echter ook mogelijk om toegang te hebben tot je H-schijf vanop een computer die zich buiten het UGentNet bevindt (bijvoorbeeld thuis). Enerzijds kan je je H-schijf ***mounten*** (aankoppelen). Dit zorgt ervoor dat je H-schijf aangesproken kan worden als een externe schijf. Meer uitleg vind je op de website van de UGent helpdesk (`http://helpdesk.ugent.be/netdisk`). Om je H-schijf te kunnen *mounten* moet eerst een verbinding gemaakt worden met UGentNet. Dit gebeurt via een **VPN verbinding**. Uitleg hieromtrent is eveneens te vinden op de website van de UGent-helpdesk.

Anderzijds kan je inloggen op **Athena** (`https://athena.ugent.be`), waar gedurende een sessie de H-schijf beschikbaar is in de verschillende programma's die op Athena aangeboden worden. De verschillende versies van MATLAB zijn te vinden in de map `Academic`. Binnen UGentNet is het mogelijk om zonder meer op Athena in te loggen. Buiten UGentNet dien je ofwel een VPN-verbinding op te zetten en dan te surfen naar `http://athena.ugent.be`, ofwel in te loggen op `http://athenax.ugent.be` waarvoor geen VPN-verbinding vereist is.

Hou je H-schijf overzichtelijk! Deze zal immers gedurende de volgende jaren gebruikt worden voor verschillende cursussen. Maak per cursus een map aan en verdeel deze mappen in submappen die elk de bestanden van één practicum bevatten.

1.2 Een eerste kennismaking

Start MATLAB versie 9.5 (R2018b) op. Dit is de versie die we in deze cursus steeds zullen gebruiken. Het programmavenster dat wordt weergegeven op je computerscherm is afgebeeld in Fig. 1.1.



Figuur 1.1: De MATLAB-omgeving.

Zoals je ziet, omvat het programmavenster meerdere deelvensters die gedetailleerd zullen worden besproken in Sectie 1.3. Voorlopig kunnen we onze aandacht beperken tot het instructievenster (*Command Window*). Dit is het meest centrale venster van MATLAB. Een instructie kan in dit venster ingegeven worden na de MATLAB-prompt (`>>`) waarna deze kan uitgevoerd worden door op de `Enter`-toets te drukken.

Net als op een zakrekenmachine zijn de volgende wiskundige bewerkingen gedefinieerd in MATLAB, waarbij de gebruikelijke volgorde van bewerkingen van toepassing is:

`+`	optelling
`-`	aftrekking
`*`	vermenigvuldiging
`/`	deling
`^`	machtsverheffing
`sqrt`	worteltrekking

Voer in het instructievenster de uitdrukking `2 + 1` in en druk op de `Enter`-toets. Dit levert onmiddellijk:

```
ans =
     3
```

Merk hierbij op dat het gebruik van extra spaties voor en na het `+`-teken toegelaten is, hetgeen de leesbaarheid veelal bevordert. Het uitvoeren van de uitdrukking `4*15 + 6*12 + 2*80` levert onderstaand resultaat:

```
ans =
     292
```

Het resultaat van beide uitdrukkingen werd automatisch toegekend aan een **variabele** met de naam **`ans`** (afkorting van *answer*). Een variabele geeft toegang tot een stukje geheugen van één of meer opeenvolgende bytes en heeft

- een **naam** die overeenkomt met het geheugenadres van de eerste byte,
- een **datatype** dat het aantal opeenvolgende bytes bepaalt en betekenis aan de waarde geeft,
- een **waarde**, nl. de inhoud van de geheugenlocatie(s).

Namen van variabelen zijn **hoofdlettergevoelig** en dienen steeds te beginnen met een letter die enkel mag gevolgd worden door letters, cijfers of *underscores* (`_`). In onderstaande sessie worden enkele variabelen gedefinieerd in het instructievenster:

```
>> appels = 4
appels =
     4
>> bananen = 6
bananen =
     6
>> meloenen = 2;
>> fruit = appels + bananen + meloenen
fruit =
    12
>> totale_kost = appels*25 + bananen*22 + meloenen*99
totale_kost =
   430
```


De waarden die toegekend worden zijn ofwel letterlijke waarden (de drie eerste toekenningen), ofwel het resultaat van de evaluatie van een uitdrukking (de twee laatste toekenningen). Het plaatsen van een puntkomma (`;`) aan het einde van de instructie zorgt ervoor dat deze wel wordt uitgevoerd, maar dat het uitschrijven van het resultaat naar het scherm wordt onderdrukt. Verder geeft bovenstaand voorbeeld aan dat de actuele waarde van een variabele later gebruikt, opgevraagd of gewijzigd kan worden. Na uitvoering van bovenstaande instructies, kun je het instructievenster opnieuw leegmaken met de instructie **`clc`**.

Merk op dat het gelijkheidsteken (=) in MATLAB een andere betekenis draagt dan het gelijkheidsteken uit de wiskunde. Het draagt de betekenis '*wordt gelijkgesteld aan*' en staat voor een toekenning. Zo betekent de toekenning `x = 1` bijvoorbeeld '*de waarde van de variabele* `x` *wordt gelijkgesteld aan 1*'. Aangezien het gelijkheidsteken in MATLAB het rechterlid toekent aan het linkerlid, dient het linkerlid steeds een variabele te zijn. Zo is de toekenning `x = 1` geldig, maar is de toekenning `1 = x` ongeldig vermits het niet mogelijk is om de variabele `x` toe te kennen aan de waarde `1`. Tenslotte merken we op dat bij de toekenning van een waarde aan een variabele, deze variabele ook in het rechterlid mag voorkomen:

```
>> x = 1;
>> x = x + 1
x =
     2
```

MATLAB heeft een enorm aanbod aan **ingebouwde** functies, waaronder (tri)goniometrische, logaritmische, hyperbolische en exponentiële functies (*cf.* Tab. 1.1). Een **functieoproep** bestaat uit de **naam** van de functie, eventueel gevolgd door een **met ronde haken omsloten lijst van inputs** die gescheiden worden door komma's. Een functie geeft meestal een waarde terug:

```
>> exp(1)
ans =
    2.7183

>> sin(pi/4)
ans =
    0.7071

>> log(exp(3))
ans =
     3

>> rem(7, 2)
ans =
     1
```

Merk in de voorlaatste instructie op dat het resultaat van een functieoproep (hier de functie `exp`) kan gebruikt worden als input voor een andere functie (hier de functie `log`). Verder merken we op dat MATLAB het **punt (.) als decimaal scheidingsteken** gebruikt.

Tabel 1.1: Overzicht van enkele operatoren en functies met betrekking tot getallen.

wiskundige operatoren	
`a + b`	optellen van `a` en `b`
`- x`	tegengestelde van `x`
`a - b`	aftrekken van `b` van `a`
`a * b`	vermenigvuldigen van `a` en `b`
`a / b`	delen van `a` door `b`
`a ^ b`	machtsverheffing van `a` tot `b`
`sqrt(x)`	worteltrekking van `x`
*goniometrische functies (in **radialen**)*	
`sin(x)`	de sinus van `x`
`asin(x)`	de inverse sinus van `x`
`cos(x)`	de cosinus van `x`
`acos(x)`	de inverse cosinus van `x`
`tan(x)`	de tangens van `x`
`atan(x)`	de inverse tangens van `x`
`cot(x)`	de cotangens van `x`
`acot(x)`	de inverse cotangens van `x`
*goniometrische functies (in **graden**)*	
`sind(x)`	de sinus van `x`
`cosd(x)`	de cosinus van `x`
`tand(x)`	de tangens van `x`
`cotd(x)`	de cotangens van `x`
exponentiële en logaritmische functies	
`exp(x)`	de exponentiële functie van `x` (grondtal e)
`log(x)`	de natuurlijke logaritme (ln) van `x`
`log10(x)`	de tiendelige logaritme ($\log_{10}$) van `x`
`log2(x)`	de logaritme met grondtal 2 ($\log_2$) van `x`
functies voor het afronden	
`fix(x)`	`x` afronden naar het volgende geheel getal in de richting van 0
`floor(x)`	`x` afronden naar het volgende kleiner geheel getal
`ceil(x)`	`x` afronden naar het volgende groter geheel getal
`round(x)`	`x` afronden naar het dichtstbijgelegen geheel getal
`rem(a, b)`	rest na **gehele** deling van `a` door `b`; `rem(a, b)` is hetzelfde als `a - fix(a/b)*b`
andere functies	
`abs(x)`	absolute waarde van `x`
`sign(x)`	`sign(x)` resulteert in `1` voor `x` groter dan nul, `0` voor `x` gelijk aan nul, en `-1` voor `x` kleiner dan nul

Opdracht 1.1

Geef achtereenvolgens onderstaande instructies in en vul de tabel aan.

input	output
`4 + 6 + 2`	
`2.7 + 1.3`	
`b = sin(pi/2)`	
`a = cos(0.5)`	
`a = cos(0, 5)`	
`x = sqrt(2)/2`	
`b + x`	
`log10(0)`	
`c = abs(-4)`	
`c^2`	
`ans + a - c*2`	
`7/0`	
`tan(pi/2)`	
`rem(6, 2)`	
`rem(7, 2)`	
`round(6.2)`	
`sign(5)`	
`sign(-5)`	

input	output
`0/0`	

Merk op dat de instructie **`tan(pi/2)`** niet oneindig teruggeeft, maar wel een zeer groot getal, nl. `1.6331e+16`. De instructie **`log10(0)`** geeft daarentegen wel **`-Inf`** (een voorstelling voor min oneindig) terug. Uitdrukkingen met een ongedefinieerd resultaat zoals `0/0` geven **`NaN`** (afkorting van *Not a Number*) terug. We komen hierop verder terug bij het bespreken van de zogenaamde drijvendekommagetallen in MATLAB in Sectie 2.3.

1.3 De MATLAB-omgeving

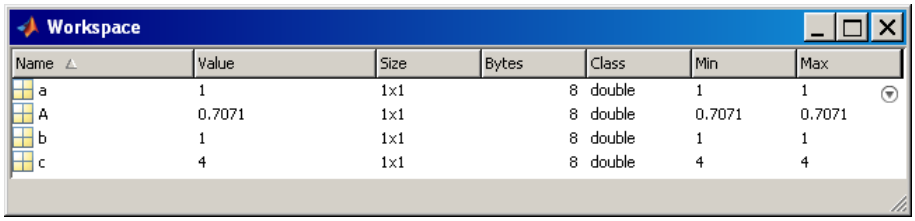
Zoals reeds aangestipt in Sectie 1.2 omvat het programmavenster van MATLAB meerdere deelvensters, waarvan voorlopig enkel het instructievenster werd belicht. In deze sectie zullen de overige deelvensters in detail aan bod komen.

1.3.1 De werkruimte

In het deelvenster 'Werkruimte' (*Workspace*) kunnen we de verschillende variabelen die gedefinieerd werden in het instructievenster, en beschikbaar zijn voor verder gebruik, bekijken. Indien we alle eerder gedefinieerde variabelen uit het geheugen wensen te wissen, kunnen we dit doen m.b.v. de functies **`clear`** of **`clear all`**. Voer één van deze functies uit en controleer dat de inhoud van de werkruimte gewist werd. Voer daarna onderstaande instructies uit:

```
>> A = sqrt(2)/2;
>> a = rem(15, 7);
>> b = sin(pi/2);
>> c = 40/10;
```

We zien deze variabelen in de werkruimte verschijnen. Indien we met de rechtermuisknop klikken op een kolomhoofding (*name* of *value*) krijgen we een lijst met alle kolommen die weergegeven kunnen worden. De opties *Name*, *Value*, *Min* en *Max* zijn standaard aangevinkt. Wanneer ook *Size*, *Bytes* en *Class* aangevinkt worden, krijg je het venster te zien dat wordt afgebeeld in Fig. 1.2. Je dient op te klikken en *Undock* te kiezen in het menu dat verschijnt om de werkruimte te ontkoppelen.



Figuur 1.2: De werkruimte met de bijkomende opties *Size*, *Bytes* en *Class* aangevinkt.

De **grootte** (*Size*) van de variabelen leert ons dat iedere variabele een scalair (1×1) is. Bij de **klasse** (*Class*) zien we het datatype staan, hier steeds een `double`. We zien in de kolom *Bytes* dat een variabele van het datatype `double` 8 bytes in het geheugen in beslag neemt. In Sectie 1.4 komen de verschillende datatypes aan bod.

Informatie over de gebruikte variabelen kan met de functies `who` en `whos` opgevraagd worden. Met eerstgenoemde krijg je een alfabetische lijst die de namen van alle variabelen in de werkruimte bevat, terwijl de functie `whos` een alfabetische lijst oplevert met de namen van alle variabelen in de werkruimte, die is aangevuld met bijkomende informatie.

```
>> who
Your variables are:
A  a  b  c

>> whos
  Name        Size              Bytes  Class     Attributes
  A           1x1                   8  double
  a           1x1                   8  double
  b           1x1                   8  double
  c           1x1                   8  double
```

Als je informatie over één enkele variabele wil opvragen, geef je de naam van die variabele als volgt mee:

```
>> whos A
  Name        Size              Bytes  Class     Attributes
  A           1x1                   8  double
```

1.3.2 De MATLAB-editor

De MATLAB-editor, die je het beste kunt vergelijken met een eenvoudige tekstverwerker, kan geopend worden door in het tabblad *HOME* op de knop *New script* () te drukken of door

eerst op () en vervolgens op () te drukken, of door op de toetsencombinatie `Ctrl + N` te drukken. In het programmavenster verschijnt een nieuw deelvenster[1], de MATLAB-editor, met daarin een leeg bestand geopend waarin je tekst kunt typen. Deze tekst kan je na aanvulling bewaren als een zogenaamde m-file die de extensie `.m` draagt. Afhankelijk van de eigenlijke inhoud, onderscheiden we twee soorten m-files: *scripts* en *functies*. Functie m-files behandelen we echter pas later in deze cursus.

Niettegenstaande instructies rechtstreeks ingegeven kunnen worden in het instructievenster, is het veelal raadzaam om deze te groeperen in een script. Een script laat je toe je opgenomen instructies later te raadplegen, eventueel te bewerken en te herhalen. Bij het sluiten van MATLAB gaan alle in het instructievenster ingegeven instructies immers verloren. Het uitvoeren van de instructies in een script gebeurt door het ingeven van de naam van het script (*i.e.* de bestandsnaam zonder de extensie `.m`) in het instructievenster. De instructies die in het bestand opgeslagen zijn, worden dan na elkaar uitgevoerd. Je kan een script ook uitvoeren door in de editor op de toets `F5` te drukken of door op de knop *Save and run* () te drukken. Het is tevens mogelijk om slechts een deel van het script uit te voeren door het gewenste deel te selecteren en op de toets `F9` te drukken.

Scripts werken met de variabelen die op dat moment in de werkruimte aanwezig zijn. Alle variabelen die in een script gecreëerd worden, blijven in de werkruimte aanwezig en kunnen gedurende het verdere verloop van de sessie gebruikt worden.

Het is belangrijk je script steeds te beginnen met enkele regels commentaar die beschrijven wat het script precies doet. Een dergelijke commentaarregel dient te beginnen met een procentteken (%). Ieder karakter op een regel na het procentteken wordt door MATLAB bij het uitvoeren genegeerd.

Namen van scripts zijn, net zoals variabelenamen, **hoofdlettergevoelig**. Ze dienen eveneens te **beginnen met** een **letter** en **enkel gevolgd** te worden **door letters**, **cijfers** of ***underscores*** (_). Let erop dat scripts en variabelen niet dezelfde naam dragen. MATLAB zal immers bij het zoeken naar een naam **eerst de lijst met variabelen in de werkruimte** doorzoeken. Hierdoor zal MATLAB geen toegang krijgen tot een script met dezelfde naam als een variabele, aangezien het de variabele eerst vindt.

Op Minerva vind je het script `cirkel.m`. Dit script berekent de oppervlakte van een cirkel met opgegeven straal 3:

Fragment 1.1: `cirkel.m`

```
1 % cirkel - Een script om de oppervlakte van een cirkel te berekenen
2 straal = 3;
3 oppervlakte = pi*straal*straal
```

[1]Door op  te klikken en *Undock* te kiezen in het menu dat verschijnt, kun je de MATLAB-editor ontkoppelen.

Opdracht 1.2

Download het script `cirkel.m` *van Minerva en plaats het in de map die je aangemaakt hebt om de bestanden van dit practicum te bewaren. Stel de huidige map in MATLAB correct in zodat je in het instructievenster de instructie*

```
>> cirkel
```

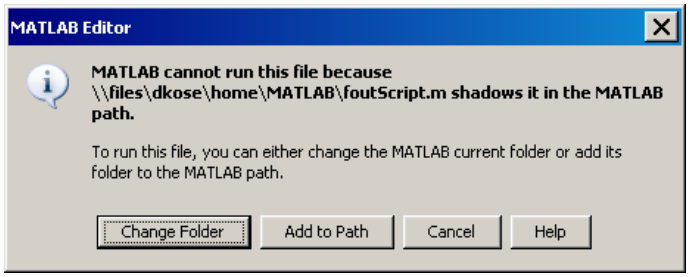
kunt ingeven en op `Enter` *drukken om het script* `cirkel.m` *uit te voeren.*

1.3.3 De huidige map

Het deelvenster 'Huidige map' *(Current Folder)* geeft de inhoud van de map weer waarin MATLAB in eerste instantie bestanden opzoekt en opslaat. Dit venster beschikt over gelijkaardige functionaliteiten als de Windows Verkenner. Zo kun je er bestanden kopiëren, plakken en wissen, van naam of locatie wijzigen, etc. In de adresbalk net boven het deelvenster zie je het pad naar de huidige map. Door te drukken op de knop kan je dit pad wijzigen. Als je op je H-schijf een map hebt aangemaakt om de bestanden te bewaren die je gedurende dit practicum nodig hebt, bv. `Practicum1`, dien je ervoor te zorgen dat het pad naar deze map staat ingesteld om scripts (of functies, zie later) uit deze map te kunnen uitvoeren. **Het is essentieel dit pad goed in te stellen.** Wanneer je een script `foutScript.m` dat zich **niet in de huidige map** bevindt toch probeert uit te voeren door de naam in te geven in het instructievenster, zal volgende foutmelding verschijnen:

```
>> foutScript
Undefined function or variable 'foutScript'.
```

Wanneer je dit script evenwel probeert uit te voeren vanuit de MATLAB-editor door te klikken op de knop *Save and run* () of door op de toets `F5` te drukken, zal een boodschap verschijnen waarin je gevraagd wordt om de huidige map correct in te stellen (zie Fig. 1.3).



Figuur 1.3: De boodschap die verschijnt als het script (of de functie) zich niet in de juiste map bevindt.

Door op de knop *Change Folder* te klikken, wordt de map waarin het script zich bevindt de huidige map.

Opdracht 1.3

Download de functie m-file `cirkelFun.m` *van Minerva en plaats ze in de map die je aangemaakt hebt om de bestanden van dit practicum te bewaren. Voer vervolgens onderstaande instructies uit:*

```
>> r = 5;                    % specificeer de input
>> opp = cirkelFun(r)        % functie-oproep met expliciete output
opp =
   78.5398

>> opp = cirkelFun(3)        % functie-oproep met expliciete output
opp =                        % en directe input
   28.2743

>> cirkelFun(2.5)            % functie-oproep zonder expliciete
ans =                        % output
   19.6350

>> cirkelFun(r^2/(r + 7))    % uitdrukking als input
ans =
   13.6354
```


1.3.4 Het logboek

Het deelvenster 'Logboek' *(Command History Window)*, dat via *Layout > Command History > Docked* als extra tabblad in het deelvesnter 'Werkruimte' verschijnt, geeft de instructies weer die in de huidige MATLAB-sessie werden uitgevoerd. Door te dubbelklikken op een instructie, wordt deze opnieuw uitgevoerd. Eerder uitgevoerde instructies kunnen daarnaast rechtstreeks in het instructievenster doorlopen (en aangepast) worden m.b.v. de pijltjestoetsen ↑ en ↓.

1.3.5 De MATLAB-help

De MATLAB-help kan op verschillende manieren geraadpleegd worden. Via het tabblad *HOME*, het helpsymbool (?) en de toets `F1` kan de zogenaamde *product help* geopend worden waarin je uitgebreide uitleg over de verschillende instructies en mogelijkheden van MATLAB

kunt terugvinden. Daarnaast beschikt MATLAB over een compendium van ingebouwde functies (*Function Browser*) (fx) waar je doorheen kunt bladeren. Het is ook mogelijk om rechtstreeks in het instructievenster een korte uitleg over een ingebouwde MATLAB-functie met de naam `fun` op te vragen m.b.v. de instructie **`help fun`**.

Een voorbeeld:

```
>> help sin
 sin    Sine of argument in radians.
    sin(X) is the sine of the elements of X.

    See also asin, sind.

    Reference page for sin
    Other functions named sin
```

Verder kan de instructie **`helpwin fun`** gebruikt worden om een grafisch venster te openen met een korte uitleg over een functie met naam `fun`, of biedt de instructie **`doc fun`** de mogelijkheid om een grafisch venster met uitgebreidere uitleg te openen.

1.4 Datatypes

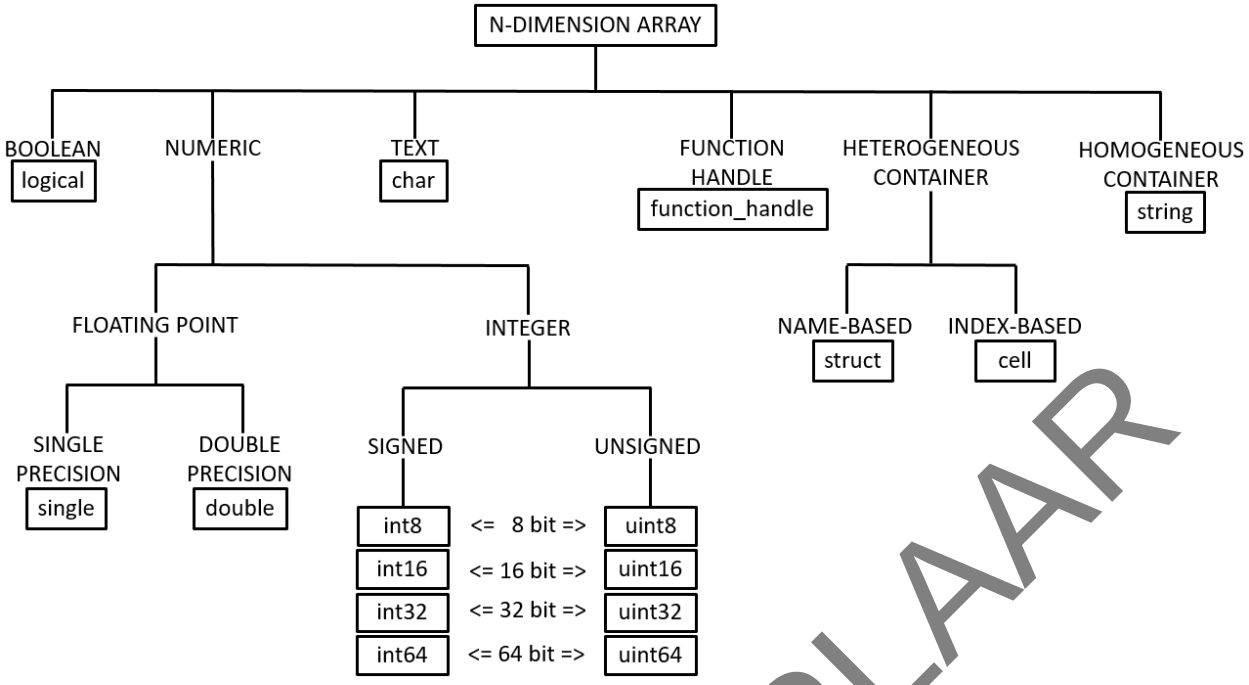
1.4.1 Inleiding

In MATLAB wordt met iedere variabele een **datatype** geassocieerd. Dit datatype bepaalt welke waarden de variabele kan aannemen, wat de betekenis van deze waarden is en welke bewerkingen met deze variabele uitgevoerd kunnen worden.

Alle datatypes in MATLAB stellen n-dimensionale tabellen of **arrays** voor. Een array is een tabelvormige collectie van elementen van eenzelfde datatype, waarin elk element geïdentificeerd wordt door een index of tupel van indices (één index per dimensie). Een array wordt intern opgeslagen op zo'n manier dat de positie van ieder element in het geheugen berekend kan worden a.d.h.v. zijn index of tupel van indices. Bij tweedimensionale arrays bestaat het tupel van indices uit de rij-index gevolgd door de kolomindex. De hiërarchie van datatypes in MATLAB wordt grafisch weergegeven in Fig. 1.4.

Een $m \times n$-matrix wordt in MATLAB opgeslagen als een tweedimensionale $m \times n$-array. Een kolomvector met m elementen wordt opgeslagen als een $m \times 1$-array en een rijvector met n elementen als een $1 \times n$-array. Een scalair, *i.e.* een getal, wordt tenslotte opgeslagen als een 1×1-array.

Een matrix wordt rij per rij ingegeven en wordt omgeven door vierkante haken. De elementen van een rij worden gescheiden door een spatie of een komma, en de rijen worden gescheiden



Figuur 1.4: De hiërarchie van datatypes (aangeduid met afgeronde rechthoeken) in MATLAB.

door een puntkomma (;) of door een nieuwe lijn te beginnen (door op de `Enter`-toets te drukken). In MATLAB wordt in de mate van het mogelijke de conventie gevolgd dat namen van vectoren met een kleine letter beginnen en namen van matrices met een grote letter. In onderstaande sessie wordt een 2×2 matrix aangemaakt en wordt het element dat zich op de eerste rij en tweede kolom bevindt, vervolgens opgevraagd:

```
>> M = [1.2, 3.5; 3.1, 0.9]  % komma's (,) overbodig
M =
    1.2000    3.5000
    3.1000    0.9000

>> M(1,2)                    % element op rij 1, kolom 2
ans =
    3.5000
```

Een rijvector kan als volgt ingegeven worden:

```
>> x = [2, 4, 6, 8, 10]      % of x = [2 4 6 8 10]
x =
     2     4     6     8    10
>> x(1,3)
ans =
     6
```

Indien er zich tussen de opeenvolgende elementen van de vector steeds een vaste stapgrootte bevindt, kan een dergelijke vector sneller (en efficiënter) gedefinieerd worden als volgt:

```
x = b:s:e
```

met

- `b` de **beginwaarde**,
- `s` de **stapgrootte** (optioneel, defaultwaarde is 1), en
- `e` de **eindwaarde**.

Bovenstaande instructie is equivalent met

```
x = [b, b+s, b+2*s, ..., b+n*s]
```

waarbij

- `n` de grootste waarde is waarvoor `b + n*s` $\leq$ `e`, als de stapgrootte `s` positief is en
- `n` de kleinste waarde is waarvoor `b + n*s` $\geq$ `e`, als de stapgrootte `s` negatief is.

Enkele voorbeelden:

```
>> v = 2:2:10
v =
2     4     6     8     10

>> z = 80:-15:-10
z =
    80    65    50    35    20     5   -10

>> x = 0:0.2:1          % stapgrootte niet-geheel
x =
       0    0.2000    0.4000    0.6000    0.8000    1.0000

>> y = 1:5              % stapgrootte s = 1
y =
     1     2     3     4     5

>> t = 10:1             % stapgrootte niet opgegeven en b > e
t =
  1x0 empty double row vector
```

Wanneer de stapgrootte `s` niet gespecificeerd wordt, gaat MATLAB ervan uit dat deze gelijk is aan 1 als `b` $\leq$ `e` en wordt er een lege matrix teruggegeven als `b` $>$ `e`. Merk ook op dat de stapgrootte `s` geen geheel getal hoeft te zijn.

Het eerste element van `x` is steeds `b`. Het laatste element van `x` is **niet noodzakelijk** gelijk aan `e`:

```
>> x = 2:3:13
x =
     2     5     8     11     % volgende waarde is 14 maar > 13
```

Opmerking: als `b` en/of `e` vectoren zijn, wordt enkel het eerste element van `b` en/of `e` gebruikt:

```
>> y = 1:3:[9, 15]
y =
     1     4     7            % 15 wordt genegeerd
```

De instructie

```
x = linspace(b, e, n)
```

met

- `b` de **beginwaarde**,
- `e` de **eindwaarde** en
- `n` het **aantal** (**equidistante**) elementen (optioneel, defaultwaarde is 100).

creëert een rijvector met `n` elementen door het interval $[b, e]$ te verdelen in `n-1` deelintervallen van gelijke grootte. De **stapgrootte** `s` wordt gegeven door `(e-b)/(n-1)`. Het eerste element van de rijvector is steeds `b` en het laatste element is steeds `e`.

Enkele voorbeelden:

```
>> rijvector = linspace(2, 10, 5)
rijvector =
     2     4     6     8    10

>> x = linspace(4, 1, 6)
x =
    4.0000    3.4000    2.8000    2.2000    1.6000    1.0000

>> y = linspace(1, 10)      % n niet opgegeven --> 100 punten
y =
  Columns 1 through 6
    1.0000    1.0909    1.1818    1.2727    1.3636    1.4545
...
  Columns 97 through 100
    9.7273    9.8182    9.9091   10.0000
```

Merk op dat wanneer `n` niet opgegeven wordt, een rijvector gecreëerd wordt met 100 equidistante elementen verdeeld over het interval $[b, e]$.

Opdracht 1.4

Maak een script `Opdracht1_4.m` *aan waarin je de vector* [10 9 ... 1 0] *creëert en toekent aan de variabele* `v`*. Achterhaal vervolgens de betekenis van de onderstaande instructies door deze één voor één in je script op te nemen en uit te voeren. Vul daartoe onderstaande tabel aan.*

input	*betekenis en output*
`v(1,4)`	
`v(1,12)`	
`v(1,0.3283)`	
`v(1,2) = 10`	

Hoe zou je de vector `v` *m.b.v. de functie* **`linspace`** *creëren?*

1.4.2 Logische waarden

Een variabele van het type **`logical`** kan slechts twee waarden aannemen: **waar** of **vals**. De waarheidswaarde waar wordt voorgesteld door een logische 1, en de waarheidswaarde vals door een logische 0. Een variabele van dit type neemt slechts één byte (*i.e.* 8 bits) in beslag en kan onder meer als volgt worden aangemaakt:

```
>> geldig = logical(1)     % of geldig = true
geldig =
     1
```

Met **`whos`** kan gekeken worden welk datatype de variabele **`geldig`** heeft:

```
>> whos geldig
  Name       Size              Bytes  Class      Attributes
  geldig     1x1                   1  logical
```

Merk op dat iedere niet-nulwaarde de waarheidswaarde waar toegekend krijgt:

```
>> res = logical(-100)
res =
     1
```

De **ingebouwde** functies `true` en `false` geven, indien je geen inputs meegeeft, resp. de logische 1 en 0 terug:

```
>> true
ans =
     1

>> false
ans =
     0
```

1.4.3 Gehele getallen

Variabelen van het type *integer* nemen **enkel gehele waarden** (*integer values*) aan, en zijn onder te verdelen in twee subtypes: de *signed integers* (datatypes `int8`, `int16`, `int32` en `int64`) en *unsigned integers* (datatypes `uint8`, `uint16`, `uint32` en `uint64`). De *signed integers* kunnen zowel positieve als negatieve gehele waarden aannemen, terwijl *unsigned integers* enkel positieve gehele waarden kunnen aannemen. Zowel *signed* als *unsigned integers* zijn beschikbaar in vier capaciteiten en nemen één byte (8 bits), 2 bytes (16 bits), 4 bytes (32 bits) of 8 bytes (64 bits) geheugenruimte in beslag (`int8` en `uint8` tot en met `int64` en `uint64`).

Een variabele van bijvoorbeeld het type `uint32` kan als volgt aangemaakt worden:

```
>> getal = uint32(29830)
getal =
       29830
```

We kunnen in het instructievenster de functie `whos` gebruiken om het datatype van de variabele `getal` op te vragen, of kunnen rechtstreeks de functie `class` gebruiken:

```
>> whos getal
  Name        Size              Bytes  Class     Attributes
  getal       1x1                   4  uint32

>> datatype = class(getal)
datatype =
    'uint32'
```

De minimum- en maximumwaarden van *integers* kunnen opgevraagd worden m.b.v. resp. de functies `intmin` en `intmax`:

```
>> min16 = intmin('int16')
min16 =
   -32768                % = -2^15

>> max16 = intmax('int16')
max16 =
   32767                 % = 2^15 - 1
```

Raadpleeg de MATLAB-help voor meer uitleg omtrent deze functies.

Opdracht 1.5

Vul onderstaande tabel aan.

datatype	minimumwaarde	maximumwaarde
int8		
uint8		
int64		
uint64		

Het omzetten van een variabele `a` naar een variabele `b` van een ander datatype `newclass` kan gebeuren door een **cast** (omzetting) met de functie `cast`:

```
b = cast(a, newclass)
```

```
>> getal = uint32(29830);
>> getal16 = cast(getal, 'uint16')
getal16 =
   29830

>> getal8 = cast(getal, 'uint8')
getal8 =
   255
```

De omzetting naar een ander datatype kan ook als volgt:

```
>> getal16 = uint16(getal)
getal16 =
   29830

>> getal8 = uint8(getal)
getal8 =
   255
```

Merk op dat de waarde van de variabele `getal` nog exact kan voorgesteld worden met een variabele van het type **`uint16`**, maar niet langer met een variabele van het type **`uint8`**. In het laatste geval is de waarde te groot en treedt **overflow** op: de maximaal toegelaten waarde voor een **`uint8`**, nl. 255, wordt toegekend aan de variabele. **Underflow** is ook mogelijk.

1.4.4 Karakters (datatype `char`)

Karakters worden voorgesteld m.b.v. het datatype **`char`**. Een karakterwaarde kan opgegeven worden door het te omgeven met enkele aanhalingstekens:

```
>> karakter = 'e'
karakter =
    'e'

>> whos karakter
  Name          Size              Bytes  Class    Attributes
  karakter      1x1                   2  char
```

Een karakter wordt intern gecodeerd door de binaire voorstelling van een niet-negatief geheel getal dat voor de eerste 128 waarden de ASCII-codering volgt en voor hogere waarden afhangt van de systeeminstelling. De ASCII-codering van de waarde van een variabele van het type **`char`** kan opgevraagd worden door de variabele te casten naar het datatype **`int16`**. Zo kan de ASCII-code van het karakter `'e'` als volgt gevonden worden:

```
>> code = int16(karakter)
code =
   101

% alternatief:
>> code = cast(karakter, 'int16')
code =
   101
```

Omgekeerd kan een ASCII-code als volgt omgezet worden naar een karakter:

```
>> karakter = char(code)
karakter =
    'e'

% alternatief:
>> code = 101;
>> karakter = cast(code, 'char')
```

```
karakter =
    'e'
```

Een **sequentie van karakters** wordt in MATLAB bewaard als een array (rijvector) van karakters:

```
>> woord = 'huis';
>> codes = int16(woord)
codes =
    104    117    105    115

>> whos woord codes  % GEEN komma (,) tussen woord en codes!
  Name       Size              Bytes  Class    Attributes
  codes      1x4                   8  int16
  woord      1x4                   8  char
```

Merk op dat de variabelen `woord` en `codes` beiden 1×4-arrays zijn. Bij de variabele `woord` is de array gevuld met waarden van het type `char`, terwijl de array bij de variabele `codes` gevuld is met waarden van het type `int16`.

Ook spaties, leestekens en cijfers kunnen uiteraard **als karakter** voorkomen in een string:

```
>> zin = 'Programmeren in MATLAB 9.5!';
>> whos zin
  Name       Size              Bytes  Class    Attributes
  zin        1x27                 54  char

>> karakter = zin(1,25)     % korter: karakter = zin(25)
karakter =
    '.'

>> code = int16(karakter)   % ASCII-code van het punt (.) opvragen
code =
   46
```

Opmerking: willen we een **array van woorden** definiëren, dan moeten we onze toevlucht nemen tot het datatype `string`. We verwijzen naar Sectie 7.1 voor meer informatie over dit datatype.

1.4.4.1 Bewerkingen op `char` arrays

MATLAB bevat een groot aantal ingebouwde functies die inwerken op `char` arrays. In hetgeen volgt, beperken we ons tot de functies in Tab. 1.3.

Tabel 1.3: Enkele handige functies die toepasbaar zijn op `char` arrays.

Functie(-oproep)	**Functionaliteit**
`count(s, zoekwoord)`	Telt het **aantal** voorkomens van `zoekwoord` in `s`. Het `zoekwoord` kan uit één of meer karakters bestaan.
`contains(s, zoekwoord)`	Controleert of `zoekwoord` **voorkomt** in `s`. Het `zoekwoord` kan uit één of meer karakters bestaan.
`replace(s, oud, nieuw)`	**Vervangt** in `s` elk voorkomen van `oud` door `nieuw`. De inputs `oud` en `nieuw` kunnen uit één of meer karakters bestaan.
`split(s, kar)`	**Splitst** `s` volgens het scheidinsteken `kar`. Het scheidingsteken `kar` kan uit één of meer karakters bestaan. Indien `kar` niet opgegeven werd: splitsing volgens spaties.

We zullen de werking van deze functies verduidelijken a.d.h.v. een aantal voorbeelden:

```
% Tellen
>> s = 'De zomermaanden zijn juni, JULI en augustus.';
>> aantal = count(s, 'i')
aantal =
     2    % de I in JULI wordt niet meegeteld!

% Controleren of iets voorkomt
>> res = contains(s, 'de')
res =
   1      % de 'De' in het begin wordt niet meegeteld!

% Zoeken en vervangen
>> s2 = replace(s, 'a', 'A')
s2 =
    'De zomermAAnden zijn juni, JULI en Augustus.'

% Splitsen
>> woorden = split(s, ',') % splitsen volgens komma
woorden =
    {'De zomermaanden zijn juni'}
    {' JULI en augustus.'       }

>> woorden = split(s)  % splitsing volgens spaties (DEFAULT)
woorden =
    {'De'          }
    {'zomermaanden'}
    {'zijn'        }
    {'juni,'       }
```

```
    {'JULI'         }
    {'en'           }
    {'augustus.'    }

>> woord = woorden{4}  % INHOUDSindexering met accolades
woord =
    'juni,'
```

Opmerkingen:

- Alle besproken functies zijn **hoofdlettergevoelig**.
- De output van de functie `split` is een `cell` array. `Cell` arrays zullen uitgebreid aan bod komen in Hoofdstuk 7. De elementen van een `cell` array kunnen opgevraagd worden door **accolades** (`{ }`) te gebruiken i.p.v. ronde haken. Dit heet **inhoudsindexering** en wordt in Sectie 7.2.2 in detail behandeld.
- Alle besproken functies zijn ook van toepassing op `string` en `cell` arrays. We verwijzen naar Hoofdstuk 7.

1.4.5 Drijvendekommagetallen

Drijvendekommagetallen worden in MATLAB gebruikt als een voorstelling van reële getallen en zijn geïmplementeerd als de datatypes `single` en `double`. De term 'drijvende komma' verwijst naar het feit dat het decimale punt kan 'drijven' en gelijk waar geplaatst kan worden relatief t.o.v. de beduidende cijfers. Het gebruikmaken van een drijvende komma laat toe om reële getallen in een groot waardenbereik voor te stellen of, in veruit de meeste gevallen, te benaderen.

Een variabele van het type `double` (dubbele precisie) neemt 8 bytes (64 bits) geheugenruimte in beslag en kan waarden tussen $2.2251\,10^{-308}$ en $1.7977\,10^{308}$ aannemen. Slechts een kleine deelverzameling van de reële getallen in dit bereik kan echter exact worden voorgesteld.

Een variabele van het type `single` (enkele precisie) neemt slechts 4 bytes (32 bits) in beslag en kan hierdoor slechts waarden tussen $1.1755\,10^{-38}$ en $3.4028\,10^{38}$ aannemen. Bovendien heeft een variabele van het type `single` een beperktere precisie dan een variabele van het type `double`: het aantal exact voorstelbare reële getallen in dit bereik is kleiner.

Het datatype `double` is het **standaard datatype voor getallen** in MATLAB. Wanneer een datatype niet werd opgegeven bij het toekennen van een getalwaarde aan de variabele, zal deze het datatype `double` toegewezen krijgen:

```
>> format long
```

```
>> r = 4.5;
>> omtrek = 2*r*pi
omtrek =
  28.274333882308138

>> whos omtrek
  Name         Size              Bytes  Class     Attributes
  omtrek       1x1                   8  double

>> omtrek32 = single(2*r*pi)
omtrek32 =
  28.2743340

>> whos omtrek32
  Name          Size              Bytes  Class     Attributes
  omtrek32      1x1                   4  single
```

De instructie `format long` zorgt ervoor dat MATLAB drijvendekommagetallen in het instructievenster weergeeft tot 15 cijfers na de komma. De default weergave, die bekomen wordt met de instructie `format short`, geeft drijvendekommagetallen slechts weer tot 4 cijfers na de komma. Raadpleeg de MATLAB-help voor meer informatie.

1.5 Matrices en vectoren

1.5.1 Indexeren van matrices en vectoren

Toegang tot een element van een matrix wordt bekomen door de rij en kolom waarin het element zich bevindt te specificeren. Zo verwijst `A(i,j)` naar het element van de matrix `A` op de `i`-de rij en in de `j`-de kolom. In het volgende voorbeeld wordt de waarde 1 toegekend aan het element op de derde rij en de vijfde kolom van `A`:

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

>> A(3,5) = 1
A =
     1     2     3     0     0
     4     5     6     0     0
     0     0     0     0     1
```

Merk op dat het aantal rijen, resp. kolommen, automatisch wordt aangevuld tot 3, resp. 5.

Ook toegang tot meerdere rijen en kolommen tegelijk is mogelijk. Zo duidt `A(m:n,k:l)` de rijen `m` tot en met `n` en de kolommen `k` tot en met `l` van `A` aan. Om alle aanwezige rijen (kolommen) te selecteren, plaatst men een dubbelpunt. Zo refereert `A(:,3:6)` naar alle elementen in de derde tot en met de zesde kolom. Dit laat ons toe om vlot met delen van matrices te werken.
Rij- en kolomvectoren kunnen eveneens geïndexeerd worden met twee indices waarbij evenwel de eerste index, resp. de laatste index, gelijk is aan 1 vermits een rijvector met n elementen een $1 \times n$-matrix is en een kolomvector met m elementen een $m \times 1$-matrix is. MATLAB laat echter toe om **vectoren** te **indexeren met slechts één index** die de positie in de rij (of kolom) aanduidt:

```
>> format                % terug naar weergave met 4 cijfers na de komma
>> x = 1:-0.2:0          % rijvector x
x =
    1.0000    0.8000    0.6000    0.4000    0.2000         0

>> x(3)                  % het derde element van x
ans =
    0.6000

>> x(1:2:5)              % element 1, 3 en 5 van x
ans =
    1.0000    0.6000    0.2000

>> y = [8; 5; 9; 4];     % kolomvector y
>> y(3:-2:1)             % elementen 3 en 1 van y
ans =
     9
     8
```

Tab. 1.4 geeft een samenvattend overzicht van de indexeringsmogelijkheden van vectoren en matrices in MATLAB.

Tabel 1.4: Indexeren van vectoren en matrices.

`x(i:j)`	het `i`-de tot en met het `j`-de element van de vector `x`
`x([k, l, m])`	een vector met het `k`-de, `l`-de en `m`-de element van de vector `x`
`x(end:-1:1)`	alle elementen van de vector `x` in omgekeerde volgorde
`x(end)`	het laatste element van de vector `x`
`A(i,j)`	het element op de `i`-de rij en de `j`-de kolom van `A`
`A(i,end)`	het element op de `i`-de rij van de laatste kolom van `A`
`A(end,j)`	het element op de laatste rij van de `j`-de kolom van `A`
`A(end,end)`	het element op de laatste rij en de laatste kolom van `A`
`A(:,j)`	een kolomvector met de elementen van kolom `j` van `A`
`A(:,m:n)`	een matrix met de elementen van kolommen `m` tot `n` van `A`
`A(:,[k, l, m])`	een matrix met de elementen van kolommen `k`, `l` en `m` van `A`
`A(i,:)`	een rijvector met de elementen van rij `i` van `A`
`A(m:n,:)`	een matrix met de elementen van rijen `m` tot `n` van `A`
`A([k, l, m],:)`	een matrix met de elementen van rijen `k`, `l` en `m` van `A`
`A(2:end-1,2:end-1)`	een matrix met alle elementen van `A` behalve deze op de rand
`A(:)`	een kolomvector met alle kolommen van `A` onder elkaar geplaatst

MATLAB zal een foutmelding genereren als er een rij- en/of kolomindex opgegeven wordt die

- **niet geheel** is, of
- **kleiner** dan of **gelijk** is aan **nul**, of groter is dan resp. het aantal rijen en/of kolommen van de beschouwde matrix.

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

>> A(0.5,1)
Subscript indices must either be real positive integers or
logicals.

>> A(4,3)
Index exceeds matrix dimensions.
```

1.5.2 Manipulatie van matrices en vectoren

Een matrix (resp. vector) wordt **getransponeerd** (kolommen worden rijen en rijen worden kolommen) door na de naam van de matrix (resp. vector) een **enkele aanhalingsteken** (') te plaatsen of de functie `transpose` te gebruiken:

```
A =
     1     2     3     0     0
     4     5     6     0     0
     0     0     0     0     1

>> A'
ans =
     1     4     0
     2     5     0
     3     6     0
     0     0     0
     0     0     1

>> v = [10 20 30]
v =
    10    20    30
>> transpose(x)
ans =
    10
    20
    30
```

Aan een bestaande matrix kan gemakkelijk een rij toegevoegd worden op voorwaarde dat die rij dezelfde lengte heeft als de rijen van de matrix. Hetzelfde geldt voor kolommen. De instructie `A = [A; u]` voegt de rijvector `u` toe, terwijl `A = [A, v]` de kolomvector `v` toevoegt. Aan een lege matrix (voorgesteld door `[]`) kan een rij of kolom met om het even welke lengte toegevoegd worden.

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

>> A = [A; u]
A =
     1     0     0
     0     1     0
     0     0     1
```

```
     5     6     7

>> B = [0 1 0; 1 0 1; 0 1 0];
>> B = [B, v]
B =
     0     1     0     2
     1     0     1     3
     0     1     0     4

>> B = [B; u]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.
```

Merk op dat er een foutmelding verschijnt bij de laatste instructie. De vector `u` bestaat immers slechts uit drie elementen terwijl de rijen van de matrix `B` op dat moment vier elementen bevatten.

Ook matrices kunnen geconcateneerd (samengevoegd) worden, op voorwaarde dat hun dimensies compatibel zijn:

```
>> M = [1 2 3; 4 5 6; 7 8 9; 10 11 12]
M =
     1     2     3
     4     5     6
     7     8     9
    10    11    12

>> N = [13 14 15; 16 17 18]
N =
    13    14    15
    16    17    18

>> [M; N]            % verticale concatenatie
ans =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
    13    14    15
    16    17    18

>> [M, N]            % horizontale concatenatie
Error using horzcat
```

```
Dimensions of matrices being concatenated are not consistent.
```

Rijen of kolommen kunnen verwijderd worden door er een lege matrix (`[]`) aan toe te kennen:

```
>> A(2,:) = [];      % verwijdert de 2de rij van A
>> A(:,2:4) = [];    % verwijdert de 2de tot en met 4de kolom van A
>> A([1, 3],:) = []; % verwijdert de 1ste en de 3de rij van A
```

1.5.3 Rekenen met vectoren

Een bewerking op een vector $1 \times n$ rijvector of een $n \times 1$ kolomvector wordt uitgevoerd op elk vectorelement en resulteert in een nieuwe rij- of kolomvector:

```
>> v = [5, -1, 6];
>> v - 5
ans =
     0    -6     1

>> v/3
ans =
    1.6667   -0.3333    2.0000
```

Twee rij- of twee kolomvectoren kunnen bij elkaar opgeteld of van elkaar afgetrokken worden enkel en alleen als beide vectoren dezelfde lengte hebben:

```
>> v = [5, -1, 6];
>> w = [1, 2, 3];
>> v + w
ans =
     6     1     9

>> u = [7, 8, 9, 10]
>> u + v
Matrix dimensions must agree.
```

Let op: als je een $1 \times n$ **rijvector** `r` en een $m \times 1$ **kolomvector** `k` bij elkaar optelt (of aftrekt), dan worden beide vectoren eerst uitgebreid tot $m \times n$ matrices: de kolomvector `k` wordt n keer verticaal gerepliceerd (herhaald), de rijvector `r` wordt m keer horizontaal gerepliceerd:

```
>> r = [1, 2, 3, 4];    % 1 x 4 rijvector   (n = 4)
>> k = [5; 6];          % 2 x 1 kolomvector (m = 2)
>> som = r + k          % of r + k
som =                   % resultaat is een 2 x 4 matrix
     6     7     8     9
     7     8     9    10

>> R = [r; r];          % verticale replicatie van r
>> K = [k, k, k, k];    % horizontale replicatie van k
>> som = R + K          % of K + R
som =
     6     7     8     9
     7     8     9    10
```

1.5.4 Rekenen met matrices

Een bewerking op een $m \times n$ matrix wordt uitgevoerd op elk matrixelement en resulteert in een nieuwe matrix. Voor de matrix A gegeven door

$$A = \begin{bmatrix} 2 & 9 \\ \sqrt{5} & -7 \end{bmatrix},$$

kan de bewerking $3 \times A$ in MATLAB eenvoudig uitgevoerd worden als volgt:

```
>> A = [2  9; sqrt(5)  -7];
>> 3*A
ans =
    6.0000   27.0000
    6.7082  -21.0000
```

De optelling (+) en aftrekking (-) worden elementsgewijs uitgevoerd op matrices met dezelfde dimensies. Zij bijvoorbeeld

$$B = \begin{bmatrix} -1 & \sin(\pi/4) \\ 3.5 & -8 \end{bmatrix},$$

dan vinden we $A + B$ als volgt:

```
>> B = [-1 sin(pi/4); 3.5 -8];
>> som = A + B
```

```
som =
    1.0000    9.7071
    5.7361  -15.0000
```

De vermenigvuldiging (`*`), de deling (`/`) en de machtsverheffing (`^`) kunnen op twee manieren uitgevoerd worden: als matrixbewerking of elementsgewijs. Door één van de operatoren vooraf te laten gaan door een punt (.), wordt aangegeven dat de bewerking **elementsgewijs** dient te gebeuren. Zo kan een elementsgewijze vermenigvuldiging uitgevoerd worden door gebruik te maken van de operator `.*`, bijvoorbeeld `Z = X.*Y`. Bij deze bewerking zal elk element van de matrix `X` vermenigvuldigd worden met het overeenkomstig element van de matrix `Y`. De resultaten worden bewaard in de matrix `Z` met dezelfde dimensies als de matrices `X` en `Y`. Merk op dat elementsgewijze bewerkingen enkel uitgevoerd kunnen worden op matrices met dezelfde dimensies, en dat matrixbewerkingen enkel uitgevoerd kunnen worden op matrices met compatibele dimensies.

De elementsgewijze vermenigvuldiging van `A` en `B` wordt uitgevoerd als volgt:

```
>> product1 = A.*B
product1 =
   -2.0000    6.3640
    7.8262   56.0000
```

Indien we echter de klassieke matrixvermenigvuldiging wensen uit te voeren kan dit als volgt:

```
>> product2 = A*B
product2 =
   29.5000  -70.5858
  -26.7361   57.5811
```

Opmerking: voor twee vectoren `x` en `y` met dezelfde lengte is `x.*y` enkel gedefinieerd indien beide rij- of kolomvectoren zijn met gelijke lengte. Zoniet wordt een foutmelding gegenereerd.

Opdracht 1.6

De vlucht van een propellervliegtuig met herkomst Brussel en bestemming Sint-Petersburg kan opgesplitst worden in vier delen. Tab. 1.5 bevat de snelheid van het vliegtuig tijdens elk deel en de duur van elk vluchtdeel.
Maak een script `Opdracht1_6.m` *aan waarin je de volgende zaken uitvoert.*

1. *Geef de snelheden in als een* ***rijvector*** `v` *en de vluchttijden als een* ***rijvector*** `t`*.*
2. *We weten dat de afgelegde afstand x berekend wordt als $x = v\,t$ waarbij x, t de tijd en v de snelheid voorstellen. Geef een instructie die een vector* `x` *aanmaakt die de afgelegde afstand* ***voor elk deel*** *van de vlucht bevat, en voer deze uit.*

Tabel 1.5: Snelheden en vluchttijden van elk deel van de vlucht.

deel	1	2	3	4
snelheid [km h^{-1}]	200	250	400	300
duur [uren]	2	3	2	1

3. *Bepaal de totaal afgelegde afstand s door het* ***matrixproduct*** *van de rijvector* `v` *met de kolomvector* `t'` *te berekenen.*

4. *Bereken de gemiddelde snelheid v_{gem} van het vliegtuig.*

MATLAB bevat een ingebouwde functie `inv` die de inverse van een matrix berekent. Een matrix is inverteerbaar (of regulier) als haar determinant verschillend is van nul. Met behulp van de inverse van een reguliere matrix kunnen vierkante stelsels, *i.e.* stelsels met evenveel vergelijkingen als onbekenden, opgelost worden.

Beschouw bijvoorbeeld het volgend vierkant stelsel met drie vergelijkingen:

$$\begin{cases} 3\,x + 2\,y - 9\,z = -65 \\ -9\,x - 5\,y + 2\,z = 16 \\ 6\,x + 7\,y + 3\,z = 5\,. \end{cases}$$

In matrixvorm kan dit stelsel geschreven worden als

$$\begin{bmatrix} 3 & 2 & -9 \\ -9 & -5 & 2 \\ 6 & 7 & 3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -65 \\ 16 \\ 5 \end{bmatrix}$$

of, in verkorte notatie, als $A\mathbf{x} = b$, met A de coëficiëntenmatrix, b het rechterlid en $\mathbf{x}$ de onbekenden. Door nu zowel het linker- als het rechterlid links te vermenigvuldigen met de inverse van A, nl. A^{-1}, krijgen we:

$$A^{-1}A\mathbf{x} = A^{-1}b\,.$$

Aangezien $A^{-1}A = I$, met I de eenheidsmatrix, volgt uiteindelijk dat $\mathbf{x} = A^{-1}b$. Voor het voorbeeld vinden we dat $[x \;\; y \;\; z]^T = [2 \;\; -4 \;\; 7]^T$.

Tab. 1.6 geeft tenslotte een samenvattend overzicht van de in MATLAB gedefinieerde matrixoperatoren.

Tabel 1.6: Overzicht van matrixoperatoren in MATLAB.

`A + B`	optellen van matrices `A` en `B`
`A - B`	aftrekken van matrices `A` en `B`
`A * B`	vermenigvuldiging van matrix `A` met matrix `B`
`A ^ n`	machtsverheffing van matrix `A` tot de macht `n`
`sqrt(A)`	elementsgewijze worteltrekking van matrix `A`
`inv(A)`	berekent de inverse van de matrix `A`
`A \ b`	lost het stelsel $A\mathbf{x} = b$ op

Opdracht 1.7

Bepaal de oplossing van het volgende stelsel in MATLAB:

$$\begin{cases} 3\,x + 2\,y - \ z = 1 \\ 2\,x - 2\,y + 4\,z = -2 \\ -2\,x + 1\,y - 2\,z = 0 \end{cases} \qquad \begin{cases} x = \\ y = \\ z = \end{cases}$$

Opdracht 1.8

Definieer de vectoren $u = [5 \quad 6 \quad 7]$, $v = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$ *en de matrix* $A = \begin{bmatrix} 1 & 8 & 3 \\ 7 & 5 & 9 \\ 4 & 2 & 6 \end{bmatrix}$ *in een script* `Opdracht1_8.m` *en ken deze toe aan, resp. de variabelen* `u`, `v` *en* `A`.

1. *Hoe vraag je de volgende zaken op:*
   - *het derde element van* `u`*:*
   - *het tweede en derde element van* `v` *(als vector):*
   - *het element op de eerste rij en derde kolom van* `A`*:*
2. *Bereken het aantal bytes dat de vectoren* `u` *en* `v` *en de matrix* `A` *in beslag nemen in het geheugen. Controleer het resultaat van je berekening door deze te vergelijken met de gegevens in de werkruimte of door gebruik te maken van de functie* **`whos`**.
3. *Vul onderstaande tabel verder aan. Bepaal steeds het datatype van de output. Controleer je antwoord door deze te vergelijken met de gegevens in de werkruimte of door gebruik te maken van de functie* **`whos`**. *Raadpleeg de MATLAB-help indien nodig.*

input	betekenis en output
`A(2,3) = 4`	
`A(:,3)`	
`A(1,:)`	
`A(:)`	
`max(A)`	
`[m, pos] = max(v)`	
`max(A, 6)`	
`sum(A)`	
`prod(u)`	
`B = inv(A)`	
`d = diag(A)`	
`B = diag(d)`	
`[A; u]`	
`a = det(A)`	
`A = [A, v]`	
`A = [A; u]`	
`B = A'`	
`A(:,2) = []`	
`A(2,6) = 1`	
`D = zeros(2, 3)`	

input	betekenis en output
`E = zeros(3)`	
`F = ones(2, 3)`	
`H = eye(3)`	
`rand(1)`	
`I = rand(3, 2)`	
`T = true(2, 4)`	
`U = false(3)`	
`[r, k] = size(D)`	
`size(F, 1)`	
`size(F, 2)`	
`length(v)`	
`length(T)`	
`length(I)`	

Merk op dat de functie `length` *in het geval van een vector de lengte, en in het geval van een matrix de grootste dimensie teruggeeft.*

4. *Bepaal de matrix die je bekomt door achtereenvolgens onderstaande manipulaties uit te voeren op de matrix* $A = \begin{bmatrix} 1 & 8 & 3 \\ 7 & 5 & 9 \\ 4 & 2 & 6 \end{bmatrix}$.

   - *Deel de elementen van de tweede kolom door* $\sqrt{3}$.

- *Tel de elementen van de derde rij op bij de elementen van de eerste rij.*

- *Vermenigvuldig de elementen van de eerste kolom met -2 en tel ze op bij de corresponderende elementen van de derde kolom (de eerste kolom blijft onveranderd).*

5. *Hoe zou je het grootste element van de bekomen matrix bepalen?*

1.5.5 Vectorisatie

In veruit de meeste gevallen (en de enige gevallen die wij momenteel beschouwen) zijn variabelen in MATLAB tweedimensionale arrays of m.a.w. matrices ($m \times n$-arrays), vectoren ($1 \times n$-arrays of $n \times 1$-arrays) of scalairen (1×1-arrays). Alle ingebouwde functies in Tab. 1.1 nemen niet enkel scalairen als input, maar ook matrices en vectoren. In het geval van een matrix (resp. vector) worden deze functies elementsgewijs toegepast, en zal het resultaat van de functieoproep dus een matrix (resp. vector) zijn. Dit is niet bij iedere ingebouwde functie het geval. Het is efficiënter om berekeningen en bewerkingen in MATLAB uit te voeren op (delen van) matrices, dan iteratief op elk element van de beschouwde matrix afzonderlijk.

Stel dat we de functie **sin** op elk van de elementen van de vector $[0 \;\; \pi/6 \;\; \pi/4 \;\; \pi/3 \;\; \pi/2]$ willen toepassen. Indien deze functie niet gevectoriseerd zou zijn, zouden we ze vijf keer moeten oproepen i.p.v. één enkele keer:

```
>> hoek = [0, pi/6, pi/4, pi/3, pi/2]
hoek =
         0    0.5236    0.7854    1.0472    1.5708

>> sinus(1) = sin(hoek(1)); % iteratief op elk element afzonderlijk
>> sinus(2) = sin(hoek(2));
>> sinus(3) = sin(hoek(3));
>> sinus(4) = sin(hoek(4));
>> sinus(5) = sin(hoek(5))
sinus =
         0    0.5000    0.7071    0.8660    1.0000
```

```
>> sin(hoek)                % gevectoriseerde oproep
ans =
         0    0.5000    0.7071    0.8660    1.0000
```

Zorg er steeds voor dat je bewerkingen in de mate van het mogelijke op **gevectoriseerde** wijze uitvoert, m.a.w. dat ze toepasbaar zijn op vectoren en matrices. Laten we als voorbeeld teruggrijpen naar het script `cirkel.m` uit Sectie 1.3.2. Vervangen we hierin de waarde van de variabele `straal` door de vector `[1, 3]`, dan zal MATLAB bij uitvoer van het script een foutmelding genereren:

```
>> cirkel
Error using  *
Inner matrix dimensions must agree.

Error in cirkel (line 3)
oppervlakte = pi*straal*straal
```

Het script werkt duidelijk niet gevectoriseerd, daar enkel het specifieke geval van 1×1-matrices, *i.e.* scalairen, ondersteund wordt. De oplossing bestaat erin de instructie voor het berekenen van de oppervlakte aan te passen:

```
oppervlakte = pi*straal.*straal
```

Door een punt te plaatsen vóór het vermenigvuldigingsteken gebeurt de berekening van de oppervlakte nu gevectoriseerd en krijgen we het gewenste resultaat terug:

```
>> cirkel
oppervlakte =
    3.1416   28.2743
```

Opdracht 1.9

Maak een nieuw script `Opdracht1_9.m` *aan waarin je eerst de vectoren* $u = [5 \quad 6 \quad 7]$, $v = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix}$ *en de matrix* $A = \begin{bmatrix} 1 & 8 & 3 \\ 7 & 5 & 9 \\ 4 & 2 & 6 \end{bmatrix}$ *definieert (cf. Opdracht 1.8), en vervolgens de instructies opneemt die je toelaten om de volgende zaken te berekenen. Als een bepaalde instructie een foutmelding genereert, plaats je deze in commentaar door deze te laten voorafgaan door een procentteken (%). Zo wordt deze instructie overgeslagen, en kun je de volgende instructies alsnog uitvoeren.*

- *de* ***sinus*** *van de elementen van* `u`*,*
- *de* ***absolute waarde*** *van de elementen van vector* `v`*,*

- *de* ***rest na deling door 3*** *van de elementen van* `A`*,*
- *de* ***natuurlijke logaritme*** *van de elementen van* `u`*,*
- *de* ***som*** *van de elementen van de vectoren* `u` *en* `v`*,*
- *het* ***matrixproduct*** *van de vectoren* `u` *en* `v'`*,*
- *de* ***quotiënten*** *van de sinus van de elementen van* `u` *en de elementen van* `u`*.*

1.6 Extra opdrachten

1. Maak de volgende vectoren aan. Doe dit indien mogelijk op twee manieren.
   - een vector `u` met 25 elementen tussen 0 en 2π
   - een kolomvector `v` met 5 willekeurige getallen tussen 0 en 1

2. Maak een willekeurige vector `x` met minstens drie elementen aan. Geef de volgende instructies in MATLAB in:

```
>> x.^1/3
>> x.^(1/3)
```

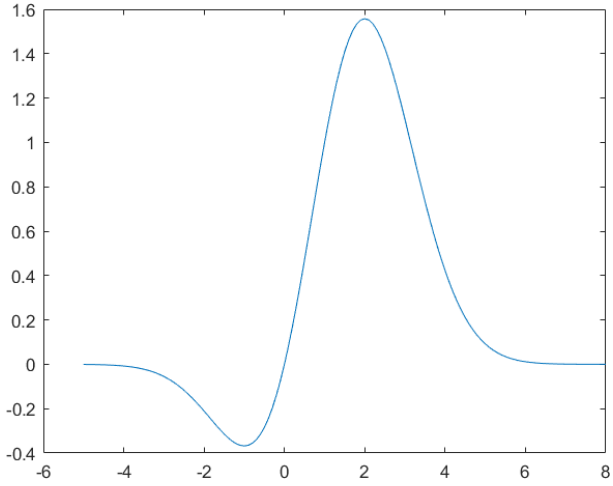
   Geven beide instructies hetzelfde resultaat? Waarom?

3. Beschouw de functie

$$f : x \to y = x\, e^{-\left(\frac{x-\mu}{\sigma}\right)^2}.$$

   Stel $\mu = 1$, $\sigma = 2$, maak een vector x aan met de getallen $-5, -4.9, -4.8, \ldots, 7.8, 7.9, 8$ en bereken $f(x)$ door gebruik te maken van vectorisatie. Stel de berekende functiewaarden $f(x)$ voor als een vector `y`. Met de instructie `plot(x, y)` kan je nu het resultaat visualiseren door de `y`-waarden uit te laten zetten tegenover de `x`-waarden. Je zou Fig. 1.5 moeten bekomen.

4. Beschouw een vector $a = [1\ 3\ 5\ 7\ 9]$ en een vector $b = [0\ 2\ 4\ 6\ 8]$. Bepaal met MATLAB
   (a) de som van a en b,
   (b) het verschil van a en b,



Figuur 1.5: Resultaat van Opdracht 3.

(c) het matrixproduct van a^T en b,

(d) de som van het product $a. * b$,

(e) de determinant van $a^T b$, en

(f) het matrixproduct van a en b^T.

5. Bereken in MATLAB het product $\prod_{i=1}^{25} i = 1 \cdot 2 \cdot 3 \cdots 25$ zonder gebruik te maken van 24 afzonderlijke vermenigvuldigingen. Raadpleeg de MATLAB-help voor meer informatie over de ingebouwde functies `prod` en/of `factorial`.

6. Beschouw twee variabelen `x` en `y`. Hoe kan je de waarden die toegekend zijn aan de variabelen `x` en `y` omwisselen, zodat `y` de waarde van `x` krijgt en vice versa, en dit zonder gebruik te maken van de waarden zelf?

7. Beschouw onderstaande matrix A en stel deze voor in een variabele `A`:

$$A = \begin{bmatrix} 11 & 2 & 8 \\ 6 & 3 & 7 \\ 5 & 9 & 4 \\ 1 & 12 & 10 \end{bmatrix}$$

Geef instructies voor

- het bepalen van het grootste element van `A`:

- het bepalen van de som van de rijen van `A`:

- het toevoegen van een kolom met 5'en aan de linkerzijde van `A`:

- het toevoegen van een rij met 0'en aan de onderzijde van `A`:

- het verwijderen van de laatste kolom van `A`:

8. Beschouw een $m \times n$-matrix `M` met $m > 6$ en $n > 6$. Zorg ervoor dat de instructies die je schrijft, toegepast kunnen worden op een matrix `M` van willekeurige grootte.

   - Geef hieronder de instructie(s) om de even rijen (resp. oneven kolommen) van `M` te selecteren en in een nieuwe matrix `E` (resp. `O`) op te slaan:

   - Geef hieronder de instructie(s) om alle rijen (resp. kolommen) met een rij(resp. kolom)-index die een veelvoud is van 3, te selecteren:

   - Geef hieronder de instructie(s) om de volledige matrix `M` zonder rand te selecteren:

   - Geef hieronder de instructie(s) om tussen de vijfde en zesde rij van M een rij met 3'en toe te voegen:

   - Geef hieronder de instructie(s) om alle kolommen vanaf de 4-de kolom van M te verwijderen: