Matlab/Simulink Cheat Sheet

Matthias Thumann
https://github.com/m-thu/sandbox/blob/master/matlab.tex

Graphics

```
clear       % Clear all variables
close all   % Close all figures
format      % Reset output format for numbers to default

figure(n); % Select figure n

plot(y);             % Plots all values in y as a function of the index
plot(x, y);          % Plots all (x,y) pairs
axis([x1 x2 y1 y2]); % Specify plotting area

title('text');             % Plot title
xlabel('text');            % Label x-axis
ylabel('text');            % Label y-axis
grid;                      % Enable grid
legend('text1', 'text2'); % Legend in the order of the plotted functions

subplot(m, n, p); % m rows, n columns, p: number of subplot (numbered left to right,
                  % top to bottom)

hold on;  % Keep graphics in current figure
hold off; % Reset hold functionality
```

Control Flow Constructs

If ... else:

```
if a == 0
    b = 2;
elseif a < 0
    b = -a;
else
    b = a;
end;
```

Switch ... case:

```
switch (x)
    case (1)
        disp('x = 1')
    case (2)
        disp('x = 2')
    otherwise
        disp('default')
end;
```

For-loop:

```
for k = 1:100
    u(k) = k^2;
    if k == 9
        break;
end;
```

While-loop:

```
z = 0;
while z <= 10
    z = z + 1;
end;
```

Analysis of AC Circuits

Resistance R: $\underline{Z}_R = R$, $u_R = R \cdot i$

Inductance: L: $\underline{Z}_L = \mathrm{j}\omega L = sL$, $u_L = L \cdot \frac{\mathrm{d}\, i_L}{\mathrm{d}\, t}$

Capacity C: $\underline{Z}_C = \frac{1}{\mathrm{j}\omega C} = \frac{1}{sC}$, $i_C = C \cdot \frac{\mathrm{d}\, u_C}{\mathrm{d}\, t}$

Impedance of the circuit: $\underline{Z} = \ldots$, total current: $\underline{I} = \frac{\underline{U}}{\underline{Z}}$, voltages at the components:

```
omega = 2*pi * f;
Z_C   = 1 / (1j * omega * C);
Z_L   = 1j * omega * L;
Z     = R + Z_C + Z_L;

i   = U / Z; % U: Amplitude
U_R = i * R;
U_C = i * Z_C;
U_L = i * Z_L;
```

Plotting voltages in the time domain:

```
T = 1 / f;
t = 0:T/200:T;

u_in = U * sin(omega*t);
u_r  = abs(U_R) * sin(omega*t + angle(U_R));
u_c  = abs(U_C) * sin(omega*t + angle(U_C));
u_l  = abs(U_L) * sin(omega*t + angle(U_L));

figure(1);
plot(t, u_in, t, u_r, t, u_c, t, u_l);
legend('u_{in}', 'u_r', 'u_c', 'u_l');
xlabel('t/s'); ylabel('u/V');
```

Phasor diagram of all voltages:

```
figure(2);
compass(U   , 'k'); hold on;
compass(U_R, 'b'); hold on;
compass(U_C, 'g'); hold on;
compass(U_L, 'r'); hold off;
legend('U_{in}', 'U_R', 'U_C', 'U_L');
xlabel('Re U_k \rightarrow'); ylabel('Im U_k \rightarrow');
```

Bode plot of a transfer function:

```
w = 2*pi*logspace(-1, 5); % Angular frequency (1/s)
H = ...;                  % Transfer function

figure(3);
subplot(2, 1, 1);               % Two rows, one column
semilogx(w, 20*log10(abs(H))); % Magnitude in dB
subplot(2, 1, 2);
semilogx(w, angle(H));          % Phase in rad
```

System Analysis using ODEs/Transfer functions

Transfer function: $H(s) = \frac{Y(s)}{X(s)}$ with input $X(s)$ and output $Y(s)$

$y(t) \circ\!\!-\!\!\bullet\, Y(s)$, $\dot{y}(t) \circ\!\!-\!\!\bullet\, s \cdot Y(s)$, $\ddot{y}(t) \circ\!\!-\!\!\bullet\, s^2 \cdot Y(s)$, $\ldots$

Calculation of the range of the angular frequnecy for the Bode plot of the transfer function:

```
omega_min = ...;
omega_max = ...;
w         = logspace(log10(omega_min), log10(omega_max));
```

State space

Conversion of an ODE of n^{th} order to a system of n ODEs of first order (input u, output y):

$$a_n \cdot y^{(n)} + a_{n-1} \cdot y^{(n-1)} + \cdots + a_2 \cdot \ddot{y} + a_1 \cdot \dot{y} + a_0 \cdot y = b_0 \cdot u$$

Introduction of state space variables:

$$x_1 = y, x_2 = \dot{y}, x_3 = \ddot{y}, \ldots, x_{n-1} = y^{(n-2)}, x_n = y^{(n-1)}$$
$$\Rightarrow \dot{x}_1 = x_2, \dot{x}_2 = x_3, \ldots, \dot{x}_{n-1} = x_n \qquad (n-1 \text{ equations})$$

$$a_n \cdot \dot{x}_n + a_{n-1} \cdot x_n + a_{n-2} \cdot x_{n-1} + \cdots + a_2 \cdot x_3 + a_1 \cdot x_2 + a_0 \cdot x_1 = b_0 \cdot u$$
$$\Rightarrow \dot{x}_n = \frac{-a_{n-1}}{a_n} \cdot x_n - \frac{a_{n-2}}{a_n} \cdot x_{n-1} - \cdots - \frac{a_2}{a_n} \cdot x_3 - \frac{a_1}{a_n} \cdot x_2 - \frac{a_0}{a_n} \cdot x_1 + \frac{b_0}{a_n} \cdot u \qquad (1 \text{ equation})$$

Matrix representation (only for linear ODEs):

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} x_1 \\ \cdots \\ \cdots \\ x_n \end{bmatrix} = \underbrace{A}_{\text{state matrix}} \cdot \vec{x} + B \cdot \vec{u}$$

$$\vec{y} = \underbrace{C}_{\text{output matrix}} \cdot \vec{x} + D \cdot \vec{u}$$

Solving ODEs with Matlab

Function-file `dgl.m` with the state space representation of the ODE in this form: $\dot{\vec{x}} = \vec{f}(\vec{x})$:

```
function xp = dgl(t, x, dummy, parameter1, parameter2, ...)

xp = zeros(n, 1); % xp: d/dt x, column vector
                  %  n: order of the ODE

xp(1) = ...;
xp(2) = ...;
% ...
xp(n) = ...;
```

Invocation of the solver:

```
t_span = [0 t_max];          % Time interval for simulation
x0     = [x1_0, x2_0, ...]; % Initial values of the ODE

[t, x] = ode45('dgl', t_span, x0, [], parameter1, parameter2, ...);

% t: Time vector returned by the solver
% x: column 1: x1, column 2: x2, ...

figure(1); plot(t, x(:, 1)); % x1 plotted against t
figure(2); plot(t, x(:, 2)); % x2 plotted against t
```

Harmonic oscillator: $\ddot{y} + \delta \cdot \dot{y} + \omega_0^2 \cdot y = \frac{1}{m} \cdot F_{\text{ext}}$

Damped free oscillation ($\delta > 0$):

1. $\delta < \omega_0$: Underdamped, $\omega = \sqrt{\omega_0^2 - \delta^2}$
2. $\delta = \omega_0$: Critically damped
3. $\delta > \omega_0$: Overdamped

Forced oscillation by excitation with an external force $F_{\text{ext}}(t) = A_0 \cdot \sin(\omega_{\text{f}} \cdot t)$:

1. $\omega_f \ll \omega_0$: System is in phase with the external excitation after some time.

2. $\omega_f \approx \omega_0$: Resonance, phase difference of 90°.

3. $\omega_f \gg \omega_0$: Phase difference of 180°

Solving ODEs with Simulink

Passing vectors from the Matlab workspace using the *From Workspace* block, data format: Matrix with the time vector as the first column and data vector(s) as additional column(s):

```
x = [1:10];          % Parameter as a row vector
t = 0:length(x)-1;   % Time as a row vector

param = [t.', x.']; % Simulink format for passing parameters
```

Transfer of vectors to the Matlab workspace using the *To Workspace* block, Save Format: Array.

Invocation of a Simulink simulation from Matlab (Name of the Simulink model must not be identical to the name of the invoking Matlab m-file!):

```
tout = sim('example_sim', time_vect);

% time_vect: defines temporal resolution and simulation interval
% tout     : internally used time vector
```

Graphical solution of ODEs using Simulink:

1. ODE of n^{th} order:
$$a_n \cdot y^{(n)} + a_{n-1} \cdot y^{(n-1)} + \cdots + a_2 \cdot \ddot{y} + a_1 \cdot \dot{y} + a_0 \cdot y = b_0 \cdot u$$

2. Solve for the highest derivative:
$$y^{(n)} = \frac{-a_{n-1}}{a_n} \cdot y^{(n-1)} - \frac{a_{n-2}}{a_n} \cdot y^{(n-2)} - \cdots - \frac{a_1}{a_n} \cdot \dot{y} - \frac{a_0}{a_n} \cdot y + \frac{b_0}{a_n} \cdot u$$

3. Successive integration, starting with the highest derivate:
$$y^{(n-1)} = \int y^{(n)},\ y^{(n-2)} = \int y^{(n-1)},\ \ldots\ \dot{y} = \int \ddot{y},\ y = \int \dot{y}$$

4. Simulink block diagram:

Simulations using Simulink

Avoiding algebraic loops: Insert *Algebraic Constraint* block.