Table of Contents

Funktionskopf

```
% +----------------------------------------------------------------------+
% | File     : sprung_plot.m                                             |
% | Title    : Unterprogramm Typ: function                               |
% | Typ      : Source                                                    |
% | Version  : HolyMoly-4.2 Kleborn/ Wulf                                |
% | Drawn    : 08.04.2014                                                |
% | Checked  : 24.04.2014                                                |
% | Released : 24.04.2014                                                |
% | Language : MATLAB Version R2012b                                     |
% +----------------------------------------------------------------------+
% | Description : Initialisierung des Simulinkmodells sprung.mdl         |
% |               und ausführen. Ploten der Sprungantwort des Modells.   |
% |               Anlegen von Dokumentationselementen                    |
% |                                                                      |
% | Ausgabe :   1 Bin : 0 = Fehlerhafte Inputparameter                   |
% |                     1 = Fehlerhafte Flageingabe                      |
% |                     2 = Unbekanntes Flag oder Flagwert               |
% |                     3 = Zielpfad/ -datei nicht zugewiesen            |
% |                     4 = sprung.mdl nicht vorhanden                   |
% |                     5 = sprung_ws.mat nicht vorhanden                |
% |                     6 = sprung.mdl konnte nicht konfiguriert werden  |
% |                     7 = sprung.mdl konnte nicht ausgeführt werden    |
% |                     8 = LaTeX-String nicht erstellet                 |
% |                     9 = Plot fehlgeschlagen                          |
% |                    10 = Fehlerfrei durchgelaufen                     |
% |             2 Outputstruct mit Funktionsergebnissen                  |
% |                                                                      |
% | Eingabe :   1 Zählerpolynom des Modells als Zeilenvektor,            |
% |               Potenz abfallend                                       |
% |             2 Zählerpolynom des Modells als Zeilenvektor,            |
% |               Potenz abfallend                                       |
% |             4 Variable Argumente ['flag',wert,...,'flagn',wertn]     |
% |               Durchschaltkonstante    flag = 'D'     wert = double   |
% |               Modellzeitbasis         flag = 'Tbase' wert = double   |
% |               1e-4 <= Auflösung < 1   flag = 'Res'   wert = double   |
% |               Überschwingweite [%]    flag = 'Delta' wert = double   |
```

```
% |               Führungs-Ü-Fkt.         flag = 'Typ'   wert = 'G_W'    |
% |               Regler-Ü-Fkt.           flag = 'Typ'   wert = 'G_R'    |
% |               Strecken-Ü-Fkt.         flag = 'Typ'   wert = 'G_S'    |
% +-------------------------------------------------------------------------+
% | Call : [Outputparameter] = sprung_plot(Inputparameter)                  |
% |                                                                         |
% |                                                                         |
% | Outputparameter :   1 bin                                               |
% |                     2 outputs                                           |
% |                                                                         |
% | Inputparameter :    1 Z                                                 |
% |                     2 N                                                 |
% |                     3 varargin = [flagn,wertn]                          |
% +-------------------------------------------------------------------------+
% | Benötigte Dateien :                                                     |
% |                                                                         |
% | sprung.mdl, sprung_ws.mat                                               |
% +-------------------------------------------------------------------------+

function [ bin, outputs ] = sprung_plot(Z,N,varargin)
```

Parameter-Structs der Funktion anlegen

```
outputs.fcn.Z = 0;          % Zähler der Ü-Fkt.
outputs.fcn.N = 0;          % Nenner der Ü-Fkt.
outputs.fcn.D = 0;          % Durchsaltkonstante (optional)
outputs.mdl.Tbase = 10;     % Zeitbasis für Modell, z.B. t_max der Über-
                            % schwingweite oder die dominante Zeitkonstante
                            % des charackteristischen Polynoms
outputs.mdl.Res = 1e-2;     % Auflösung Fixed-Step / (StopTime - StartTime)
outputs.mdl.Config = 0;     % Simulink Konfigurationseinstellung
outputs.mdl.t = 0;          % Simulation-Output-Objekt (t,u,y,dy/dt)
outputs.mdl.u = 0;
outputs.mdl.y = 0;
outputs.mdl.dy = 0;
outputs.plt.Delta = 0;      % Überschwingweite, Entscheidungshilfe für die
                            % Plotdarstellung
outputs.plt.Typ = 'G';      % Standardplot falls nicht anders eingegeben
outputs.plt.fpath = '';     % Speicherot der Figure
outputs.plt.fname = '';     % Dateiname der Figure
outputs.plt.latex = '';     % LaTeX-Code zur Darstellung der Ü-Fkt.
```

Inputparameter überprüfen & Parmeter-Struct füllen (Inputs/ Flags)

- Validierung der festen Inputparameter

```
try
    validateattributes(Z,{'double'},{'row','real'},'sprung_plot','Z',1);
    validateattributes(N,{'double'},{'row','real'},'sprung_plot','N',2);
    outputs.fcn.Z = Z;
```

```
    outputs.fcn.N = N;
catch inputException
    bin = 0;
    disp(inputException.getReport());
    return;
end
```

- Validierung der variablen Flag-Inputparameter (varargin)

```
if nargin > 2
    % Prüfung ob immer Flag und Flagwert übergeben worden sind
    try
        validateattributes(varargin,{'cell'},{'row'},'sprung_plot','',3);
        if rem(length(varargin),2) ~= 0
            throw(MException('ValidFlags:UnbalancedFlag',...
                             'Flag ohne zugehörigen Wert eingegeben'));
        end
    catch flagException
        bin = 1;
        disp(flagException.getReport());
        return;
    end
    % Wandeln und Abspeichern der Flagwerte
    try
        for i=1:2:length(varargin)
            flag = cell2mat(varargin(i));
            wert = cell2mat(varargin(i+1));
            switch flag
                case 'D'
                    if isscalar(wert)
                        outputs.fcn.D = wert;
                    else
                        throw(MException('ValidFlag:WrongFlagValue',...
                                         '"D" muss Scalar sein!'));
                    end
                case 'Tbase'
                    if isscalar(wert)
                        outputs.mdl.Tbase = wert;
                    else
                        throw(MException('ValidFlag:WrongFlagValue',...
                                         '"TMax" muss Scalar sein!'));
                    end
                case 'Res'
                    if isscalar(and(wert >= 1e-4, wert < 1))
                        outputs.mdl.Res = wert;
                    else
                        throw(MException('ValidFlag:WrongFlagValue',...
                                         'Ausßerhalb 1e-4 <= Res < 1 !'));
                    end
                case 'Delta'
                    if isscalar(wert)
                        outputs.plt.Delta = wert;
                    else
                        throw(MException('ValidFlag:WrongFlagValue',...
```

```
                                        '"Delta" muss Scalar sein!'));
                    end
                case 'Typ'
                    if strcmp(wert,'G_S')
                        outputs.plt.Typ = wert;
                    elseif strcmp(wert,'G_W')
                        outputs.plt.Typ = wert;
                    elseif strcmp(wert,'G_R')
                        outputs.plt.Typ = wert;
                    else
                        throw(MException('ValidFlag:WrongFlagValue',...
                                         ['Unbekannter Wert "',...
                                          wert,...
                                          'für Flag "Typ"!']));
                    end
                otherwise
                    throw(MException('ValidFlag:UnknownFlag',...
                                     'Unbekanntes Flag aufgetreten'));
            end
        end
    catch flagUnknownException
        bin = 2;
        disp(flagUnknownException.getReport());
        return;
    end
end
% Ende der Inputparametervalidierung
```

Zielverzeichnis & Dateiname des Plots

Aktuelles Datum & Zeit, Double gezählt vom Januar 00-00-0000 Datumausgabe im gewünschten Format z.B. Apr, 07 2014 - 13:15:34 Uhr Aktueller Arbeitsordner als String

```
          date = now
          floor(now) % gibt Datum
          rem(now,1) % gibt Uhrzeit
          date = datestr(now,'mmm, dd yyyy - HH:MM:SS')
          path = pwd
          strcat(path,'\',date,' Uhr') %verbindet Strings

try
    outputs.plt.fpath = strcat(pwd,'\Plots-',datestr(floor(now),...
                                                     'yyyy-mmm-dd'));
    if exist(outputs.plt.fpath,'dir') ~= 7
        mkdir(outputs.plt.fpath);
    end
    outputs.plt.fname = strcat(outputs.plt.Typ,'-',datestr(now,...
                                                     'yyyymmdd-HH-MM-SS'),'h');
catch outputsException
        bin = 3;
        disp(outputsException.getReport());
        return;
end
```

```
% Ende der Zielzuweisung
```

Konfiguration & Ausführung des Simulinkmodells sprung.mdl

Ein zum Teil vorkonfiguriertes ConfigSet-Objekt befindet sich bereits im Modell sprung.mdl. Das Setup wird jetzt geladen, angepasst und mit den zusätzlich benötigten Variablen in sprung_ws abgespeichert. Damit ist im Modell-WS immer der letzte Durchlauf vorhanden. Das Vorgehen bietet somit auch schutz vor externen Aufrufen der sprung.mdl Datei, zudem bleibt der Base-Workspace sauber.

- Prüfen sprung.mdl im Arbeitsordner ist

```
try
    if exist('sprung.mdl','file') ~= 4
        bin = 4;
        throw(MException('ExistModel:NoModel',...
                         'Abbruch, sprung.mdl nicht vorhanden!'));
    end
catch existException
    disp(existException.getReport());
    return;
end
```

- Berechnen und laden der Modellparameter und -variablen

```
try
    % Benötiget Flagparameter in den Fkt.-WS zwischenspeichern
    D = outputs.fcn.D;            % Durschaltkonstante
    Tbase = outputs.mdl.Tbase;    % Zeitbasis
    Res = outputs.mdl.Res;        % Auflösung

    % Simulationsparameter aus Zeitbasis berechnen
    Tstart = 0.0;                 % Startzeit
    Tstop = ceil(10 * Tbase);     % Stopzeit
    Ts = Res * Tstop;             % Simulationsschrittweite
    Smp = (Tstop - Tstart) / Ts; % N Abtastwerte == length(mdlOut)

    % Modell laden und ConfigSet anpassen
    mdl = 'sprung';
    load_system(mdl);
    cs = getActiveConfigSet(mdl);
    mdl_cs = cs.copy;
    set_param(mdl_cs,'SolverMode','Auto',...
                     'Solver','ode4',...
                     'SolverType','Fixed-Step',...
                     'FixedStep',num2str(Ts),...
                     'StartTime',num2str(Tstart),...
                     'StopTime',num2str(Tstop),...
                     'Name','SprungConfig',...
                     'SaveTime','on',...
                     'TimeSaveName','t',...
                     'SaveOutput','on',...
```

```
                        'OutputSaveName','uydy',...
                        'LimitDataPoints','on',...
                        'MaxDataPoints',num2str(Smp),...
                        'SaveFormat','Array',...
                        'Decimation','1');
    outputs.mdl.Config = mdl_cs.copy;

    % Modellvariablen in den Model-WS speichern und laden
    mdl_ws = get_param(mdl,'modelworkspace');
    mdl_ws.DataSource = 'MAT-File';
    mdl_ws.FileName = 'sprung_ws';
    mdl_ws.assignin('D',D);
    mdl_ws.assignin('Z',Z);
    mdl_ws.assignin('N',N);
    mdl_ws.assignin('Tbase',Tbase);
    mdl_ws.saveToSource;
    mdl_ws.reload;

catch paramException
    bin = 6;
    disp(paramException.getReport());
    return;
end
```

- Ausführen der Simulation und Ergebnisse für den Plot speichern

```
try
    mdl_out = sim(mdl,mdl_cs);
    outputs.mdl.t = mdl_out.get('t');
    uydy = mdl_out.get('uydy');
    outputs.mdl.u = uydy(:,1);
    outputs.mdl.y = uydy(:,2);
    outputs.mdl.dy = uydy(:,3);
    close_system(mdl,1);
catch runModelException
    bin = 7;
    disp(runModelException.getReport());
    return;
end
% Ende der Konfiguration
```

LaTeX Outpung der Funktion und Plot der Simulation

- LaTex-String der Ü-Fkt. bauen

```
try
    % symbolische Variable anlegen
    % begrenzen des symbolischen Ausdrucks auf 6 Digits Dezimalzahlen ink.
    % Vorzeichen und Punkt
    syms('s');
    sD = vpa(D,6);
```

```
    sZ = vpa(poly2sym(Z,s),6);
    sN = vpa(poly2sym(N,s),6);
    f = vpa(sD + sZ / sN,6);
    % Digits für die Stringverarbeitung auf 6 begrenzen und danach wieder
    % auf 32 zurücksetzen
    d1 = digits(6);
    outputs.plt.latex = ['$',outputs.plt.Typ,'(s)=',latex(f),'$'];
    digits(d1);
catch latexParserException
    bin = 8;
    disp(latexParserException.getReport());
    return;
end
```

- Plot der Sprungantwort

```
try
    scrsz = get(0,'ScreenSize');
    shownFigure = figure('Position',...
        [1 scrsz(4)*0.2 scrsz(3)*0.8 scrsz(4)*0.6]);
    clf;
    hold on;
    plot(outputs.mdl.t,outputs.mdl.u,'b',... % Sprung
         outputs.mdl.t,outputs.mdl.y,'r');   % Systemantwort
    axis([outputs.mdl.t(1),max(outputs.mdl.t),...
          min(outputs.mdl.y)-0.1,1.2*max(outputs.mdl.y)]);
    grid on;
    legend('u(t)','y(t)');
    ylabel('Sprungantwort','FontSize',14);
    xlabel('Zeit [sec]','FontSize',14);
    title(outputs.plt.latex,'interpreter','latex','FontSize',16);
    % Speichern
    saveas(shownFigure,[outputs.plt.fpath,'\',...
                        outputs.plt.fname,'.fig'],'fig');
    hold off;
catch plotException
    bin = 9;
    disp(plotException.getReport());
    return;
end
```

- Anlyse der Sprungantwort, für den Fall das Typ = 'G_W' ist.

```
if strcmp(outputs.plt.Typ,'G_W')
    % Bandgrenze des Systems
    [fg,n] = max(outputs.mdl.dy);
    outputs.fcn.fg = fg;
    Tg = 1 / fg;
    % Wendepunkt
    tWP = outputs.mdl.t(n);
    yWP = outputs.mdl.y(n);
    % Sprungmarke
    for n=1:Smp-1
        if(outputs.mdl.u(n+1) > 10*outputs.mdl.u(n))
            tSP = outputs.mdl.t(n);
```

```
    end
end
% Verzugszeit
Tu = tWP - tSP - yWP / fg;
outputs.fcn.Tu = Tu;
% Plot der Wendetangent und Wendepunkt
shownFigure = figure('Position',...
    [1 scrsz(4)*0.2 scrsz(3)*0.8 scrsz(4)*0.6]);
clf;
hold on;
plot(outputs.mdl.t,outputs.mdl.u,'b',... % Sprung
     outputs.mdl.t,outputs.mdl.y,'r');   % Systemantwort
grid on;
legend('u(t)','y(t)');
ylabel('Sprungantwort','FontSize',14);
xlabel('Zeit [sec]','FontSize',14);
title('Ausschnitt 1 von $G_W(s)$','interpreter','latex','FontSize',16);
plot([tSP+Tu,tSP+Tu+Tg],[0,max(outputs.mdl.u)],'k');
plot(tWP,yWP,'g*');
text(tWP,yWP,['\leftarrow 1/T_g = ',num2str(fg),],...
    'FontSize',12);
text(tSP+Tu,0,['\leftarrow T_u = ',num2str(tSP+Tu),...
    ' - ',num2str(tSP),' = ',num2str(Tu)],...
    'FontSize',12);
% Zoomen
axis([tSP-Tu,tSP+Tg+2*Tu,-0.1,max(outputs.mdl.u)+0.1]);
% Speichern
saveas(shownFigure,[outputs.plt.fpath,'\',...
                    outputs.plt.fname,'-01.fig'],'fig');
hold off;
% Überschwingweite größer gleich 10[%], d.h. es gibt merklichen
% Unterschwinger und somit eine Einschwingperiode
if outputs.plt.Delta >= 10.0
    % Suche des 1. Sollwertdurchbruchs
    for n=1:Smp-1
        if outputs.mdl.y(n+1) >= max(outputs.mdl.u)
            tDP1 = outputs.mdl.t(n+1);
            break;
        end
    end
    % Ammplituden des ersten Über- Unterschwingers
    yf = outputs.mdl.y(n+1:Smp);
    A1 = max(yf); [A2,k] = min(yf);
    % Suche des 2. Sollwertdurchbruchs
    for n=n+k:Smp-1
        if outputs.mdl.y(n+1) >= max(outputs.mdl.u)
            tDP2 = outputs.mdl.t(n+1);
            break;
        end
    end
    % Überschwingerperiode
    Te = tDP2 - tDP1;
    outputs.fcn.Te = Te;
    % Plotten
```

```
            shownFigure = figure('Position',...
                [1 scrsz(4)*0.2 scrsz(3)*0.8 scrsz(4)*0.6]);
            clf;
            hold on;
            clf;
            plot(outputs.mdl.t,outputs.mdl.u,'b',... % Sprung
                 outputs.mdl.t,outputs.mdl.y,'r');   % Systemantwort
            grid on;
            legend('u(t)','y(t)');
            ylabel('Sprungantwort','FontSize',14);
            xlabel('Zeit [sec]','FontSize',14);
            title('Ausschnitt 2 von $G_W(s)$',...
                'interpreter','latex','FontSize',16);

            text((tDP1+tDP2)/2,A1,...
                ['T_e = ',num2str(Te)],'FontSize',12);
            axis([tDP1,tDP2,A2-0.1,A1+0.1]);
            % Speichern
            saveas(shownFigure,[outputs.plt.fpath,'\',...
                                outputs.plt.fname,'-02.fig'],'fig');
            hold off
        end
    end
    % Ende des Plotten
```

Ende der Funktion und Pass bin

- Pass-Bin setzen

```
bin = 10;

end
```

Beispiel 1 - Führungsübertragungsfunktion, annähernd aperiodisch

```
Tbase = 50;
Delta = 0.2;
Z_S = 0.0005;
N_S = [1 0.4 0.0 0.0];
[bin, Result.G_S] = sprung_plot(Z_S,N_S,'Typ','G_S')
Z_W = [0.0121];
N_W = [1 0.8694 0.1737 0.0121];
[bin, Result.G_W] = sprung_plot(Z_W,N_W,'Typ','G_W',...
                                       'Tbase',Tbase,...
                                       'Delta',Delta,...
                                       'Res',1e-4)
D_R = 24.2;
Z_R = [-11.3595 -4.2305];
N_R = [1 0.8694 0.1737];
[bin, Result.G_R] = sprung_plot(Z_R,N_R,'Res',1e-4,'Typ','G_R','D',D_R)
```

Beispiel 2 - Führungsübertragungsfunktion, sichtbares Überschwingen

```
t_max = 5;
Delta = 10.0;
Z_S = 16;
N_S = [625 500 150 20 1];
pole = roots(N_S);
domP = min(abs(pole));
if domP ~= 0.0
   domT = 1 / domP;
else
domT = 10.0
[bin, Result.G_S] = sprung_plot(Z_S,N_S,'Typ','G_S',...
                                        'Tbase',domT,...
                                        'Res', 1e-3)
Z_W = [3.386];
N_W = [1 5.5271 10.1848 7.8258 3.386];
[bin, Result.G_W] = sprung_plot(Z_W,N_W,'Typ','G_W',...
                                        'Tbase',t_max,...
                                        'Delta',Delta,...
                                        'Res',1e-3)
D_R = 132.25;
Z_R = [-625.154 -1315.1998 -1030.7301 0.2116];
N_R = [1 5.5271 10.1848 7.8258 0];
[bin, Result.G_R] = sprung_plot(Z_R,N_R,'Res',1e-3,...
                                        'Tbase',t_max,...
                                        'Typ','G_R',...
                                        'D',D_R)
```

Published with MATLAB® R2013b