Matlab Controller Design

1. Control system toolbox
2. Functions for model analysis
3. Linear system simulation
4. Biochemical reactor linearization
Control System Toolbox

- Provides algorithms and tools for analyzing, designing and tuning linear control systems.
- System can be specified as a transfer function, state-space and pole-zero-gain model.
- Provides tools for model representation conversion and low-order approximation of high-order systems.
- Allows series, parallel, feedback and general block-diagram connection of linear models.
- Interactive tools and command-line functions, such as the step response plot, allow visualization of system behavior in the time domain.
- Provides tools for automatic PID controller tuning, root locus analysis, and other interactive and automated techniques.
- Controller design can be validated by verifying rise time, overshoot, settling time and other requirements.
Selected Functions for Model Analysis

- Model creation and conversion
  - tf – create transfer function (TF) model
  - zpk – create zero/pole/gain (ZPK) model
  - ss – create state-space (SS) model

- System gain and dynamics
  - dcgain – steady-state gain
  - pole – system poles
  - zero – system zeros
  - pzmap – pole-zero map

- Linear system simulation
  - step – step response
  - stepinfo – step response characteristics (rise time, ...)
  - impulse – impulse response
  - lsim – response to user-defined input signal
  - lsiminfo – linear response characteristics
  - gensig – generate input signal for lsim

- Time delays
  - pade – pade approximation of time delay
Transfer Function Model Creation

- \( g = \text{tf}(\text{num},\text{den}) \) creates a transfer function \( g \) with numerator(s) and denominator(s) specified by \( \text{num} \) and \( \text{den} \)

\[
\text{>> num} = [3 \ -2 \ 1];
\]
\[
\text{>> den} = [4 \ 3 \ 2 \ 1];
\]
\[
\text{>> g} = \text{tf}(\text{num},\text{den})
\]

Transfer function:
\[
\frac{3 s^2 - 2 s + 1}{4 s^3 + 3 s^2 + 2 s + 1}
\]

- \( \text{>> zero}(g) \)
  \[
  \text{ans} = 0.3333 + 0.4714i \\
  0.3333 - 0.4714i
  \]

- \( \text{>> pole}(g) \)
  \[
  \text{ans} = -0.6058 \\
  -0.0721 + 0.6383i \\
  -0.0721 - 0.6383i
  \]

- \( \text{>> dcgain}(g) \)
  \[
  \text{ans} = 1
  \]
Transfer Function Model Conversion

```plaintext
>> ss1=ss(g)

a =

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>-0.75</td>
<td>-0.5</td>
</tr>
<tr>
<td>x1</td>
<td></td>
<td>-0.5</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x3</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

b =

<table>
<thead>
<tr>
<th>u1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
</tr>
<tr>
<td>x2</td>
</tr>
<tr>
<td>x3</td>
</tr>
</tbody>
</table>

>> g1=tf(ss1)

Transfer function:

\[
\frac{0.75 s^2 - 0.5 s + 0.25}{s^3 + 0.75 s^2 + 0.5 s + 0.25}
\]
```
Linear System Step Response

- \([y,t] = \text{step}(\text{sys})\) plots the step response of the model \(\text{sys}\) (created with either \text{tf}, \text{zpk}, \text{or ss}).

\[
\begin{align*}
\text{>> step(g)} \\
\text{>> stepinfo(g)} \\
\text{ans =} \\
\text{RiseTime: 6.2094} \\
\text{SettlingTime: 53.9244} \\
\text{SettlingMin: 0.5793} \\
\text{SettlingMax: 1.5898} \\
\text{Overshoot: 58.9773} \\
\text{Undershoot: 6.3113} \\
\text{Peak: 1.5898} \\
\text{PeakTime: 8.8029}
\end{align*}
\]
lsim(sys,u,t) plots the time response of the model sys to the input signal described by u and t.

[u,t] = gensig(type,tau) generates a scalar signal u of class type and period tau. The following classes are supported:

- type = 'sin'  sine wave
- type = 'square'  square wave
- type = 'pulse'  periodic pulse

gensig returns a vector t of time samples and the vector u of signal values at these samples. All generated signals have unit amplitude.

>> [u,t] = gensig('square',10);

>> lsim(g,u,t)
Nonlinear Model Linearization

- `load_system('sys')` invisibly loads the Simulink model `sys`.
- `open('sys')` opens a Simulink system window for the Simulink model `sys`.
- `[x,u,y,dx]=trim('sys',x0,u0)` finds steady state parameters for the Simulink model `sys` by setting the initial starting guesses for `x` and `u` to `x0` and `u0`, respectively.
- `io=linio('blockname',portnum,type)` creates a linearization I/O object that has the type given by: 'in', linearization input point; 'out', linearization output point
- `lin = linearize('sys',io)` takes a Simulink model name 'sys' and an I/O object `io` as inputs and returns a linear state-space model `lin`. The linearization I/O object can be created with the function `linio`.
Biochemical Reactor Example

- Continuous bioreactor model

\[
\frac{dX}{dt} = -DX + \mu(S)X = f_1(X, S) \quad \mu(S) = \frac{\mu^\text{max} S}{K_S + S}
\]

\[
\frac{dS}{dt} = D(S_i - S) - \frac{1}{Y_{X/S}} \mu(S)X = f_2(X, S)
\]

- \( K_S = 1.2 \text{ g/L, } \mu^\text{max} = 0.48 \text{ h}^{-1}, Y_{X/S} = 0.4 \text{ g/g, } D = 0.15 \text{ h}^{-1}, S_i = 20 \text{ g/L} \)

```matlab
>> sys = 'bioreactor_stability';
>> load_system(sys);
>> open_system(sys);
>> [x1,u1,y1,dx1]=trim(sys,[1; 1],[]);
>> x1
```

\[
x1 = \\
7.7818 \\
0.5455
\]
function [sys,x0] = bioreactor_basic(t,x,u,flag) 
    Ks=1.2;
    Yxs=0.4;
    mumax=0.48;
    Si=20.0;
    D=u;
    switch flag,
    case 1,
        X=x(1);
        S=x(2);
        mu=mumax*S/(Ks+S);
        sys = [-D*X + mu*X; D*(Si-S)-mu*X/Yxs];
    case 3,
        X=x(1);
        Y=x(2);
        sys = X;
    case 0,
        NumContStates = 2;
        NumOutputs = 1;
        NumInputs = 1;
        sys = [NumContStates,0,NumOutputs,NumInputs,0,0];
        x0 = [7.78 0.545];
    case { 2, 4, 9 },
        sys = [];
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
    end
Linear Model Generation

```matlab
>> sys_io(1) = linio('bioreactor_stability/Dilution', 1, 'in');
>> sys_io(2) = linio('bioreactor_stability/Bioreactor', 1, 'out');
>> linsys = linearize(sys, sys_io)
```

```matlab
a =

Bioreactor(1)  Bioreactor(2)
Bioreactor(1)  -8.596e-005  1.472
Bioreactor(2)  -0.3748    -3.829
```

```matlab
b =

Dilution (pt
Bioreactor(1)  -7.78
Bioreactor(2)  19.45
```

```matlab
c =

Bioreactor(1)  Bioreactor(2)
bioreactor_s   1     0
```

```matlab
d =

Dilution (pt
bioreactor_s   0
```

```matlab
>> lambda = eig(linsys.a)
lambda =
-0.1500
-3.6793

>> g = tf(linsys)
-7.78 s - 1.16
----------------------
s^2 + 3.829 s + 0.5519

>> pole(g)
ans =
-3.6793
-0.1500

>> zero(g)
ans =
-0.1491

>> dcgain(g)
ans =
-2.1012
```