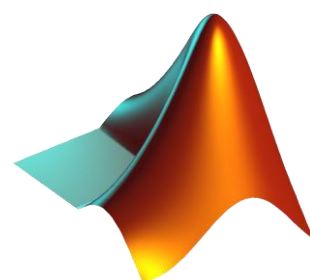




BIOSTATICA

Door Mark Schrauwen en Alistair Vardy



Matlab Wk1.1

Inhoudsopgave

Versiebeheer	2
1 Introductie van deze cursus en dit document	3
2 Waarom Matlab?	4
3 De Matlab omgeving	6
3.1 Command Window (A)	8
3.1.1 Command prompt	8
3.2 Workspace (B)	8
3.3 Current Folder (C)	8
3.4 Toolstrip (D)	9
3.5 Script Editor (E)	9
3.6 Samenhang tussen een script en het Command Window	11
3.7 Vragen	12
3.8 Antwoorden	13
4 Matlab operaties	14
4.1 Vermenigvuldigen	15
4.2 Delen	15
4.3 Machtsverheffen	15
4.3.1 Hogere machten	16
4.4 Haakjes	17
4.4.1 Een specifiek voorbeeld	17
4.5 Worteltrekken	17
4.6 Vragen en opdrachten	19
4.7 Antwoorden	20
5 Variabelen	21
5.1 Aanmaken van een variabele	21
5.2 Workspace	22
5.3 Tekst (strings)	24
5.3.1 Aan elkaar koppelen van strings	25
5.4 Vragen en Opdrachten	29
5.5 Antwoorden	30
5.6 Vectoren	31
5.6.1 Vector index (indices)	31
5.7 Vectoren optellen	33
5.7.1 Genereren van een numerieke vector	33
5.7.2 Optellen van vectoren	33
5.8 Element-by-Element vector operaties	34
5.8.1 Aanmaken van vectoren	35
5.9 Type vectoren	36
5.10 Vectoren van vectoren (een Matrix)	36
5.11 Vragen en opdrachten	38
5.12 Antwoorden	40
5.13 Veel gebruikte vectorfuncties	41
5.13.1 whos()	41
5.13.2 size()	41
5.13.3 length()	42
5.13.4 randn()	42
5.13.5 fliplr()	43

Versiebeheer

Versie	Datum	Beschrijving	Door
0.0	01-07-2017	Eerste versie	Mark Schrauwen
0.1	11-07-2017	Verbeteringen doorgevoerd n.a.v. commentaar Bart van Trigt.	Mark Schrauwen
0.2	29-08-2017	Verbetering doorgevoerd n.a.v. commentaar Denice Vis en Timothy Roos	Mark Schrauwen
0.3	01-09-2017	Kleine verbeteringen en aanvullingen doorgevoerd. Commentaar van studenten verwerkt.	Mark Schrauwen
0.4	25-09-2017	Commentaar van Herre Faber verwerkt.	Mark Schrauwen
0.5	02-10-2018	Kleine verbeteringen doorgevoerd. Opmerkingen van Denice Vis en Chadler Wilson verwerkt.	Mark Schrauwen

1 Introductie van deze cursus en dit document

Deze cursus bestaat uit een aantal hulpmiddelen die op Blackboard staan. Deze en de overige readers vormen de rode draad van de cursus. De readers bevatten alle informatie die een student nodig heeft om kennis te maken met Matlab en om het tentamen te doen in lesweek 5. De readers komen met ondersteunde materiaal zoals online oefenopgave en in de toekomst ook video's.

Er zijn voor deze cursus geen hoorcolleges. Er zijn alleen interactieve Matlab sessies waarbij de docent iets voor doet en/of uitlegt tijdens gedurende het practicum (werken op zaal). Op de overige momenten werkt de student deze reader door en de bijbehorende opdrachten door.

Aan het einde van elke week (na twee practica) krijgt de student een individuele opdracht. Deze opdrachten moeten op de **woensdag daarop om uiterlijk 23:59u** zijn ingeleverd via Blackboard.

Zie je een fout? Of heb je een aanbeveling, dan horen we dat graag! Stuur dan een e-mail naar mjschrau@hhs.nl en wij passen het dan z.s.m. aan.

2 Waarom Matlab?

Wat is Matlab? Waarom moet jij Matlab gebruiken? Wat heb je aan Matlab? Dat zijn vragen die je jezelf kunt stellen als je als Bewegingstechnoloog voor het eerst met Matlab gaat werken. Want Matlab ga je de rest van je opleiding gebruiken en is misschien wel het belangrijkste softwarepakket voor een Bewegingstechnoloog.

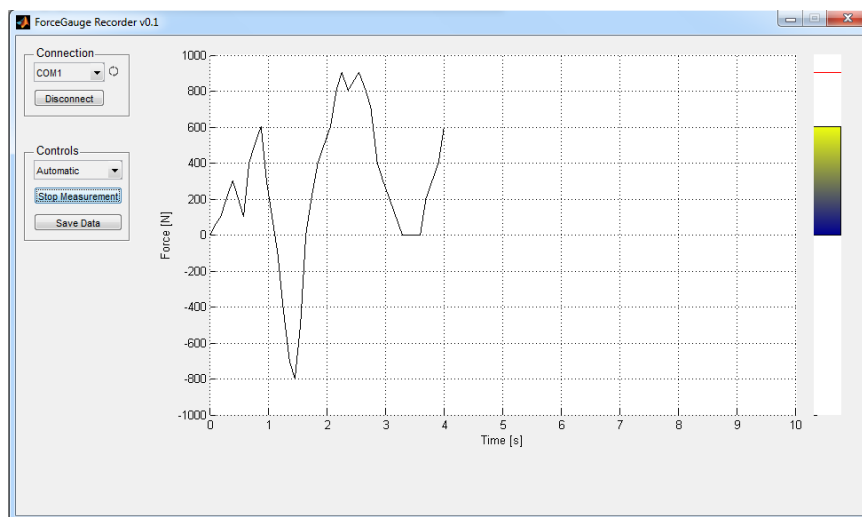
Een eigenschap die Matlab uniek maakt is dat het een programmeertaal is waar heel gemakkelijk met matrices en vectoren kan worden gewerkt. De naam Matlab staat dan ook voor *Matrix Laboratory*. Wat matrices en vectoren zijn, is nu nog niet van belang. Daar komen we later vanzelf bij uit.

Waarom is Matlab handig voor een Bewegingstechnoloog? Matlab is voor een Bewegingstechnoloog ontzettend handig omdat:

- data uit metingen kan worden verwerkt
- data uit metingen kan worden geanalyseerd
- data uit metingen kan worden gevisualiseerd
- er direct gecommuniceerd kan worden met externe apparaten
- er gemakkelijk digitale modellen mee kunnen worden gemaakt
- er gemakkelijk wiskundige bewerkingen, die voor de Bewegingstechnoloog van belang zijn, mee gedaan kunnen worden
- er software interfaces mee kunnen worden gemaakt (zie Figuur 1)

In veel projecten is op één of andere manier gebruik gemaakt van Matlab. Een kleine greep uit de vele Bewegingstechnologie projecten:

Alumnus Erik van de Kerkhof heeft Matlab gebruikt om een krachtmeter uit te lezen en deze signalen verder te verwerken. Deze verwerkte signalen heeft hij gesynchroniseerd met EMG-data (data die spieractiviteit representeert, zie Figuur 1). Dit met als doel de relatie te bepalen tussen de beenspreiding en de kracht die het kost om tot een handstand te komen.



Figuur 1: Voorbeeld van de grafische gebruiker interface gemaakt door Erik van de Kerkhof in Matlab.

Twee andere alumni Bewegingstechnologie hebben in Matlab een grafische interface (Engels: graphical user interface, GUI) gemaakt om een Arduino uit te lezen (Wat een Arduino is, leer je later in je opleiding) die is aangesloten op een aantal box-sensoren (zie Figuur 2). Dat zijn sensoren waar

karateka's en boxers tegen aan kunnen slaan en trappen. Deze opstelling wordt specifiek gebruikt om karateka's bepaalde oefeningen te laten doen en om hun progressie te meten.



Figuur 2: twee alumni hebben m.b.v. Matlab een GUI gemaakt om de kracht van meerdere sensoren op een boxzak uit te lezen en om verschillende oefeningen op te leggen aan de gebruiker.

Andere studenten hebben video-data afkomstig uit Kinovea ingelezen in Matlab en vergeleken met data uit de Microsoft Kinect om vast te stellen in welke mate de videodata overeenkomstig waren met de Kinect data.

Nu is hopelijk een klein beetje duidelijk geworden waarom Matlab een handig en belangrijk softwarepakket is voor een Bewegingstechnoloog. Er zullen nog veel vragen zijn. Die zullen gedurende deze cursus worden beantwoord. Als je een vraag hebt, stel deze dan aan de begeleidende docent en wacht er niet te lang mee.

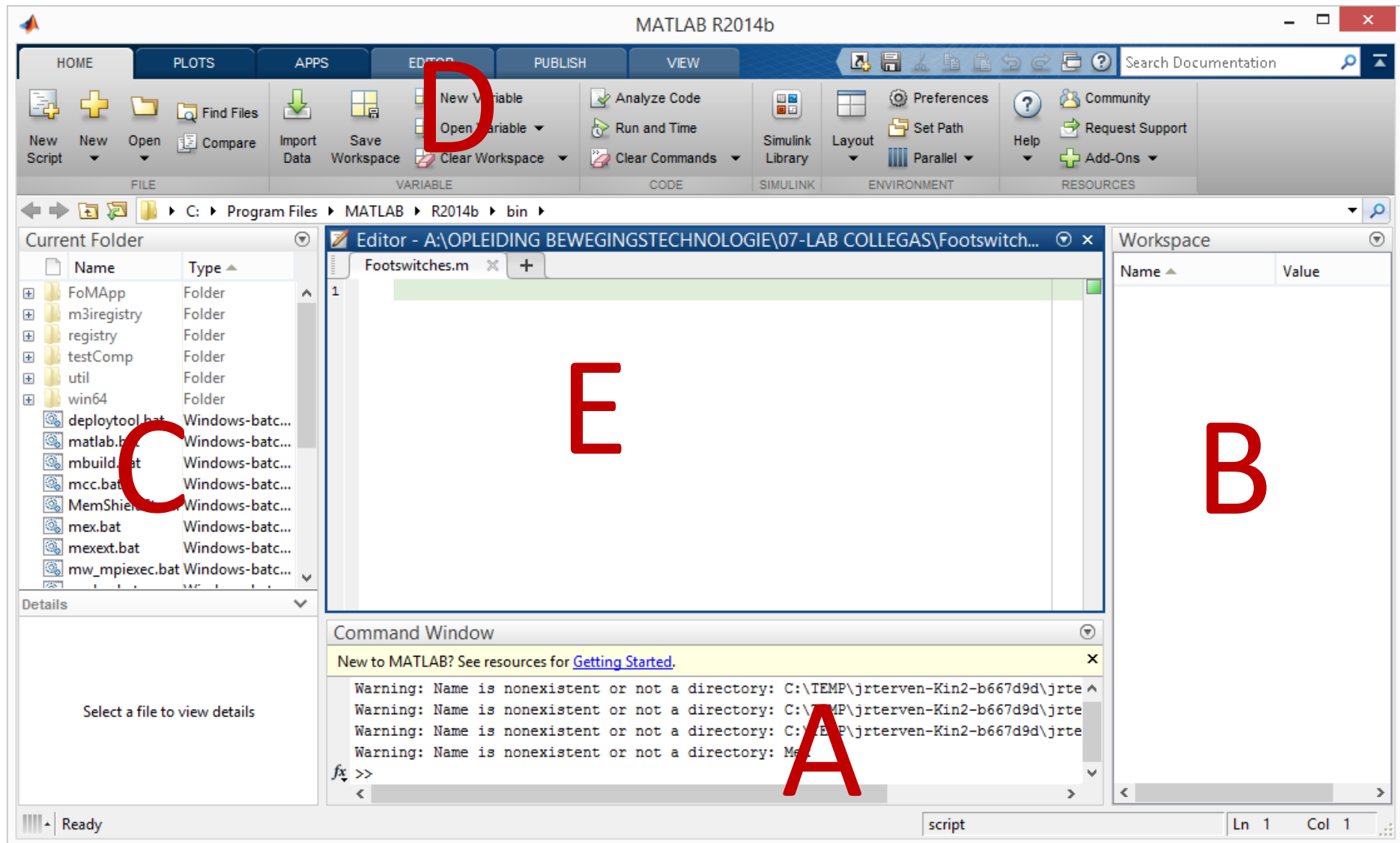
3 De Matlab omgeving

Zoals eerder aangegeven komt Matlab met een programmeeromgeving. Dat is heel handig want dan heeft de Matlab gebruiker na het installeren van Matlab alles-in-één. Er zijn veel verschillende versies van Matlab in de omloop. Op de Haagse Hogeschool wordt op dit moment versie 2017a gebruikt.

Als je in dikgedrukte letters iets ziet in deze reader dan moet **JJJ een actie uitvoeren:**

Start nu Matlab op

Na het opstarten van Matlab krijgt de gebruiker de afbeelding te zien zoals in Figuur 3. Binnen dit figuur zijn verschillende elementen aangegeven.



Figuur 3: Zo ziet Matlab er uit als het programma net is opgestart. A: het Command Window (uitleg zie tekst), B: de Workspace, een plek waar de variabelen in het geheugen worden weergegeven, C: de huidige folder waar Matlab uit werkt, D: menubalk, E: een script, merk op dat er verschillende tabbladen zijn.

Kijk eens naar Figuur 3.

Daarin is aangegeven wat je ziet. Het is belangrijk even de tijd te nemen en te realiseren wat welke onderdelen van de Matlab omgeving doen.

3.1 Command Window (A)

het Command Window (onderdeel A) ga je nog vaak gebruiken. Als je in Matlab een bepaalde operatie uitvoert dan wordt het resultaat van deze handeling weergegeven in het Command Window.

Voer in het Command Window de volgende tekst in: 4+4.

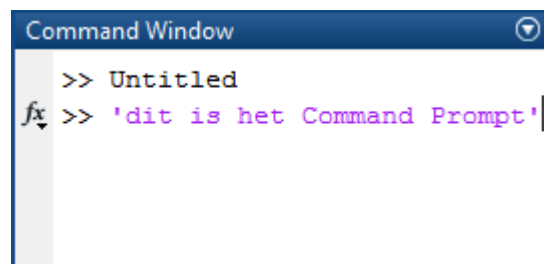
Druk op enter

Wat zie je?

Je ziet dat Matlab gebruikt kan worden als een veredelde rekenmachine.

3.1.1 Command prompt

In het Command Window vind je de Command Prompt. De Command Prompt (CP) is een regel in het Command Window waarin je een bepaalde instructie kunt uitvoeren (zie Figuur 4).



Figuur 4: In het Command Window zit de 'command prompt'.

Een opdracht op de CP voer je uit door op **Enter** te drukken.

3.2 Workspace (B)

Na het uitvoeren van bovenstaande handeling, zie je dat de Workspace (B) is veranderd. Wat is de Workspace? De Workspace is een onderdeel van de Matlab omgeving die laat zien welke variabelen in het geheugen zijn ingeladen. Wat een variabele is leer je in hoofdstuk 5 Variabelen.

De Workspace is handig, want de gebruiker kan tijdens het uitvoeren van een programma in de gaten houden welke variabelen in het geheugen staan. Ook kan de gebruiker controleren of de waardes van een bepaalde berekening correct zijn.

Lees de waarde van de variabele *ans* in de Workspace af. Klopt deze waarde met de berekening hierboven?

3.3 Current Folder (C)

De Current Folder laat, zoals de naam al aangeeft, zien in welke folder de Matlab omgeving aan het werken is. In de toekomst zal je bepaalde tekstbestanden moeten gaan inlezen. Als je dat doet, dan is het belangrijk dat de Current Folder wijst naar de folder waar de tekstbestanden in staan. Als de Current Folder niets wijst naar de tekstbestanden kan Matlab er geen gebruik van maken.

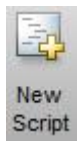
3.4 Toolstrip (D)

De toolstrip behoeft weinig uitleg. Hij bevat allerlei buttons waarachter een bepaalde functionaliteit schuil gaat. In de praktijk zal een Bewegingstechnoloog niet heel veel verschillende knoppen gaan gebruiken.

Klik eens op de verschillende tabbladen om te bekijken welke knoppen er allemaal zijn.

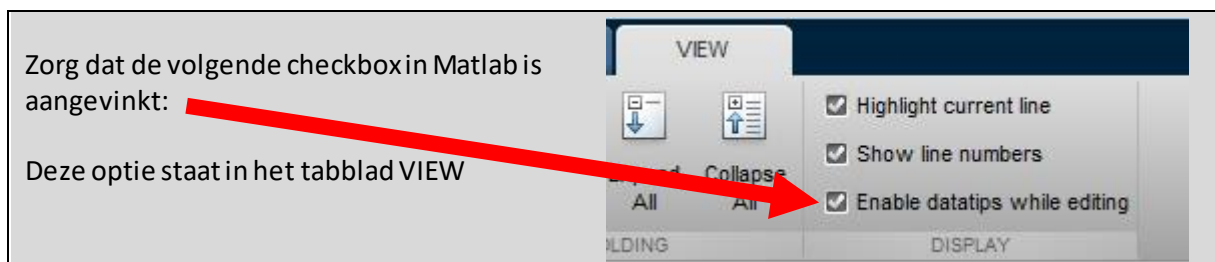
3.5 Script Editor (E)

De Script Editor is het gedeelte waarin je in de nabije toekomst scripts gaat bouwen. Om de script-editor te zien, moet er een script geopend zijn. Een script is een verzameling aan code regels die bij elkaar horen en in een specifiek bestand staan: *een script-file*. Een Matlab script heeft de extensie *.m*. Elke regel code in een script kan worden uitgevoerd in het Command Window (A). Je kunt een script dus zien als een stel gecombineerde operaties.

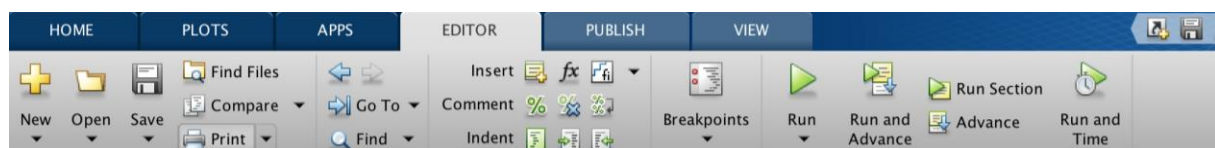


Druk op de button New script (in tabblad EDITOR).

Nu is er in de Script Editor (een nieuw script aangemaakt dat waarschijnlijk de naam *Untitled* heeft).



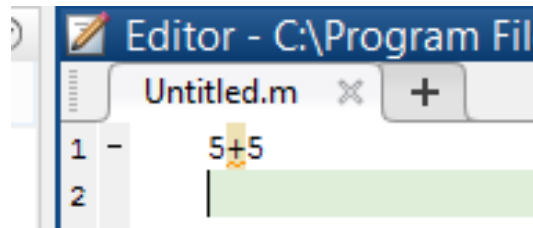
Druk op save (in tabblad EDITOR) en geef het script een handige en snel herkenbare naam.



1. LET OP! Je mag geen spaties in Matlab scriptnamen gebruiken!
2. Ook mag een scriptnaam niet beginnen met een numerieke waarde (een getal).
3. Zorg dat je een script in een folder plaatst met een foldernaam waaruit je kunt afleiden wat er in staat. Doe je dit niet, dan pluk je daar later de 'rotte' vruchten van. Gebruik bijvoorbeeld de volgende folder hiërarchie en scriptnaam:
 - a. Biostatica – Matlab \ Week1_1 \ Scripts \ *hetEersteScript.m*

Typ in de eerste regel van het script: 5+5

Je hebt, als het goed is, de volgende situatie (zie Figuur 5):



Figuur 5: de operatie vijf plus vijf in een script binnen de Script Editor.

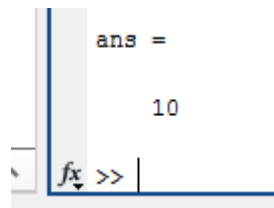
Druk op save (of maak gebruik van de snelkoppeling CTRL+s)



Druk nu op Run (of maak gebruik van de snelkoppeling F5 of COMMAND+ALT+R op MAC)

Wat is er nu veranderd in het Command Window?

Als het goed is, zie je dit staan in het Command Window:

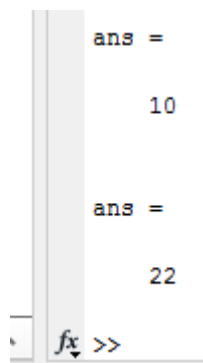


Figuur 6: resultaat in het Command Window na het uitvoeren van een script.

Een berekening zoals: 5+5 noemen we in Matlab (of in een andere programmeertaal) een *operatie*.

Vul het script aan met: 33-11

Sla het script op en voer het opnieuw uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).



Figuur 7: resultaat in het Command Window na het uitvoeren van het aangepaste script.

Het valt nu op dat er twee operaties zijn uitgevoerd. Namelijk 5+5 en 33-11. Daarom zijn er twee antwoorden te zien. Een script is dus een verzameling van operaties die achter elkaar worden uitgevoerd. Deze handelingen (operaties) kun je ook één voor één in het Command Window uitvoeren. Maar dat wordt een onoverzichtelijke bende.

3.6 Samenhang tussen een script en het Command Window

De samenhang tussen een script en het Command Window is dat het *resultaat* van een script in het Command Window wordt getoond. We zeggen dat de output van het script in het Command Window wordt getoond. Je kunt in het Command Window ook zelf een operatie uitvoeren zonder een script. Dit kan handig zijn om een regel code te testen en heb je hiervoor al gedaan.

Als een script heel veel regels code krijgt, dan wordt de output in het Command Window een grote puinhoop. Elke regel code heeft een resultaat die wordt weergegeven in het Command Window. Dat noemen we een 'echo'. Daarom is het verstandig om de echo's van een script te beperken. Dit doen we door aan het einde van een regel in een script een ; te plaatsen.

Om de output van een script te beperken plaatsen we aan het einde van een regel een ;

Plaats aan het einde van elke regel in jouw script een puntkomma.

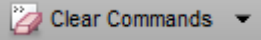
Voer het script opnieuw uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Wat valt op?

Als het goed is, wordt het resultaat van de operaties 5+5 en 33-11 nu niet meer weergegeven in het Command Window. Je kunt dus met een puntkomma voorkomen dat de output van een regel code wordt weergegeven in het Command Window.

Als je het Command Window zou willen opschonen dan kun je dit commando gebruiken **clc**. Dit commando staat voor: Clear Command Window.

3.7 Vragen

1. In de Workspace wordt de output van een operatie weergegeven. (Ja/Nee)
2. Welke 3 knoppen heb je in de bovenstaande opdracht gebruikt (niet spieken)?
3. Wat is de sneltoets(combinatie) voor het runnen van een script?
4. Is het toegestaan spatie te plaatsen in een Matlab script naam?
5. Wat is het nut van de Current Folder window in de Matlab omgeving?
6. In welk tabblad staat de knop: New Script?
7. Wat doet deze knop: ? **Test dit!**
8. In welk tabblad staat de knop: Save?
9. Met welk commando kun je het Command Window schoonmaken/leegmaken.
10. Wat is het verschil tussen het uitvoeren van een operatie in een script en in het Command Window?
11. Hoeveel operaties kun je tegelijkertijd in het Command Window uitvoeren?
12. Mag een Matlab script beginnen met een getal (BV. 1Ditiseenscript.m)?
13. Wat voor extensie heeft een Matlab script?
14. Type in het script file $X = 4 + 6$: en $Y = 12$: en run het script. Wat gebeurt er? Hoe los je dit op?
15. Hoe wordt een berekening in Matlab ook wel genoemd?
16. Wat is het nut van de puntkomma aan het einde van een regel?

3.8 Antwoorden

1. Nee, in de Workspace wordt weergegeven welke variabelen in het geheugen staan. De output van een operatie wordt weergegeven in het Command Window
2. De knoppen, New Script, Save en Run.
3. F5 of COMMAND+ALT+R op een Mac.
4. Nee!
5. Laten zien vanuit welke folder Matlab runt. Dit is bijvoorbeeld belangrijk om te weten als een bepaald bestand wordt ingelezen.
6. Tabblad Home
7. Deze knop maakt het Command Window leeg.
8. Tabblad Editor
9. clc
10. Er is geen verschil! De operatie(s) wordt exact hetzelfde uitgevoerd in het Command Window als in het script. *Wat is dan het verschil?* Een script is een verzameling van operaties die achtereenvolgens worden uitgevoerd als je het script uitvoert.
11. Meer dan één. Is dat verwarrend? Ja, wel een beetje. Hoe kun je meerdere operaties in het Command Window uitvoeren? Door een operatie af te sluiten met een puntkomma en direct rechts daarvan de nieuwe operatie te plaatsen. Bijvoorbeeld: **Typ in het Command Window:** 4+4; 2+2; **Druk op enter.** Mocht het nog niet duidelijk zijn wat het verschil is tussen het Command Window en het uitvoeren van code in script, wees dan gerust. Dat wordt later vanzelf duidelijk.
12. Nee dat mag niet.
13. Een extensie is iets waaronder je het opslaat, zoals bij een Word bestand een .docx en Excel een .xlsx bij Matlab is het een .m extensie
14. De regels code bevatten beide een dubbele punt i.p.v. een puntkomma, Matlab laat zien in welke regel een error is, waarin staat dat de code niet compleet is of onjuist. Dit is op te lossen door de dubbele punt te vervangen door de puntkomma.
15. Een operatie. Echter de term 'operatie' is algemene term die een bepaalde handeling beschrijft. Die handeling kan ook een berekening zijn, maar ook iets anders.
16. De output of echo van een regel code wordt niet afgedrukt in het Command Window.

4 Matlab operaties

Tot nu toe heb je slechts een paar simpele operaties¹ gebruikt. Er zijn echter nog veel meer mogelijkheden met Matlab. In dit gedeelte van de reader gaan we Matlab verder verkennen als veredelde rekenmachine. We gaan de volgende operaties gebruiken: delen, vermenigvuldigen, machtsverheffen en worteltrekken.

Niet alle van deze genoemde operaties worden daadwerkelijk geïmplementeerd met *operators*. Sommige operaties worden geïmplementeerd met een *functie*. Wat dat is leer je later. In dit gedeelte gaan we verschillende mathematische operaties leren kennen en toepassen.

Maar eerst het volgende. Wat is een operator?

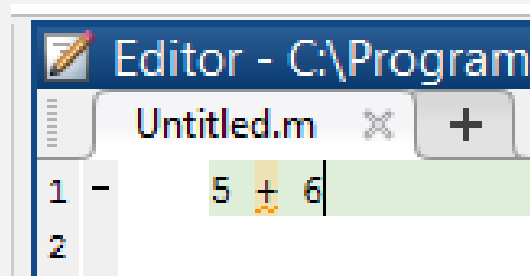
Een operator is een programmeertaalconstructie die werkt op twee operands. Ok... Wat is een operand? Dat laat zich het beste uitleggen met een voorbeeld:

In het vorige gedeelte hebben we 5 opgeteld bij 5. Nu gaan we 5 bij 6 optellen (zie Figuur 8). De eerste linker *operand* is de waarde 5.

De operator is hier de *plus operator*.

De plus operator werkt op de linker *operand* en de rechter *operand*.

De rechter operand is de waarde 6.



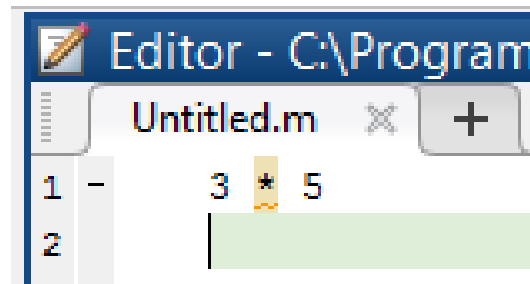
Figuur 8: de operatie vijf plus zes. Het gaat in deze afbeelding om de terminologie die wordt gebruikt in een programmeertaal. De linkeroperand (de waarde 5) en de rechteroperand (de waarde 6) worden door de plus-operator (+) gecombineerd tot een resultaat.

De operands in dit voorbeeld en de vorige voorbeelden zijn dus waarden waar de operator op werkt.

¹ Handelingen in de vorm van berekeningen.

4.1 Vermenigvuldigen

Vermenigvuldigen gaat in Matlab met de vermenigvuldig operator (het sterretje): $*$ (zie Figuur 9).



Figuur 9: Voorbeeld van vermenigvuldigen in Matlab

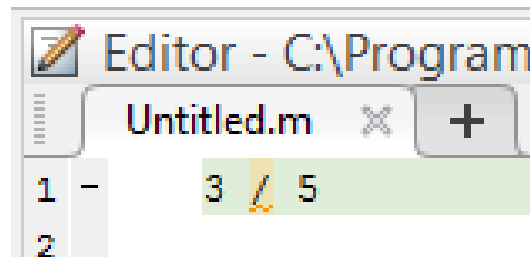
Pas het script aan zodat het er hetzelfde uitziet als Figuur 9.

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Test zelf een aantal verschillende waarden door de rechtoverand en de linkeropand aan te passen.

4.2 Delen

Delen gaat in Matlab met de deel operator: $/$ (zie Figuur 10).



Figuur 10: Voorbeeld van vermenigvuldigen in Matlab

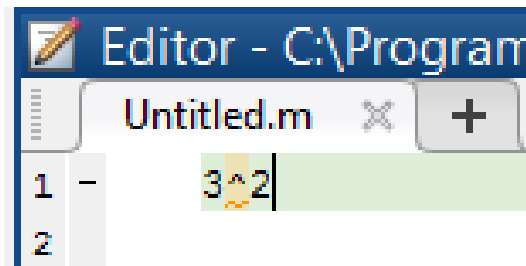
Pas het script aan zodat het er hetzelfde uitziet als Figuur 10.

Voer het script uit.

Test zelf een aantal verschillende waarden door de rechtoverand en de linkeropand aan te passen.

4.3 Machtsverheffen

Tot nu toe hebben we getallen opgeteld, afgetrokken, vermenigvuldigd en gedeeld. Maar er zijn nog meer operaties. Je kunt getallen ook kwadrateren. Dat wordt gedaan met de operator $^$ (dakje) en ziet er als volgt uit:



Figuur 11: Voorbeeld van het kwadrateren van

Pas het script aan zodat het er hetzelfde uitziet als Figuur 11.

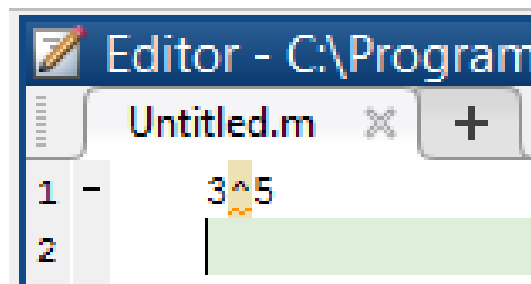
Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Zoals je ongetwijfeld weet is het resultaat van $3^2 = 9$.

1. Hoeveel verschillende operatoren ken je nu? Voor het antwoord zie de footnote: ²
2. Heb je een idee hoe je een operand tot een hogere macht kunt heffen? Dus 3^3 of 3^4 dus 3^x .

4.3.1 Hogere machten

Als je de voorgaande stappen hebt bestudeerd en uitgevoerd dan heb je vast een idee hoe je hogere machten kunt uitvoeren in Matlab. Juist! Door de rechteroperand van de machteroperator te verhogen (zie Figuur 12)



Figuur 12: Hogere macht.

Pas het script aan zodat het er hetzelfde uitziet als Figuur 12

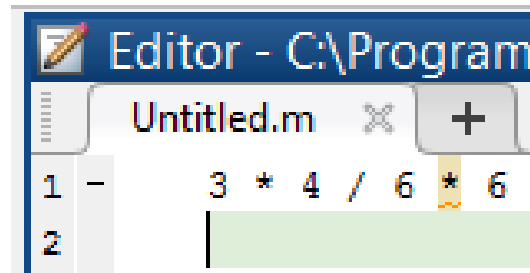
Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Test zelf een aantal verschillende waarden door de macht (rechtoperand) en het grondgetal (linkeroperand) aan te passen.

² Je kent als het goed is vier verschillende operatoren: de plus-operator, de min-operator, de macht-operator en de puntkomma. De puntkomma kun je ook opvatten als een operator omdat die zorgt dat de output van de operatie niet wordt weergegeven. De output van een operatie kun je ook opvatten als ene operand.

4.4 Haakjes

Zoals je vast wel eens hebt geleerd in je vorige opleiding, moet je soms bij berekeningen haakjes gebruiken. Dat is om prioriteit te geven aan bepaalde operaties. Dat wil zeggen dat de operaties tussen haakjes als eerste worden uitgevoerd. Zie Figuur 13.



Figuur 13: Wat is de uitkomst van deze berekening?

Wat denk jij dat uit de berekening komt in Figuur 13?

Pas het script aan zodat het er hetzelfde uitziet als Figuur 13.

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Waar moet je haakjes zetten om als antwoord: 0.333 te krijgen?

Probeer de voorgaande vraag zelf te beantwoorden met Matlab. Kortom, zelf proberen want Matlab geeft automatisch een antwoord.

4.4.1 Een specifiek voorbeeld

Als je geen haakjes gebruikt, dan weet je ook niet wat je exact aan het berekenen bent. Kijk maar eens naar dit voorbeeld: $6 / 3 * 5$. Wat is het resultaat van deze bewerking zonder haakjes? Natuurlijk in Matlab kun je dit gemakkelijk uittesten. Maar als de waardes in zogenaamde variabelen zijn ingepakt dan wordt het minder gemakkelijk te zien. Immers je ziet niet snel welke waardes in de variabelen zijn opgeslagen.

Wat verwacht je dat uit $6 / 3 * 5$ komt?

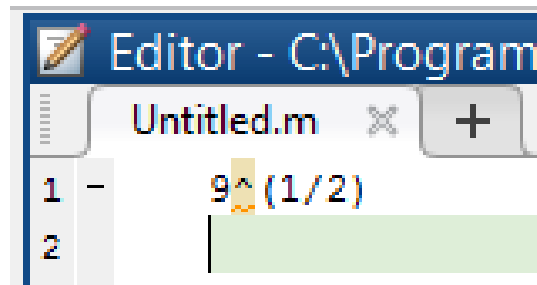
Voer de expressie: $6 / 3 * 5$ in Matlab en voer deze uit.

Je kunt het resultaat van deze expressie zelf voorspellen als je weet welke prioriteit een bepaalde operator heeft. Als je meer informatie over de prioriteiten van sommige operatoren wil verkrijgen, ga dan naar: https://nl.mathworks.com/help/matlab/matlab_prog/operator-precedence.html.

4.5 Worteltrekken

Als we getallen kunnen optellen, dan moeten we ook getallen kunnen aftrekken. Als we getallen kunnen vermenigvuldigen dan moeten we ze ook kunnen delen. Als we getallen kunnen kwadrateren dan moeten we ze ook kunnen..... Juist! Worteltrekken.

In Matlab zit geen wortel operator. Althans niet in de vorm waardoor je met een specifieke wortel operator in één keer de wortel kunt trekken. Je kunt natuurlijk wel de macht operator gebruiken (zie Figuur 14).

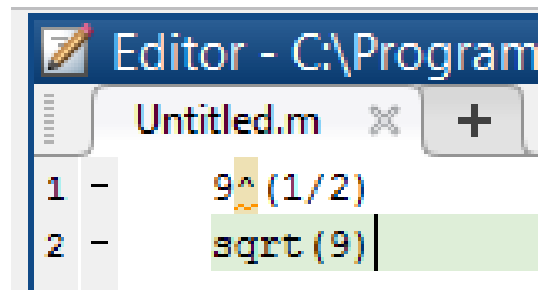


Figuur 14: Wat is de uitkomst van deze berekening?

Pas het script aan zodat het er hetzelfde uitziet als Figuur 14.

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Je weet dat $3^3=9$. Dus dan is $9^{1/2}=3$. We zouden het liefst een operator gebruiken voor de wortel, maar zoals gezegd die is er niet. Daarom heeft Matlab een iets anders bedacht zie Figuur 15:



Figuur 15: Wat is de uitkomst van deze berekening?

Matlab heeft een functie gemaakt genaamd `sqrt()`. Wat een functie is, wordt je later duidelijk gemaakt.

Pas het script aan zodat het er hetzelfde uitziet als Figuur 15

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Het valt op dat de antwoorden van regel 1 en van regel 2 exact hetzelfde zijn.

Verander in beide regels de grondgetallen en controleer of de resultaten van regel 1 en regel 2 telkens hetzelfde zijn.

4.6 Vragen en opdrachten

1. Hoeveel is 8^8 ? Gebruik Matlab.
2. Wat is een operator?
3. Wat is het antwoord op de berekening: $3 + 4 - 5 / 6 * 7 ^ 8$?
4. Waar of niet waar: in Matlab is het gebruik van haakjes in principe overbodig.
5. Wat is het antwoord op de berekening: $((3 ^ 4) - ((5 / 2) * (2 ^ 8)))$?
6. Bestaat er in Matlab een wortel operator?
7. Bereken de wortel van 121 en 144 en 119025.
8. Welke waarde heeft de rechter operand in de volgende regel code: $5 / 6$?
9. Welke waarde heeft de rechter operand in de volgende regel code: $2 ^ 8$?
10. Welke waarde heeft de linker operand in de volgende regel code: $4 - 2$?
11. Wat wordt er onder een operand verstaan?
12. Waar of niet waar? In de volgende code regel heeft de min-operator voorrang op de macht-operator: $76 - 6^2$.
13. Voer in Matlab de volgende som in: $((9*8)/(2*38))$; . Wat gebeurt er? Hoe los je dit op?

4.7 Antwoorden

1. Vul het in: 16777216
2. Een operator is een bewerking op twee operands. Een voorbeeld is de min-operator. Deze zorgt dat de waarde van de rechter operand wordt afgetrokken van de linker operand.
3. -4.8040×10^6
4. Niet waar! Haakjes heb je met regelmaat nodig.
5. -559
6. Nee, die bestaat niet. Je hebt een functie om de wortel te berekenen (`sqrt()`). Ook kun je de macht-operator gebruiken om de wortel te berekenen (zie voorbeelden).
7. Dit is een Matlab exercitie en moet je dus met Matlab uitvoeren.
8. De rechter operand is de waarde 6.
9. De rechter operand is de waarde 8.
10. De linker operand is de waarde 4.
11. Een operand is het 'ding' links of rechts van de operator. Hetgeen waar de operator 'op werkt'.
12. Dat kun je natuurlijk snel zelf testen door het in te voeren in Matlab. Dat is het gemakkelijkste aan deze cursus: 'Matlab geeft je bij veel vragen een direct antwoord'. Het antwoord is niet waar. De \wedge operator heeft een hogere prioriteit en wordt dus eerder uitgevoerd. Als \wedge geen hogere prioriteit zou hebben dan zou het antwoord zijn: $76 - 6^2 = 70^2 = 4900$.
13. Je krijgt een foutmelding. Dat is een situatie waar je vertrouwd mee moet raken. Matlab geeft namelijk, zoals je zult gaan merken, snel foutmeldingen. Het is jouw taak als programmeur om de foutmelding te lezen en te interpreteren en daaruit af te leiden wat er moet gebeuren om de fout op te lossen. In het begin kan dat frusterend zijn, helaas.

In dit geval is de foutmelding: "Unbalanced or unexpected parenthesis or bracket". Matlab vertelt dat de som niet 'in evenwicht' is. Dit komt omdat er een ongelijk aantal haakjes is. Het aantal haakjes openen (en haakjes sluiten) moet exact hetzelfde zijn!

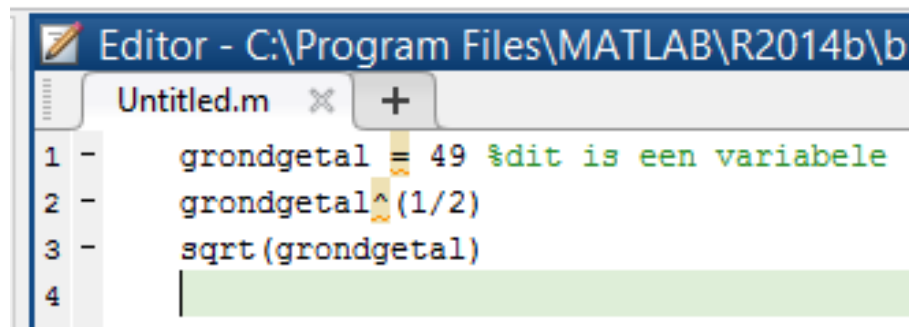
5 Variabelen

Tot nu toe hebben we verschillende rekenkundige operaties direct ingevoerd in het Command Window (Command Window) en in een script. Je kunt Matlab dus gebruiken als een rekenmachine. Matlab is in essentie een waanzinnig complexe rekenmachine. Complex? Ja complex! We hebben in de vorige gedeeltes slechts het tipje van de speekwoordelijk sluier opgelicht.

In het laatste voorbeeld in Figuur 15 moesten we telkens twee grondgetallen (in het geval van Figuur 15 het getal 9) aanpassen. Dit kan een stukje slimmer door de grondgetallen te onthouden (op te slaan). Het opslaan van 'iets' doen we in Matlab met behulp van een variabele.

5.1 Aanmaken van een variabele

Het aanmaken van een variabele wordt geïllustreerd aan de hand van het voorbeeld in Figuur 16.



Figuur 16: Het gebruik van een variabele.

Je ziet dat in regel 1 van het script deze regel staat: `grondgetal = 49`. In deze regel is de waarde 49 opgeslagen in een variabele met de naam `grondgetal`. Overall in het script waar nu de naam `grondgetal` voorkomt, zal Matlab de naam veranderen naar de waarde 49.

In regel 2 en 3 zie je de variabele met de naam `grondgetal` staan. Wat gaat het script nu doen?

Pas het script aan zodat het er hetzelfde uitziet als Figuur 16

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

Geef de variabele `grondgetal` een aantal andere waardes en test of regel 2 en regel 3 telkens hetzelfde resultaat produceren.

Een variabele maakt het leven van de Bewegingstechnoloog gemakkelijker. Want nu hoeft hij slechts in één regel de waarde aan te passen. In elke regel daarna waar de variabele in voorkomt, zal automatisch de juiste waarde worden gebruikt. Dit is enigszins vergelijkbaar met Excel, waarin veranderingen in de ene cel automatisch worden doorgevoerd in andere cellen. Dit bespaart op het schrijven van code, vooral als programma's heel veel langer gaan worden.

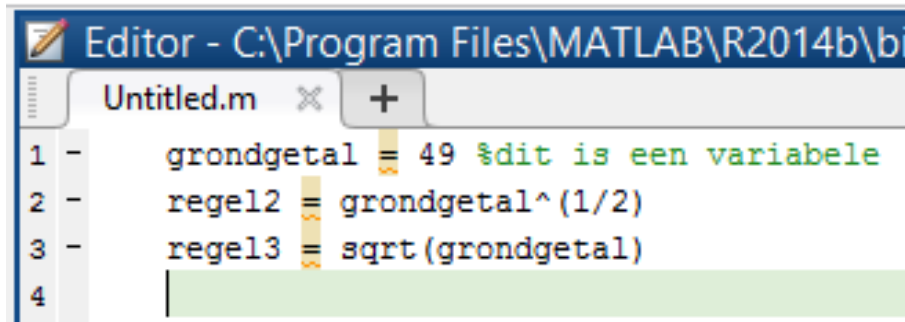
1. Wat is het voordeel van het gebruik maken van variabelen?
2. Kan het resultaat van regel 2 en regel 3 in Figuur 16 ook in afzonderlijke variabelen worden opgeslagen?³

³ Ja!

Pas het script dat je tot nu toe hebt gemaakt aan zodat het resultaat van de operatie in regel 2 wordt opgeslagen in een variabele.

Doe hetzelfde voor regel 3

Een voorbeeld van hoe je de vorige twee opdrachten had kunnen doen, is te zien in Figuur 17.



Figuur 17: Het gebruiken van 3 variabelen binnen één script. Herken je de drie variabelen?.

Voer het script uit (Druk op Run of F5 of op COMMAND-ALT-R op een Mac).

```
grondgetal =
    49

regel2 =
     7

regel3 =
     7




fx >>
```

Figuur 18: Het resultaat van Figuur 17.

5.2 Workspace

Dit is een mooi moment om eens terug te grijpen naar de Workspace. Wat deed de Workspace ook alweer? Als je het niet meer weet, ga dan eens terug naar paragraaf Workspace (B).

Kijk eens hoe de Workspace er uit ziet en wat er in staat:

Workspace	
Name ▲	Value
 grondgetal	49
 regel2	7
 regel3	7

Figuur 19: De Workspace van de Matlab omgeving na het uitvoeren van de code in Figuur 17.

Vul het programma in Figuur 17 aan met een aantal door jou aangemaakte variabelen en controleer de Workspace.



Maak een nieuw script aan, druk op .

Bereken met behulp van de informatie in de vorige paragrafen de volgende kwadratische vergelijking: $y = ax^2 + bx + c$. Zorg dat de a , b , c , x en y variabelen zijn.

Geef de variabelen a , b , c en x de volgende waarden: 1, 1, 1, 2.

Zorg dat het resultaat van de vergelijking in een variabele wordt opgeslagen met de naam y .

Voer het script uit.



Het resultaat van de vorige acties staat op de volgende pagina. Het enige dat je nu hoeft te doen om een kwadratische vergelijking te berekenen, is het script te openen en de waardes die je toekent aan de variabelen aan te passen. Het is natuurlijk zo dat als je geen gebruik maakt van variabelen, dus de waardes direct invult, het script een heel stuk korter is. Maar is het ook beter te lezen? Is het script ook beter te begrijpen? Het antwoord is natuurlijk: nee!

Doorgaans is code opgebouwd met variabelen veel beter leesbaar dan code waarin dat niet is gebeurd. Vooral als het script heel lang wordt.

TIP: maak een variabele nooit te kort, gebruik bij voorkeur hele woorden!

Een ander voordeel is dat als je de code uitbreidt met bijvoorbeeld een derdegraads vergelijking je gebruik kunt maken van de eerder aangemaakte variabelen (de coëfficiënten) a , b en c .

Variabelen zijn HOOFDLETTER gevoelig. De variabele A is dus een andere variabele dan a

Workspace	
Name ▲	Value
 a	2
 A	1

5.3 Tekst (strings)

Tot zover hebben we telkens *numerieke waardes* (getallen) opgeslagen in variabelen. Maar dat is niet het enige wat we in variabelen kunnen opslaan. We kunnen ook stukken tekst opslaan in een variabele. Zo'n stukje tekst wordt een 'string' genoemd. Elk karakter in een string wordt een karakter (char) genoemd. Een string bestaat uit een aantal karakters aan elkaar geregen karakters (Eng: *stringing*).

Maar, hoe herkent Matlab een stuk tekst? Niet door het onderstaande voorbeeld:

```
>> a = tekst
Undefined function or variable 'tekst'.
```

Figuur 20: een mislukte poging om een tekst op te slaan in de variabele *a*.

Je ziet dat Matlab in het rood een foutmelding geeft. Maar wat zegt de fout melding?

De foutmelding geeft aan dat er in het geheugen (zie de Workspace) geen variabele bestaat met de naam tekst. Wat hier staat is dat een variabele met de naam *tekst* wordt toegekend aan de variabele met de naam *a*. Echter, de variabele tekst bestaat nog niet. Dat wordt in de Matlab foutmelding aangegeven met de tekst *undefined function or variable*.

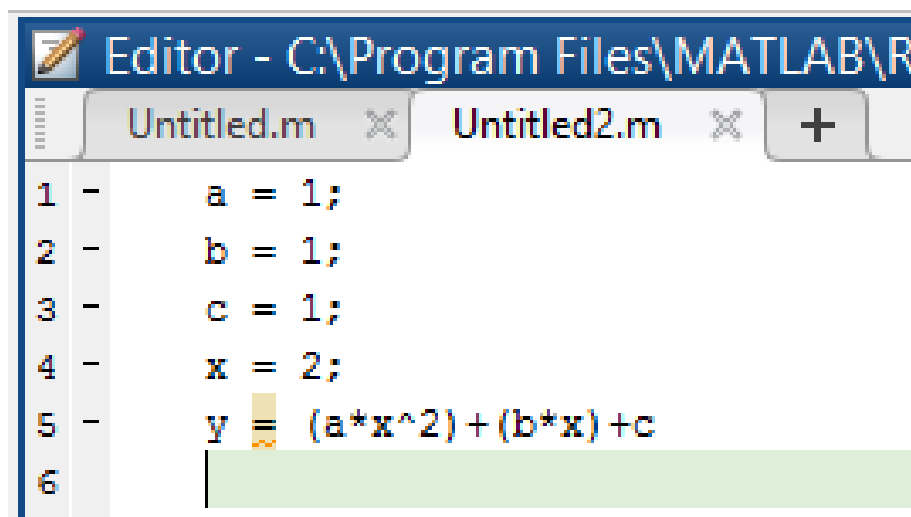
De juiste manier om een stuk tekst in Matlab op te slaan, is door het gebruikmaken van enkele quotes. Zie het volgende voorbeeld:

```
>> a = 'tekst'

a =

tekst
```

Figuur 21: De juiste manier om een stuk tekst op te slaan in een variabele



Figuur 22: De kwadratische vergelijking van de acties op de vorige pagina..

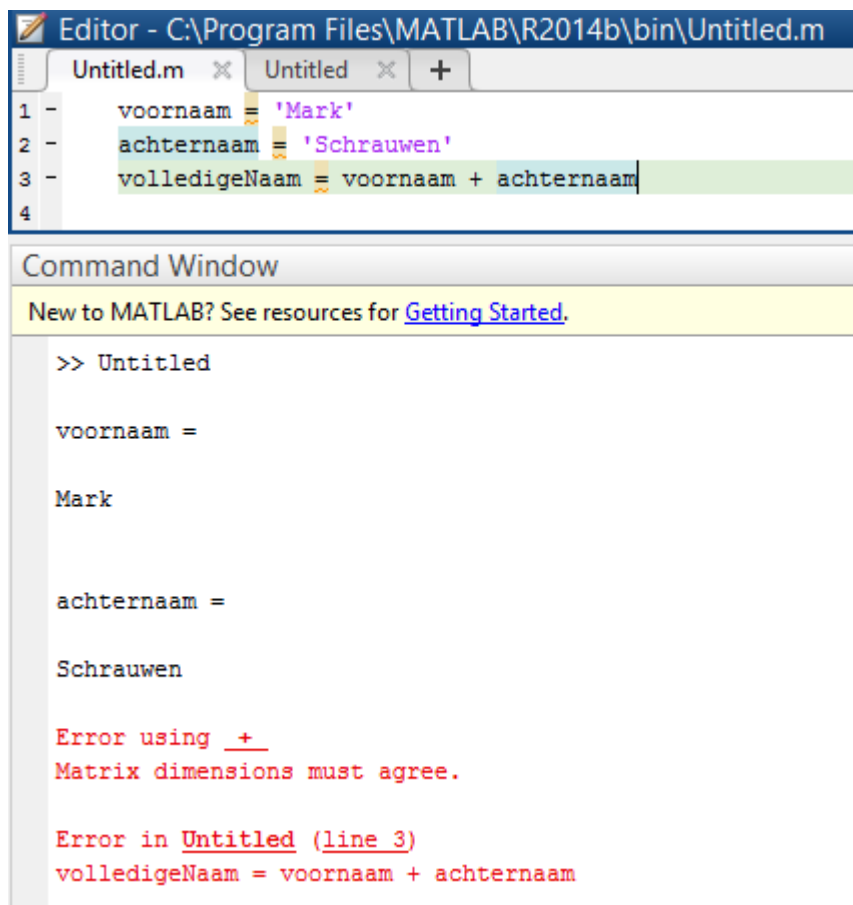
```
>> Untitled2  
  
y =  
  
7  
  
fx >> |
```

Figuur 23: Het resultaat van de kwadratische vergelijking.

5.3.1 Aan elkaar koppelen van strings

Soms wil je twee strings (tekst) in Matlab aan elkaar koppelen. Een voorbeeld zou kunnen zijn dat je een voornaam en achternaam hebt opgeslagen in een variabele en je wilt één naam maken van de voor- en de achternaam.

Niet gehinderd door enige voorkennis zou je het volgende kunnen doen (zie Figuur 24):



```
Editor - C:\Program Files\MATLAB\R2014b\bin\Untitled.m  
Untitled.m x Untitled x +  
1 - voornaam = 'Mark'  
2 - achternaam = 'Schrauwen'  
3 - volledigeNaam = voornaam + achternaam  
4  
  
Command Window  
New to MATLAB? See resources for Getting Started.  
  
>> Untitled  
  
voornaam =  
  
Mark  
  
achternaam =  
  
Schrauwen  
  
Error using +  
Matrix dimensions must agree.  
  
Error in Untitled (line 3)  
volledigeNaam = voornaam + achternaam
```

Figuur 24: het aan het elkaar koppelen van strings. Helaas gaat het in dit voorbeeld fout.

Je ziet de foutmelding in Figuur 23. Daar staat dat de variabele *voornaam* en de variabele *achternaam* verschillende *dimensions* (afmetingen) hebben. Dat klopt! Mark is vier letters en Schrauwen is negen letters. Hoe lossen we dit probleem van het aan elkaar koppelen van strings op?

In Matlab heb je de `[en]` operator. Dit is een operator bestaande uit twee delen. De operand moet in dit geval *tussen* de rechte haken komen te staan. In Matlab wordt deze operator gebruikt om een vector te maken. Wat een vector is en hoe je een vector kunt gebruiken leer je na deze paragraaf. Voor nu volstaat het om te zeggen dat een vector een variabele is waar meerdere variabelen in zijn opgeslagen. Vergelijkbaar met een tabel: in 1 tabel zijn meerdere cellen opgeslagen.

Matlab ziet de variabelen *voornaam* en *achternaam* als twee vectoren van losse karakters. Zo bestaat de variabele⁴ uit de karakters: m – a – r – k. Hoe kun je dit testen?

Typ in het Command Window de volgende tekst: *voornaam = 'Mark'*

Druk op enter.

Controleer in de Workspace of de variabele *voornaam* is aangemaakt

Als de variabele *voornaam* een vector is en we een vector opvatten als een variabele *waarin* andere variabele zijn opgeslagen dan zou je elk afzonderlijk karakter moeten kunnen uitlezen. In Matlab doe je dat als volgt:

```
>> voornaam = 'Mark'

voornaam =

Mark

>> voornaam(1)
```

*Figuur 25: de variabele (vector) met de naam **voornaam** bestaat uit meerdere karakters. Dit voorbeeld laat zien hoe je een los karakter van een string kunt uitlezen.*

Typ de code van Figuur 25 over een druk op enter

Wat valt op?

Als het goed is, valt op dat het karakter 'M' in het Command Window is te zien. Dat is inderdaad de eerste letter van de naam 'Mark'.

Hoe moet je de tweede letter van de voornaam laten zien?⁵

Het is ook mogelijk om een *selectie* aan karakters weer te geven. Bijvoorbeeld de karakters: 'ar' van de naam "Mark". Dat doe je als volgt:

⁴ Eigenlijk vector variabele

⁵ Als het goed is, is dit evident: `voornaam(2)`.

```
>> voornaam = 'Mark'

voornaam =

Mark

>> voornaam(2:3)

ans =

ar
```

Figuur 26: een selectie van een string.

Maar wacht! Wat wilde we ook alweer bereiken? Juist! Het combineren van verschillende strings (stukken tekst). Dat hebben we nog niet gedaan. Het enige wat we tot nu toe hebben gedaan is uitzoeken waarom we de specifieke foutmelding van Figuur 23 hebben gekregen. Dat zou nu een stuk duidelijker moeten zijn.

We weten dat de variabele *voornaam* en *achternaam* vectoren zijn. We kunnen een stukje van een string printen in het Command Window. Ook kunnen we één enkel karakter afdrukken in het Command Window. De foutmelding in Figuur 23 laat zien dat de vectoren *voornaam* en *achternaam* niet even lang zijn. Dat vindt Matlab niet leuk. Vectoren die bij elkaar worden opgeteld moeten exact even lang zijn. De oplossing voor ons probleem is simpel maar zonder Matlab kennis zeker niet voor de hand liggend: *maak van de twee strings één nieuwe string*.

Hoe doen we dat? Welke operator gebruiken we daarvoor?

We hebben eerder gezien dat Matlab een `[]` operator heeft om daar een vector mee te maken. Het aanmaken van een vector is eenvoudig:

```
>> [voornaam achternaam]

ans =

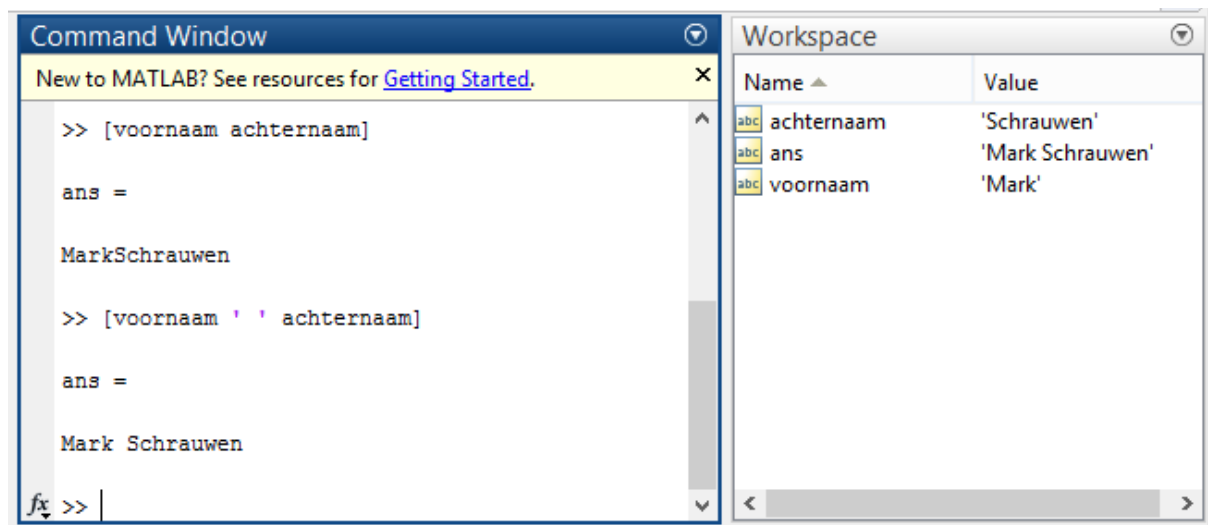
MarkSchrauwen
```

Figuur 27: twee strings gecombineerd door er één vector (string) van te maken.

Maar wacht dit resultaat is nog niet perfect. Immers 'MarkSchrauwen' is aan elkaar geschreven. We willen een string die er zo uit ziet: 'Mark Schrauwen'.

Hoe kunnen we dit doen?

We kunnen naast de variabelen ook een spatie aan de vector toe voegen. Dat doen we als volgt (Figuur 28):



Figuur 28: een overzicht van de resultaten van verschillende acties

5.4 Vragen en Opdrachten

1. Kunnen we een getal (bijvoorbeeld 23) in Matlab opslaan als een string?
2. Wat is het voordeel van het gebruik maken van variabelen?
3. Kunnen we een getal (bijvoorbeeld 23) in Matlab opslaan als een numerieke variabele?
4. Schrijf de code om de situatie in vraag 1 te illustreren.
5. Schrijf de code om de situatie in vraag 2 te illustreren.
6. Stop de volgende namen in afzonderlijke strings (variabelen):
 - Mark
 - Aad
 - Schrauwen
 - Vardy
 - Lagerberg
 - Alistair

Schrijf de code om de namen correct te combineren (denk aan de spaties!).

7. Maak een script dat de derde graads vergelijking: $y = ax^3 + bx^2 + cx + d$ berekend. Stop de coëfficiënten (a,b,c en d) en de onafhankelijke variabele x in Matlab variabelen. Test je script door a=b=c=d=1 en x=2. Het antwoord moet dan 15 zijn. (Tip, lees paragraaf 5.3.1)
8. Maakt het in de vorige opgave uit of je eerst de vergelijking declareert (opschrijft) en daarna pas de variabelen? Test dit met Matlab
9. Maak een string met de tekst: `langtekst = 'abcABCdefDEFghiGHI'`. Zorg dat je een nieuwe string maakt waarbij de kleine karakters van de variabele *langtekst* bij elkaar staan. Maak ook een nieuwe string waarbij de hoofdletters van de variabele *langtekst* bij elkaar staan.
10. Print van de in opgave 5 gemaakte voor- en achternamen de eerste letter.
11. Hoe kunnen we een spatie toevoegen tussen een variabele *voornaam* en een variabele *achternaam* zoals gebruikt in de bovenstaande tekst?
12. Welke operator wordt gebruikt voor het maken van een vector? Welke heb je hiervoor geleerd?

```
a = 1;  
x = 2;  
d = 4;  
c = 3;  
b = 2;  
a = 3;  
y = (a*x^3) + (b*x^2) + (c*x) + d
```

Figuur 29: voorbeeld code.

13. Zie de code in Figuur 29 en neem de code over. De variabele y zou eigenlijk de waarde 26 moeten hebben. Waarom geeft Matlab een andere uitkomst terug? Hoe los je dit op?

5.5 Antwoorden

1. Ja, dat hebben we in de tekst gezien en in Matlab uitgevoerd.
2. Een variabele maakt van een (numerieke) waarde, als het ware, een naam. Dat maakt code beter leesbaar. Bovendien, en dit is het belangrijkste voordeel, zorgt een variabele dat je een (numerieke) waarde slechts op één plek hoeft in te voeren. Dus als in een stuk code een aantal keer de waarde 100 nodig hebt, laten we zeggen zeven keer, en je moet die waarde veranderen naar 110. Dan moet je die waarde zeven keer aanpassen naar 110. Bij het gebruik van een variabele hoeft je alleen de waarde van de variabele aan te passen. Het gebruik van variabelen zorgt simpelweg voor tijdsbesparing en code duidelijkheid.
3. Ja, dat hebben we in de tekst gezien en in Matlab uitgevoerd.
4. Variabele = '23';
5. Variabele = 23
6. Het resultaat van deze oefening moet worden verkregen in Matlab. Je weet dus automatisch of je code correct is.
7. Breidt de tweede graads oefening uit. Je kunt zelf het antwoord testen m.b.v. Matlab.
8. Ja! Dan krijg je deze foutmelding:

```
16
17 - y = (a*x^3) + (b*x) + (c*x) + (d)
18 - a = 1;
19 - b = 1;
20 - c = 1;
21 - d = 1;
22 - x = 2;

Command Window

>> ReaderTryouth5
Undefined function or variable 'd'.

Error in ReaderTryouth5 (line 17)
y = (a*x^3) + (b*x) + (c*x) + (d)
```

In regel 17 staat eerst een vergelijking die aanneemt dat de variabelen a, b, c, d en x reeds in de workspace staan. Maar dat staan ze nog niet totdat regels 18-22 zijn uitgevoerd. Voor de vergelijking in regel 17 is het alsof de variabelen in regels 18-22 niet bestaan. In een programma / script worden regels achtereenvolgens uitgevoerd. Dat wil zeggen dat volgorde van belang is. Als je sochtend opstaat en eerst plast en daarna naar de wc gaat heb je een probleem. Je gaat eerst naar de wc en gaat dan plassen.

9. Er zijn verschillende manieren om tot het juiste antwoord te komen. De meest korte is: `kleineletters = langtekst([1:3 7:9 13:15])` en `hoofdletter = langtekst([4:6 10:12 16:18])`
10. Dit is een Matlab opdracht. Matlab laat automatisch zien of je het goed hebt gedaan. Pas jouw code aan totdat het werkt.
11. Door de spatie karakter toe te voegen aan de vector zoals in Figuur 28.
12. De rechte haken [en].
13. Als er goed gekeken wordt, zie je dat de variabele a er twee keer in staat. Namelijk in regel 6 en in regel 11. Echter hebben beide variabele een andere waarde. Matlab werkt van boven naar beneden dus de waarde van a in regel 6 wordt overgeschreven door de waarde van regel 11; oftewel de waarde 3. Hierdoor wordt in de vergelijking gewerkt met a = 3 wat als uitkomst 42 geeft.

5.6 Vectoren

Zonder dat je het door hebt, heb je nu al gewerkt met vectoren. Maar toch verdient deze belangrijke Matlab variabele zijn eigen paragraaf.

Een vector is, zoals eerder aangegeven, een type variabele waarin andere variabele kunnen worden gestopt. In het voorbeeld van een string (zie in de vorige paragrafen) hebben we karakters (chars) toegevoegd aan de vector. We konden elk afzonderlijk karakter uitlezen uit de vector (variabele).

Hoe deden we dat ook al weer?

We zijn zelfs in staat om delen van een vectoren af te drukken naar het Command Window. Maar we kunnen nog veel meer met vectoren. Een vector is niet alleen nuttig om karakters op te slaan om zodoende een string te creëren hij kan ook worden gebruikt een lijst van getallen op te slaan.

Waarom is het opslaan van een lijst getallen handig?

We gaan wederom de blokhaken operator gebruiken Stel dat we de getallen van 0 tot en met 9 willen opslaan in een vector. Dan doen we dat in Matlab als volgt:

```
>> vector = [0 1 2 3 4 5 6 7 8 9]

vector =

     0     1     2     3     4     5     6     7     8     9
```

Figuur 30: een vector van getallen opgeslagen in een vector met de naam vector.

Typ de code in Figuur 30 over

Druk op enter

Zorg dat de getallen 3 tot en met 7 worden afgedrukt

5.6.1 Vector index (indices)

Zodra een vector in het geheugen staat kan de vector in zijn geheel worden afgedrukt door het typen van de naam van deze vector. Tevens kan **een deel** van de vector worden afgedrukt door het opgeven van een aantal getallen.

Een index is de plek in een vector waar een bepaalde variabele staat. Zo is de waarde van de variabele `vector` in Figuur 30 met index 7 'vector(7)' gelijk aan de waarde 6. Tevens kan met de index van een vector een bereik aan getallen worden opgevraagd:


```

>> vector = 2:13

vector =

     2     3     4     5     6     7     8     9    10    11    12    13

>> vector(1:3)

ans =

     2     3     4

>> vector([1:3 2:5])

ans =

     2     3     4     3     4     5     6

```

Figuur 31: In de eerste regel is een vector aangemaakt. In de tweede regel (begint met '>>') zijn de eerste drie waarde van de vector geprint. In de laatste regel is iets bijzonders gebeurd. Zie voor uitleg de tekst.

In Figuur 31 is in de laatste regel (die begint met '>>') te zien dat er in dit geval geen sprake is van een volgorde. Je kunt de elementen van een vector in willekeurige volgorde ophalen. In de laatste regel is te zien dat de eerste drie elementen worden opgehaald en vervolgens het tweede tot en met het vijfde element. Het resultaat van deze regel code staat direct daaronder.

Wat in Matlab raar genoeg niet mogelijk is, is dit:

```

>> vector([3:1 5:2])

ans =

Empty matrix: 1-by-0

>> 5:2

ans =

Empty matrix: 1-by-0

>> 2:5

ans =

     2     3     4     5

```

Figuur 32: In Matlab is niet alles mogelijk. Als met behulp van de colon-operator (:) een sequentie wordt gegenereerd dan moet dit een oplopende sequentie zijn. Dus de getallen moeten steeds groter worden. In elk ander geval is het resultaat een lege matrix.

Je denkt nu, wat is hier handig aan? Eerlijk gezegd nog niet heel veel. De volgende sectie laat zien waarom numerieke vectoren toch handig zijn.

Typ de code van Figuur 31 over en voer de code uit in Matlab.

5.7 Vectoren optellen

Je hebt eerder al getallen bij elkaar opgeteld. Maar in Matlab kunnen we ook vectoren bij elkaar optellen. Eerder heb je al gezien dat we dat hebben geprobeerd bij strings in Figuur 23. Dat ging daar mis omdat de strings verschillende lengtes hadden.

5.7.1 Genereren van een numerieke vector

Je kunt in Matlab snel een vector maken zoals in Figuur 30 door de volgende code:

```
>> vector = 0:9  
  
vector =  
  
    0    1    2    3    4    5    6    7    8    9
```

Figuur 33: vector snel aanmaken

De `:` operator (in het Engels: de colon operator genoemd, onthoud dit goed, want je zult de term 'colon' regelmatig tegenkomen in foutmeldingen) wordt net als de blokhaken-operator gebruikt om een vector aan te maken. Echter, wordt deze operator gebruikt om alleen numerieke vectoren⁶ aan te maken.

Maak in Matlab een vector aan zoals in Figuur 33 stop deze in een variabele met de naam *vector1*

Maak in Matlab als volgt een tweede vector aan genaamd *vector2* aan: `vector2 = vector1;`

Controleer de inhoud van de twee verschillende vectoren, wat valt op?

5.7.2 Optellen van vectoren

Als het goed is, heb je in de gaten dat in de variabele *vector1* en *vector2* exact dezelfde waardes zitten. Als vectoren dezelfde lengte hebben dan kan de vector worden opgeteld. In ons geval kan dat als volgt:

```
vector1 =  
  
    0    1    2    3    4    5    6    7    8    9  
  
>> vector2 = vector1  
  
vector2 =  
  
    0    1    2    3    4    5    6    7    8    9  
  
>> vector1+vector2  
  
ans =  
  
    0    2    4    6    8   10   12   14   16   18
```

Figuur 34: vectoren bij elkaar optellen.

In Figuur 34 zijn twee vectoren bij elkaar opgeteld. Wat valt op? Elk element van de *vector1* is bij hetzelfde element van *vector2* opgeteld. Dit noemen we een *pair-wise addition* operatie. Elke vector-

⁶ Numerieke vectoren zijn vectoren met alleen getallen.

element paar is bij elkaar opgeteld. Nu zie je waarom vectoren zo handig zijn. Met 1 optelling kan je een heleboel getallen tegelijk optellen.

Kunnen we dit ook voor aftrekken, vermenigvuldigen en delen doen?

Trek de vectoren van elkaar af in Matlab

Wat laat het resultaat zien? Gebruik de Work space.

5.8 Element-by-Element vector operaties

Je kunt elke vector vermenigvuldigen met een getal:

```
>> vector * 3  
  
ans =  
    0     3     6     9    12    15    18    21    24    27
```

Figuur 35: vector maal een getal, elk element van de vector wordt met de waarde 3 vermenigvuldigt.

Probeer dit zelf in Matlab met een andere waarde

Dit zorgt dat elk element van een vector wordt vermenigvuldigd met dat getal. Dit kun je ook doen voor het optellen van een getal.

Je kunt dit ook doen voor het optellen van een getal bij een vector.

Typ in Matlab: 'vectorNieuw = vector1 + 100'

Wat valt op? Wat is er veranderd aan elk element van de vector? Onthoudt dat Matlab alle antwoorden op de gestelde vragen geeft, als je ze zelf invoert. Het resultaat van de vector genaamd 'vectorNieuw' is bijvoorbeeld te zien in de Workspace.

Kun je ook vectoren met elkaar vermenigvuldigen?

Probeer de vector met de naam **vector1** te vermenigvuldigen met de vector **vector2**.

Deze laatste actie gaat waarschijnlijk fout. Waarom? Omdat Matlab niet weet wat jij als gebruiker wil. Wil je het eerste element van de eerste vector vermenigvuldigen met alle elementen van de tweede vector? Of juist alleen met het tweede element? Of alle elementen van de ene vector met alle corresponderende elementen van de andere vector? Matlab weet dat niet.

```
>> vector1 * vector2
Error using *
Inner matrix dimensions must agree.

>> vector1 .* vector2

ans =

    0    1    4    9   16   25   36   49   64   81
```

Figuur 36: vector vermenigvuldiging.

De oplossing is als volgt. Je gebruikt voor vector vermenigvuldiging (element-by-element) de combinatie van twee operatoren namelijk de `.` en de `*`. Deze combinatie zorgt dat een vector element voor element wordt vermenigvuldigd (zie Figuur 36). De `.` operator (punt-operator) wordt ook wel de selectie-operator genoemd. Feitelijk zeg je tegen Matlab bij gebruik van `.*` dat Matlab een waarde van de eerste vector moet selecteren en die moet vermenigvuldigen met de corresponderende waarde van de tweede vector. Dat wil zeggen de waarde van de tweede vector op dezelfde plek in de vector. Daarom moeten vectoren altijd dezelfde lengte (evenveel elementen) hebben.

Vermenigvuldig met de opgedane kennis de twee vectoren in Matlab

Maak als volgt twee nieuwe vectoren aan: `vector1 = [2 2 2]; vector2[1 2 3];`

Bereken de tweede machten van de vector met behulp van de `^` operator

5.8.1 Aanmaken van vectoren

Eerder heb je al gezien dat je vectoren op verschillende manieren kunt aanmaken. Je kunt bijvoorbeeld de colon-operator gebruiken. Deze specifieke operator heeft nog meer handigheidjes aan boord. Tot nu toe heb je gezien dat de colon-operator gehele getallen van `x:y` kan genereren. Maar wat als je puntkomma getallen zou willen genereren? Dat is bijvoorbeeld handig voor het aanmaken van een tijdvector. Dat doe je op de volgende manier (Figuur 37):

```
>> 0:3

ans =

    0    1    2    3

>> 0:0.1:3

ans =

Columns 1 through 11

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000

Columns 12 through 22

    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000    2.1000

Columns 23 through 31

    2.2000    2.3000    2.4000    2.5000    2.6000    2.7000    2.8000    2.9000    3.0000
```

Figuur 37: een tijd-vector aanmaken met behulp van de colon-operator.

Probeer dit zelf eens

Verander de waarde 0.1 in andere waardes en test wat er gebeurt

5.9 Type vectoren

De vectoren die je tot nu toe hebt gezien, zijn zogenaamde rij-vectoren. De waarden van een vector worden afgedrukt in een rij. Je hebt ook kolom-vectoren. Matlab is zo vriendelijk geweest ons te voorzien van een operator die heel gemakkelijk van een rij-vector een kolom-vector maakt of andersom. Dat doen we in Matlab met de `'` operator (zie Figuur 38). Deze operatie noemen we transponeren of noemen we de transponatie-operator.

```
>> 0:3  
  
ans =  
  
    0    1    2    3  
  
>> (0:3)'  
  
ans =  
  
    0  
    1  
    2  
    3
```

Figuur 38: van een rij-vector een kolom-vector maken.

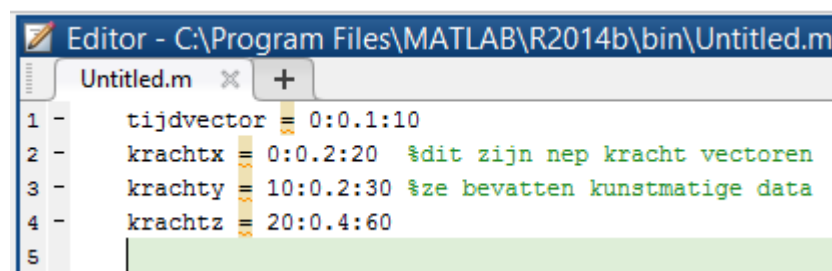
Probeer dit ook eens met een vector opgeslagen in een variabele

5.10 Vectoren van vectoren (een Matrix)

Dit had je vast al bedacht: kan je een vector van vectoren maken? Het antwoord is natuurlijk ja, anders hadden we er geen paragraaf aan gewijd.

Waarom wil je een vector van vectoren maken?

Het aanmaken van een vector van vectoren kan handig zijn wanneer je een meting met bijvoorbeeld een krachtenplaat hebt gemaakt. Dan krijg je een tijd-vector terug en 3 andere krachtvectoren (met dezelfde lengte als de tijd-vector). Vaak wil je deze vectoren combineren in één variabele. Hoe doe je dit? Eerst maken we verschillende vectoren aan:



```
Editor - C:\Program Files\MATLAB\R2014b\bin\Untitled.m  
Untitled.m x +  
1 - tijdvector = 0:0.1:10  
2 - krachtx = 0:0.2:20 %dit zijn nep kracht vectoren  
3 - krachty = 10:0.2:30 %ze bevatten kunstmatige data  
4 - krachtz = 20:0.4:60  
5 -
```

Figuur 39: zelf aangemaakte vectoren die allemaal andere getallen bevatten. In het echt komt de data in de vectoren van een meetapparaat, hier hebben we de data zelf aangemaakt.

Maak een nieuwe script aan.

Typ de code in Figuur 39 over en voer het uit.

Inmiddels is duidelijk dat we met de blokhaken-operator een vector kunnen maken bestaande uit verschillende variabelen. Een vector is op zichzelf een variabele. Maar een vector bestaat ook uit meerdere variabelen. Met de blokhaken-operator gaan we een vector van vectoren maken (zie Figuur 40):

```
9.1000  18.2000  28.2000  56.4000
9.2000  18.4000  28.4000  56.8000
9.3000  18.6000  28.6000  57.2000
9.4000  18.8000  28.8000  57.6000
9.5000  19.0000  29.0000  58.0000
9.6000  19.2000  29.2000  58.4000
9.7000  19.4000  29.4000  58.8000
9.8000  19.6000  29.6000  59.2000
9.9000  19.8000  29.8000  59.6000
10.0000 20.0000  30.0000  60.0000

fx >> [tijdvector' krachtx' krachty' krachtz']
```

Figuur 40: een vector van vectoren.

Opvallend zijn de apostroffen in Figuur 40. Die staan er natuurlijk, omdat de eerder aangemaakte vectoren rij-vectoren zijn en we er kolomvectoren van willen maken.

Sla de eerder aangemaakte vectoren als rij-vectoren op in een vector

Kun je een vector van een vector ook opslaan in een variabele?

De eerder aangemaakte vector van een vector noemen we een **matrix**! Daar komt de naam Matlab ook vandaan (MATrix LABoratory). Matlab kan heel goed met matrices omgaan. Matrices kunnen we ook opslaan in variabelen. In het voorbeeld van Figuur 40 doen we dat als volgt:

```
9.8000  19.6000  29.6000  59.2000
9.9000  19.8000  29.8000  59.6000
10.0000 20.0000  30.0000  60.0000

fx >> Matrix = [tijdvector' krachtx' krachty' krachtz']
```

Figuur 41: de matrix van Figuur 40 op slaan in een variabele genaamd Matrix.

5.11 Vragen en opdrachten

1. Welke twee type vectoren bestaan er in Matlab?
2. Met welke operator(en) kan een vector worden aangemaakt?
3. Met welke operator kan een rij-vector in een kolom-vector worden omgevormd?
4. Herhaling: met welke operator kunnen we een macht berekenen van een getal?
5. Kan in Matlab een vector worden gemaakt bestaande uit strings & numerieke waardes?
6. Kan elk element van een vector individueel worden aangepast?
7. Maak in Matlab een 3-bij-3 matrix aan gevuld met de getallen 0 tot en met 9.
8. Waarom is de punt-operator (.) nodig bij het vermenigvuldigen en delen van vectoren?
9. Wat is het symbool van de transponatie-operator en waar gebruik je deze operator voor?
10. Bestudeer de onderstaande regels code. Welke code 'begrijpt' Matlab?

```
Data1 = 15:0.3:5  
Data2 = 5:0.5:10
```

11. Wat wordt er met een pair-wise addition operatie bedoeld?
12. Wat klopt er niet aan onderstaande code?

```
Vector1 = 1:3;  
Vector2 = 4:6;  
Vector3 = 7:9;  
Matrix = (Vector1' Vector2' Vector3')
```

Figuur 42: Wat klopt er niet?

13. Zie de volgende code in Figuur 43. Na het uitvoeren van het script had de variabele y de waarde 26 moeten bevatten. Er gaat iets mis. Onderzoek wat er mis gaat en verbeter de code.

```
1 - getalAlsString = '23';  
2 - a = 1;  
3 - x = 2;  
4 - d = 4;  
5 - c = 3;  
6 - b = 2;  
7 - a = 3;  
8 - y = (a*x^3) + (b*x^2) + (c*x) + d
```

Figuur 43: zie opgave 13.

14. Een student besluit de code van de vorige opgave in minder regels op te schrijven (zie Figuur 44). Werkt deze code nog? Is het verstandig om de code op deze manier te presenteren?

```
1 - getalAlsString = '23'; a = 1; x = 2; d = 4; c = 3; b = 2; a = 3;  
2 - y = (a*x^3) + (b*x^2) + (c*x) + d
```

Figuur 44: dezelfde code als in Figuur 43 maar dan in minder regels.

15. Voer de code in Figuur 45 uit. Wat gaat er fout?

```
1 - clear all;  
2 - y = (a*x^3) + (b*x^2) + (c*x) + d;  
3 - a = 1;  
4 - x = 2;  
5 - d = 4;  
6 - c = 3;  
7 - b = 2;
```

Figuur 45: de code van Figuur 43 maar dan herschreven met een aanvulling..

5.12 Antwoorden

1. Rij-vectoren en kolom-vectoren
2. Met de colon-operator en de blokhaken-operator
3. Met de aanhalingsteken-operator.
4. Met ^
5. Nee dat kan niet. Er bestaan wel andere type variabelen waarin dat mogelijk is.
6. Ja dat kan. Stel je hebt een numerieke vector genaamd *schrauben* dat bestaat uit 100 elementen. Dan kan element 99 als volgt worden aangepast: `schrauben(99) = 1;`

```
>> [1 2 3; 4 5 6; 7 8 9;]

ans =

     1     2     3
     4     5     6
     7     8     9
```

7. dit mag ook: `[[1 2 3]' [4 5 6]' [7 8 9]']`
8. Die is nodig voor een pair-wise operatie zodat elk element van vector A wordt gebruikt op het element van vector B met dezelfde index.
9. Deze operator maakt van een rij-vector een kolom-vector. Of hij maakt van een 4-bij-2 matrix een 2-bij-4 matrix. Het symbool is de apostrof.
10. Dit kan je, als zo vaak, zelf testen met Matlab. Het antwoord is Data2. Met de colon-operator kan je alleen oplopende sequenties aanmaken (van laag naar hoge getallen). Als je toch een sequentie (vector) van getallen wilt maken m.b.v. de colon-operator krijg je een lege vector.
11. Dat elk element van een vector bij hetzelfde element wordt opgeteld/afgetrokken/vermenigvuldigd met een andere vector.
12. In deze codering wordt er in plaats van blokhaken `[]`, haken `()` gebruikt. In H5.3.1 heb je geleerd dat vectoren worden gemaakt met blokhaken en niet met haken. Er kan dus geen Matrix worden gemaakt.
13. In regel 7 staat opnieuw de variabele *a* gedeclareerd. De oude waarde van de variabele *a* (namelijk de waarde 1) wordt nu overschreven door variabele declaratie op regel 7.
14. Ja de code werkt nog. Het is niet verstandig omdat de leesbaarheid van de code op deze manier in het gedrang komt.
15. Matlab geeft een foutmelding die je verder helpt:

```
Undefined function or variable 'a'.
```

```
Error in Untitled (line 2)
```

```
y = (a*x^3) + (b*x^2) + (c*x) + d;
```

Het probleem is dat je op regel 2 een formule uitvoert die gebruikt maakt van verschillende variabelen. Een programma wordt altijd van boven naar beneden uitgevoerd. Dus regel 2 wordt eerst uitgevoerd daarna wordt regel 3 uitgevoerd. In regel 2 wordt er gebruik gemaakt van verschillende variabelen. Deze variabelen zijn nog niet aangemaakt op het moment dat regel 2 wordt uitgevoerd. Matlab geeft om die reden een foutmelding; er kan geen gebruik worden gemaakt van een variabele die niet is aangemaakt.

5.13 Veel gebruikte vectorfuncties

Matlab komt standaard met een hoop functionaliteit. Deze functionaliteit is ingepakt in zogenaamde functies. Hoe je een functie moet maken, leer je later.

In deze paragraaf gaan we kijken naar veel gebruikte standaard functies in relatie tot vectoren en matrices. Deze functies heb je later nodig in de eind- en weekopdrachten.

5.13.1 whos()

Een handige functie die vaak wordt gebruikt in relatie tot vectoren is de whos() functie. Met deze functie krijg je snel te zien van Matlab welke variabelen in de Work space staan:

```
>> whos
  Name      Size      Bytes  Class  Attributes
  ans       1x2         16  double
  mark      3x3         72  double
```

Figuur 46: een voorbeeld van de output van de functie whos

Op deze manier kun je snel inzicht verkrijgen in de grootte van een bepaalde variabele in de Work space.

Wat is de grootte van de (standaard)variabele ans?

De grootte is in dit geval 1x2. Wat zegt dit nu? Dit hangt samen met rij-vectoren en kolom-vectoren of beter gezegd: met de hoogte en de breedte van een vector (of matrix).

Gebruik zelf in Matlab de functie whos() door 'whos' te typen in het Command Window.

Druk op enter

5.13.2 size()

Met de functie size() kun je de afmetingen van een variabele opvragen. Je moet wel weten welke van welke variabele je de afmetingen wilt opvragen. Hier kun je de functie size() voor gebruiken:

```
>> size(ans)

ans =

     1     2
```

Figuur 47: een voorbeeld van de output van de functie size

Vier spieren hebben momentsarmen 3, 5, 2 en 4 cm. De fysiologische doorsneden zijn 12, 15, 3 en 7 cm². Maak een matrix in Matlab die deze gegevens bevat en gebruik de functie size() om de afmetingen van deze matrix op te vragen. Transponeer de matrix en vraag nogmaals de afmetingen op.

Merk op dat de informatie die de functie `size()` geeft ook is te zien in de output van de functie `whos()`.

5.13.3 `length()`

De functie `length()` hangt nauw samen met de functie `size()`. Op de achtergrond is de functie `length()` niets meer dan een functionaliteit die de grootste afmeting pakt van de functie `size()`. Dit is een voorbeeld van het gebruik van de functie `length()`:

```
>> length(ans)

ans =

     2
```

Figuur 48: een voorbeeld van de output van de functie `length`. Merk op dat `ans` in Figuur 47 de afmeting 1x2 had. De grootste afmeting is dus 2.

Gebruik zelf de functie `length()` voor de matrix uit de vorige vraag, maar voorspel eerst wat er uit zal komen voordat je hem uitvoert.

5.13.4 `randn()`

Het kan handig zijn om een matrix of vector te vullen met een aantal willekeurig gegenereerde getallen. Dat kan m.b.v. de functie `randn()`:

```
>> randn(1,2)

ans =

    0.5377    1.8339

>> randn(2,1)

ans =

   -2.2588
    0.8622
```

Figuur 49: een voorbeeld van de output van de functie `randn()`. Merk op dat de functie twee variabelen krijgt.

Merk op dat in Figuur 49 de functie `randn()` twee keer op een verschillende manier is aangeroepen. Wat is het verschil tussen de twee aanroepen?

Hoe kun je met de functie `randn()` een matrix (vectoren in een vector) maken?

Zoals je hebt gezien kun je met de getallen die aan de functie `randn()` geeft de dimensies van de output bepalen. *Hoe moet je een 3 bij 3 matrix genereren m.b.v. de functie `randn()`?*

Typ in Matlab: ‘`randn(3,3)`’

Druk op enter

5.13.5 `fliplr()`

De laatste handige functie m.b.t. vectoren is de `fliplr()`. Inmiddels moet zijn opgevallen dat de functies in Matlab herkenbare namen hebben.

Wat doet de functie `fliplr()`?

De functie `fliplr()` zorgt dat de volgorde van een vector wordt omgedraaid:

```
vector =  
      0      1      2      3      4      5      6      7      8      9  
  
>> vectorlr = fliplr(vector)  
  
vectorlr =  
      9      8      7      6      5      4      3      2      1      0
```

Figuur 50: een voorbeeld van de output van de functie `fliplr()`.

Typ de code van Figuur 50 over

Druk op enter

Verzin zelf een paar andere vectoren (of matrices) en gebruik `fliplr()`.

Hopelijk heb je opgemerkt dat `fliplr()` alleen werkt voor rijvectoren en niet voor kolomvectoren. Dat kun je ook opmaken uit de naamgeving van de functie. Let dus goed op bij wat voor soort vectoren je deze functie toepast.

Merk op dat in Matlab de code `vector = 10:1` niet werkt maar dat we dit wel zelf kunnen creëren door gebruik van de functie `fliplr()`.

Typ voor de lol ook eens `why` in.