MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC

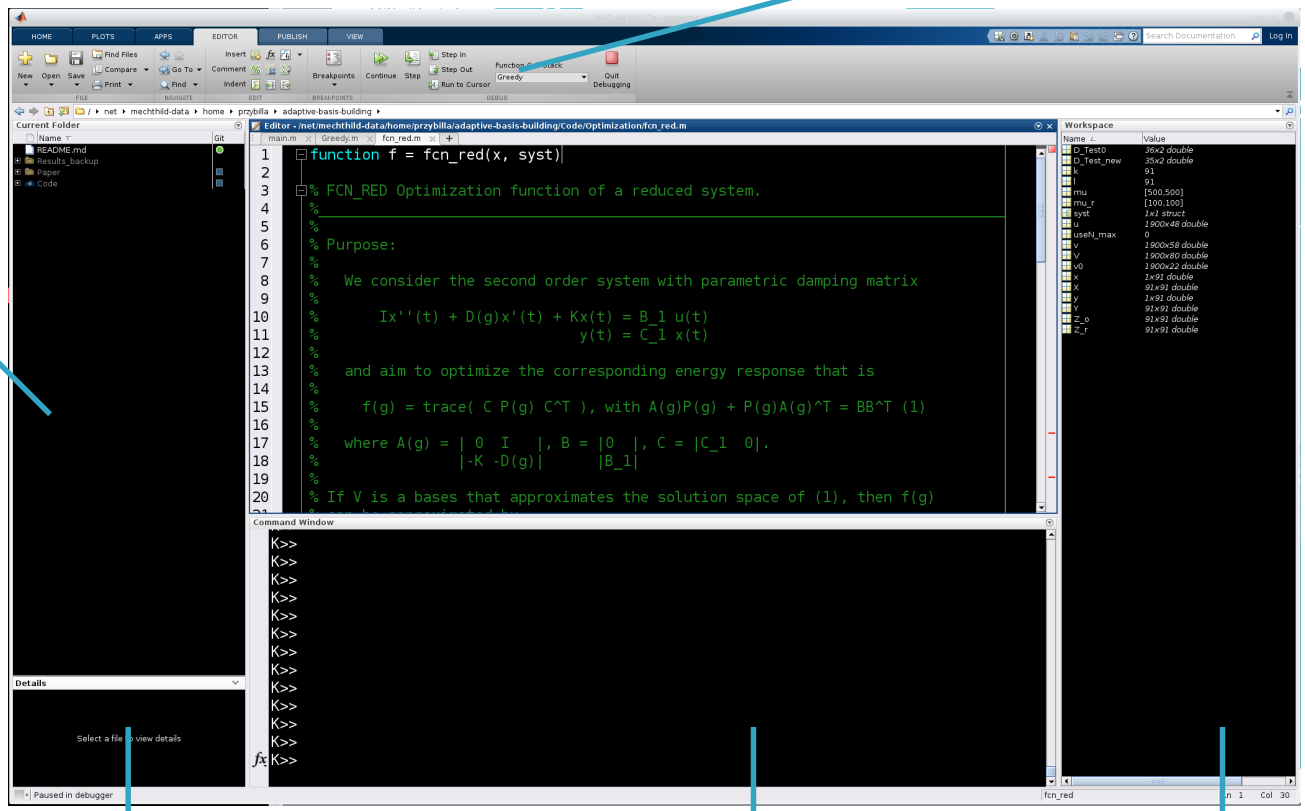
COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Matlab Einfühung

Jennifer Przybilla

02.11.2021

Aufbau


Menü
Datein & Ordner
Kommando-Fenster
Kurzbeschreibungen von ausgewählten Datein
Workspace, Liste der Variablen

Aufgabe

Aufgabe 1

- Öffnet Matlab
- Definiert die Variablen

```
m = 5;
n = 3;
```

- Gebt die folgenden Befehle ein:

```
m + n
m + n;
```

Was ist der Unterschied?

Ein wenig Syntax

- Multiplizieren: $m = 2n \quad \Rightarrow \quad \texttt{m} = 2 * \texttt{n}$
- Kommentare $\%$ `Alles hinter the Prozent Zeichen wird nicht ausgeführt, kann und soll also für Kommentare genutzt werden.`
- Vordefinierte Variablen: `pi`, `eps` (Maschinengenauigkeit)
- Vordefinierte Funktionen: $\sin(\cdot)$, $\cos(\cdot)$, $\mathrm{eps}(\cdot)$
- Test als Variable is ein *String* und wird definiert durch

$$\texttt{str} =' \texttt{Hallo}' \quad \text{oder} \quad \texttt{str} = \text{"}\texttt{Hallo}\text{"}.$$

- Code auf mehrere Zeilen verteilen:

$$m = 1 + 2 + 3 + 4 \ldots$$
$$+ 5 + 6 + 6 + 7 + 8.$$

Deklanierung von Variablen

- Boolean: $\mathtt{a} = \mathtt{true}$, $\mathtt{a} = \mathtt{false} \quad \Rightarrow \quad \mathtt{a} = 1$, $\mathtt{a} = 0$
- Skalar: $\mathtt{a} = 5 + 7;$
- Vektor:
  - Zeilenvektor:

$$\begin{aligned} \mathtt{v} = 1:4 \quad &\Rightarrow \quad v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \\ \mathtt{v} = [1,2,3,4] \quad &\Rightarrow \quad v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \\ \mathtt{v} = [1\ \ 2\ \ 3\ \ 4] \quad &\Rightarrow \quad v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \end{aligned}$$

  - Spaltenvektor: $\mathtt{v} = [1;2;3;4] \quad \Rightarrow \quad v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$

Deklanierung von Variablen

- Vektor:
  - Zeilenvektor: $\mathtt{v} = [1, 2, 3, 4] \quad \Rightarrow \quad v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix},$
  - Spaltenvektor: $\mathtt{v} = [1; 2; 3; 4] \quad \Rightarrow \quad v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$
- Matrizen:

$$\mathtt{A} = [1,\ 2,\ 3,\ 4;\ 5,\ 6,\ 7,\ 8;\ 9,\ 10,\ 11,\ 12]$$

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

Die Dimensionen müssen so gewählt werden, dass sie eine quadratische Matrix ergeben!

Deklanierung von Variablen

- Wie dürfen Variablen aussehen?
- Sie dürfen enthalten:
  - Kleine & große Buchstaben, Zahlen und Unterstriche _

Aufgabe 2

Welche der Variablennamen sind zulässig?

1. Variable1
2. String%
3. hallo_world
4. def Variable
5. Variable 1
6. nameFuerVariablen
7. hallo-Welt

Aufgabe

Aufgabe 3

Welchen Wert hat n nach den vorgegebenen Schritten?

$$\mathtt{n} = 8;$$
$$\mathtt{n} = \mathtt{n}\hat{}\,2;$$
$$\mathtt{n} = \mathtt{n}/8;$$
$$\mathtt{n} = \mathtt{n} * 2;$$
$$\mathtt{n} = \mathtt{n} + \mathtt{n};$$
$$\mathtt{n} = \mathtt{n} + 4;$$
$$\mathtt{n} = \mathtt{n}/2;$$
$$\mathtt{n} = \mathtt{n} + 3;$$
$$\mathtt{n} = (\mathtt{n} - 1) * \mathtt{n};$$

Zugriffe

- Es sei $a = \begin{bmatrix} 4 & 8 & 11 & 10 & 2 \end{bmatrix}$.
- Es sei $A = \begin{bmatrix} 9 & 1 & 11 & 4 \\ 2 & 13 & 6 & 14 \\ 8 & 3 & 7 & 20 \end{bmatrix}$.
- $\mathtt{a}(2) = 8$, $\mathtt{a}(1, 2) = 8$
- $\mathtt{a}(2 : 4) = \begin{bmatrix} 8 & 11 & 10 \end{bmatrix}$
- Definiere $\mathtt{v} = [1,\ 3] \quad \Rightarrow \quad \mathtt{a}(\mathtt{v}) = [4,\ 11]$.
- $\mathtt{A}(2,\ 3) = 6$,
- $\mathtt{A}(1 :\ 2,\ 3 :\ 4) = [11,\ 4;\ 6,\ 14]$.

Skripte und Funktionen

- Matlab bestehen aus Funktionen und Skripten $\Rightarrow$ .m - Dateien

Skripte

- Skript kennt alle Variablen aus dem aktuellen Workspace, kann sie verändern und überschreiben
- Das ist äquivalent dazu Befehle im Skript nacheinander in die Kommandozeile zu schreiben und auszuführen

Funktionen

- Funktionen haben eigenen Workspace
- Sie bekommen Variablen übergeben und geben sie aus.
- Variablen sind nur innerhalb der Funktion definiert.

Aufgabe

Aufgabe 4

1. Erstellt einen Ordner *NLA_Matlab*.
2. Erstellt ein Skript *MeinErstesSkript.m*.
3. Öffnen Sie mit Doppelklick das Skript.
4. Scheibt einige der eben gelernten Befehle in das Skript.
5. Führt das Skript aus indem ihr in das Kommandofenster `MeinErstesSkript` eingebt.
6. Gebt `clear all` ein, was passiert? Was wenn ihr `clc` eingebt?

Die Befehle `clear all` und `clc` könnt ihr an den Anfang von Skripten und Funktionen schreiben.

Operatoren

- Relationen/Bedingungen:
  - a == b, a ~= b, a < b, a <= b, a > b, a >= b:
    gleich, ungleich, kleiner (oder gleich), größ (oder gleich)
- Logische Operatoren:
  - && und, || oder, ~ nicht

Probiert aus:

```
x = 5;
y = (x < 6)||(x > 4);
disp(y);
z = (x > 0)&&~(x > 3);
disp(z);
```

If Abfrage

Aufbau

```
if Bedingung
    Anweisung
elseif Bedingung
    Anweisung
elseif Bedingung
    Anweisung
else
    Anweisung
end
```

Beispiel

```
if y < 0
    z = 0;
elseif x == 0
    z = 0;
elseif (x <= 0) && (y <= 0)
    z = -1;
else
    z = x + y;
end
```

for Schleife

Aufbau

```
for Variable = Startwert : Endwert
    Anweisung
end
```

Beispiel

```
for n = 0 : 10
    disp(n);
end
```

- Wenn wir nicht wollen dass n in 1-er Schritten größer wird, setzen wir:

$$\texttt{n} = 10 : -2 : 0 \quad \Rightarrow \quad n = 10,\ 8,\ 6,\ 4,\ 2,\ 0$$

Beispiel

Wir berechnen

$S = \sum_{k=0}^{n} \frac{(-1)^{-1}}{2k+1}$:

```
n = 5;
S = 0;
for k = 0 : n
    S = S + (-1)^1;/(2 * k + 1);
end
```

while Schleife

Manchmal wissen wir nicht wie viele Schleifen-Iterationen wir brauchen:

Beispiel

- Wir betrachten die Folge $x_{k+1} = \frac{1}{2}\left(x_k + \frac{2}{x_k}\right)$, $x_1 = 2$ die nach $\sqrt{2}$ konvergiert.
- Berechne x_{k+1} mit $k = 1, 2, \ldots$ bis $|x_{k+1} - x_k| < \mathrm{tol}$ für eine Toleranz tol.

Aufbau

```
while Bedingung
    Anweisung
end
```

Beispiel

$\mathtt{x\_old} = 0;\ \ \mathtt{x} = 2;\ \ \mathrm{tol} = 10\mathrm{e} - 10;$

```
while |x - x_old| < tol
    x_old = x;
    x = (x + 2/x)/2;
end
```

break and continue

- continue: aktueller Schleifendurchlauf wird übersprungen und es wird zum Schleifenanfang gesprungen.
- break: komplette Schleife wird unterbrochen und es wird zum Skriptteil nach der Schleife gesprungen.

Aufgabe 5

```
for i = 1 : 10
    if i == 4
        continue;
    end
    if i == 8
        break;
    end
    disp(i);
end
```

Welche Zahlen werden ausgegeben?

Funktionen

- Im Gegensatz zu Skripten köennen Funktionen so angelegt werden, dass sie Funktionswerte erwarten und Rückgabewerte liefern.

Beispiel

```
function[ar,per] = area(a,b)
  % AREA Compute area and
  perimeter of a rectange with
  side lenghts a and b.
  ar = a * b;
  per = 2 * a + 2 * b;
end
```

- Dateiname muss mit Funktionsname übereinstimmen ⇒ Datei muss `area.m` heißen.
- Funktion erwartet beim Aufruf zwei Variablen.
- Wenn Funktion keine Antwort erwartet ⇒ leere Klammern ( ).
- Funktion liefert zwei Ausgabewerte.
- Liefert Funktion keine Ausgabewerte ⇒ leere Klammern [ ].

Noch ein paar wichtige Funktionen

- `sin`$(\cdot)$, `cos`$(\cdot)$, `exp`$(\cdot)$, `log`$(\cdot)$
- `sqrt`$(\cdot)$, `abs`$(\cdot)$
- `length`$(\cdot)$, `size`$(\cdot)$
- `plot`(x, y), `plot`$(x, y, options)$
- `qr`$(\cdot)$, `lr`$(\cdot)$, `norm`$(\cdot)$
- $(\cdot)'$ - komplex konjugiert transponiert, `tranpose`$(\cdot)$ transponiert.

Aufgabe

Aufgabe 6

Schreibt eine Funktion `geomSum`, die die N-te Partialsumme der geometrischen Reihe

$$S = \sum_{k=0}^{N} q^k$$

berechnet. Die Eingaben sind N und q und die Ausgabe S.

Aufgabe

Aufgabe 6

Schreibt eine Funktion `geomSum`, die die N-te Partialsumme der geometrischen Reihe

$$S = \sum_{k=0}^{N} q^k$$

berechnet. Die Eingaben sind N und q und die Ausgabe S.

Lösung

```
function[S] = geomSum(N, q)
  % GEOMSUM Computes partial sum.
  S = 0;
  for k = 0 : N
      S = S + q^k;
  end
```

Thank you for your attention!