Aufgaben (SySi)

1 Unterschiedliche Interpolationsverfahren

Füllen Sie einen Stützstellenvektor ***u*** mit den Werten 0 bis 10. ***colon***

1.1 Wieviele Elemente hat der Stützstellenvektor? ***length***

Berechnen Sie die Sinuswerte an den Stützstellen und weisen Sie die Werte dem Vektor ***y*** zu.

Definieren Sie einen 10 mal feiner unterteilten Stützstellenvektor ***uu***.

1.2 Wieviele Elemente hat ***uu***? Erst überlegen, dann überprüfen.

Berechnen Sie die Sinuswerte von ***uu*** und weisen Sie diese dem Vektor ***yy*** zu.

Interpolieren Sie linear zwischen den Punkten (***u***, ***y***) an den Zwischenstellen ***uu*** und weisen Sie die interpolierten Werte dem Vektor ***yl*** zu. ***interp1***

1.3 Wieviele Elemente hat ***yl***?

1.4 Ist ***yl*** ein Zeilen- oder ein Spaltenvektor? ***size***

Interpolieren Sie mit einem Polynom dritten Grades und weisen Sie die Zwischenwerte ***yk*** zu.

Legen Sie einen Spline über die Punkte (***u***, ***y***) und weisen Sie den Spline ***ys*** zu.

Zeichnen Sie alle Punkte und Kurven mit *einem* Zeichenbefehl gemeinsam in *ein* Fenster. ***plot***

Zeichnen Sie dabei:

- die (***u***, ***y***) Punkte mit blauen Sternchen,
- den fein abgetasteten Sinus als rote Kurve,
- die linear interpolierten Werte als türkisfarbene Kurve,
- die kubisch interpolierten Werte als grüne Kurve und
- den Spline als lila Kurve.

Schalten Sie die Vergrößerungmöglichkeit ein und nutzen Sie diese. ***zoom***

1.5 Bewerten Sie die verschiedenen Interpolationsverfahren.

1.6 Läßt sich diese Aussage verallgemeinern?

1.7 Wann kann eine lineare Interpolation sinnvoll sein?

Interpolation kann auch total schiefgehen:

1.8 Öffnen Sie ein neues Fenster, lassen Sie die Funktion $f(x) = 3x^2 + 1 + \frac{1}{\pi^4} \ln\left[(\pi - x)^2\right]$ im Intervall 0 bis 10 darstellen und schätzen Sie aus dem Verlauf der Kurve den Wert bei $x = \pi$. ***pause, figure, fplot***

1.9 Wie groß ist der wahre Wert bei π und wie stark müssen Sie das darzustellende Intervall einschränken, bis die Unstetigkeit auch im Kurvenverlauf sichtbar wird?

2 Zweidimensionale Interpolation

Zeichnen Sie die Peaks-Funktion über einem 25 x 25 Raster. ***peaks***

Fordern Sie ein neues Fenster an, in das das nächste Bild gezeichnet wird, um beide Bilder miteinander vergleichen zu können.

Füllen Sie die Matrix ***yp*** mit einer 7 x 7 Peakfunktion und zeichnen Sie ***yp***. ***surf***

2.1 Stellt die 7 x 7 Abtastung in Ihren Augen noch ein sinnvolles Abbild der Originalfunktion dar?

Füllen Sie die Stützstellenvektoren ***u1*** und ***u2*** mit ganzen Zahlen zwischen -3 und 3. Ganz wichtig: ***u2*** muß dabei ein Spaltenvektor sein. **'**

Definieren Sie ein feineres Stützstellenraster mit einem Abstand von 0.25 und weisen sie es den beiden Vektoren ***uu1*** und ***uu2*** zu, wobei ***uu2*** wieder ein Spaltenvektor sein soll.

Interpolieren Sie ***yp*** zweidimensional kubisch mit dem feineren Raster und weisen Sie das Ergebnis der Matrix ***yc*** zu. ***interp2***

Zeichnen Sie ***yc*** in ein neues Fenster. ***surf***

2.2 Vergleichen Sie das Interpolationsergebnis mit der 7 x 7 Peakfunktion und der Originalfunktion.

Verwenden Sie eine Nearest Neighbor-Interpolation und zeichnen Sie in ein neues Fenster.

2.3 Beschreiben Sie die Funktion der NN-Interpolation.

3 Regression (Best Fit)

Füllen Sie einen Eingangsvektor ***u*** mit 101 Werten zwischen 0 und 1.

Definieren Sie den Ausgangsvektor ***y*** so, dass er zusammen mit ***u*** eine Einheitsgerade beschreibt.

Erzeugen Sie einen Rauschvektor ***r*** mit einem normalverteilten weißen Rauschen mit einer Standardabweichung von 1 und einem Mittelwert von 0 mit der gleichen Länge wie ***y***. Verrauschen Sie ***y***, indem Sie ***r*** zu ***y*** addieren und nennen Sie den verrauschten Vektor ***yr***. ***randn***

Bestimmen Sie die Koeffizienten der Regressionsgerade (Polynom ersten Grades) durch die verrauschten Punkte und legen Sie die Koeffizienten der Geraden im Vektor ***p*** ab. ***polyfit***

Berechnen Sie die zum Eingangsvektor ***u*** gehörigen Geradenwerte. ***polyval***

Zeichnen Sie in ein gemeinsames Fenster

- die Originalgerade in rot,
- die verrauschten Meßpunkte als grüne Punkte und
- die angepaßte Gerade in blau.

Lassen Sie das Programm mehrfach ablaufen.

3.1 Warum kommen immer andere Ergebnisse?

Legen Sie um das gesamte Programm eine Schleife, so dass Sie die Anzahl der Punkte automatisch variieren lassen können. ***for***

Wählen Sie 11, 101, 1001, 10001 und 100001 Punkte.

Zeichnen Sie jedes Bild in ein neues Fenster.

3.2 Warum liegen die Geraden bei 11 Punkten so weit auseinander?

4 Schwinger zweiter Ordnung

Definieren Sie für ein P-T_2:

- die Dämpfung ***d*** zu $\frac{1}{\sqrt{2}}$,
- die Eigenfrequenz ***w0*** zu $\sqrt{2}$ und
- den stationären Verstärkungsfaktor ***k*** zu 2.

Bilden Sie das entsprechende Zählerpolynom ***num*** und das Nennerpolynom ***den***. Erzeugen Sie daraus das lineare zeitinvariante System ***sys_tf*** in Form einer Übertragungsfunktion. ***tf***

Berechnen Sie:

- die Nullstellen ***zer***,
- die Pole ***pol*** und
- die Verstärkung ***gain***. ***tf2zp***

4.1 Warum stimmt ***gain*** nicht mit ***k*** überein?

Erzeugen Sie das lineare zeitinvariante System ***sys_zp*** in Form einer Polnullstellendarstellung. ***zpk*** Transformieren Sie die Übertragungsfunktion (***num*** und ***den***) in den Zustandsraum (***amat***, ***bmat***, ***cmat*** und ***dmat***). ***tf2ss*** Erzeugen Sie das lineare zeitinvariante System ***sys_ss*** in Form einer Zustandsraumdarstellung. ***ss***

Zeichnen Sie die Polnullstellenverteilung der Übertragungsfunktion ***pzmap (sys_tf)*** und schreiben Sie Zähler- und Nennerpolynome als Text in das Bild. ***text***, ***mat2str***, ***[ ]***

Zeichnen Sie die Polnullstellenverteilung unter Verwendung von ***sys_zp*** und schreiben Sie Pole, Nullstellen und Verstärkung in das Bild.

4.2 Warum muss ***gain*** nicht an ***pzmap*** übergeben werden?

Zeichnen Sie die Polnullstellenverteilung der Zustandsraumdarstellung und schreiben Sie die Systemmatrizen in das Bild.

4.3 Stimmt die von MATLAB berechnete Zustandsraumdarstellung mit der aus der Vorlesung überein? Wer hat also einen Fehler gemacht? Skizzieren Sie sich das Blockschaltbild der MATLAB-Zustandsraumdarstellung und erläutern Sie die Unterschiede.

Zeichnen Sie die Sprungantwort der Übertragungsfunktion. ***step*** Lassen Sie MATLAB dabei entscheiden, wie lange Sie simulieren.

Zeichnen Sie die Sprungantwort der Zustandsraumdarstellung in das gleiche Bild. ***hold***

4.4 Wieviele Kurven sehen Sie und warum?

4.5 Was ist das für eine Gerade?

Zeichnen Sie in ein neues Bild die Sprungantwort über 8 Sekunden mit einem Abtastintervall von 2 Sekunden.

Zeichnen Sie in das gleiche Bild die Sprungantwort über 8 Sekunden mit einem Abtastintervall von 0.01 Sekunden.

4.6 Welche Simulation ist zu den gemeinsamen Zeitpunkten genauer?

4.7 Welche Vor- und Nachteile haben beide Simulationen?

Berechnen Sie Eigenwerte, Dämpfungen und Eigenfrequenzen der Zustandsraumdarstellung. ***damp***

4.8 Welche Matrix des Systems benötigt ***damp*** eigentlich nur und warum?

5 Identifikation im Zeitbereich

Machen Sie die noch zu definierende Variable ***verlauf*** global verfügbar, damit sie in der Kostenfunktion verwendet werden kann. ***global***

Definieren Sie wie in der 4. Aufgabe ein System zweiter Ordnung (***sys_tf***) mit

- Dämpfung ***d*** = $\frac{1}{\sqrt{2}}$,
- Eigenfrequenz ***w0*** = $\sqrt{2}$ und
- Verstärkungsfaktor ***k*** = 2.

Berechnen Sie die Sprungantwort des Systems im Zeitintervall zwischen 0 und 10 Sekunden mit einer Schrittweite von 0.1 Sekunden und legen Sie den Zeitverlauf in der globalen Variablen ***verlauf*** ab.

Definieren Sie einen Parametervektor ***par***, der während der Optimierung die drei identifizierten Parameter ***d***, ***w0*** und ***k*** enthält und geben Sie folgende Anfangswerte vor:

- Dämpfung: 1
- Eigenfrequenz: 1
- Verstärkungsfaktor: 1

Rufen Sie den Simplex-Optimierer auf. ***fminsearch*** Übergeben Sie ihm den Namen der Kostenfunktion ***'kosten'*** (s. u.) und den Anfangsparametervektor ***par***.

Weisen Sie das Identifizierungsergebnis dem Vektor ***erg*** zu und geben Sie die identifizierten Parameter einzeln mit einer entsprechenden Bezeichnung aus.

Schreiben Sie jetzt in die Datei ***kosten.m*** die entsprechende Kostenfunktion ***kosten***, die als Eingangsparameter den Vektor ***par*** erhält und die den Kostenwert ***fehler*** zurückgibt. ***function***

Definieren Sie wieder die Variable ***verlauf*** als global.

Ziehen Sie die einzelnen Variablen aus dem Parametervektor ***par*** heraus und weisen Sie sie den Modell-Variablen ***d_m***, ***w0_m*** und ***k_m*** zu.

Definieren Sie damit ein Modell-System zweiter Ordnung (***sys_tf_m***).

Berechnen Sie die Sprungantwort des Modells (***verlauf_m***).

Bilden Sie den quadratischen Fehler des Sprungantwortenverlaufs (Wurzel aus der Quadratsumme der Amplitudendifferenzen) ***norm*** und weisen Sie ihn der Variablen ***fehler*** zu.

6 P-T_2 Simulation

Führen Sie unter MATLAB die 4. Aufgabe aus. So können Sie dann später unter SIMULINK direkt auf die jetzt definierten Variablen (***d***, ***w0***, ***k***, ***num***, ***den***, ...) zugreifen.

Öffnen Sie unter MATLAB die SIMULINK-Bibliothek mit den Standard-Blöcken. ***simulink*** und erstellen Sie ein neues Modell. ***File New Model***

Speichern Sie das neue, leere SIMULINK-Fenster, das momentan noch ***untitled*** heißt, unter ***loes6_mdl.mdl*** ab. ***File Save As***

Sie sollten, während Sie das Modell vervollständigen, immer mal wieder abspeichern ***File Save***, da auch SIMULINK leider bisweilen abstürzt und Sie bei einem Neustart immer nur Ihre letzte gespeicherte Version laden können.

Ziehen Sie aus den entsprechenden SIMULINK-Unterfenstern die folgenden Blöcke in Ihr neues ***loes6_mdl.mdl*** -Fenster:

- ***Sources, Signal Generator***
- ***Sources, Clock***
- ***Signals & Systems, Mux***
- ***Sinks, Scope***
- ***Sinks, To Workspace***
- ***Continuous, Transfer Fcn***
- ***Continuos, Zero-Pole***
- ***Continuous, State-Space***
- ***Continuous, Integrator***
- ***Math, Gain***
- ***Math, Sum***

Doppelklicken Sie jeden Block mit der linken Maustaste und machen Sie sich mit seiner Funktion vertraut.

Verbinden Sie den ***Signal Generator*** jeweils mit den Blöcken ***Transfer Fcn***, ***Zero-Pole*** und ***State-Space***.

Lernen Sie dabei:

- Verbindungslinien mit der linken Maustaste zu ziehen,
- eine neue Verbindungslinie an eine bestehende mit der rechten Maustaste anzuknüpfen,
- Blöcke mit der linken Maustaste zu verschieben,
- einen Block mit der linken Maustaste und der SHIFT-Taste von seinen Linien zu lösen,
- Blöcke zu kopieren ***Edit Copy Edit Paste*** oder rechte Maustaste (gedrückt ziehen),
- Linien und Blöcke zu löschen ***Edit Clear*** oder DELETE-Taste,
- die Größe der Blöcke mit der linken Maustaste zu verändern.

Bauen Sie aus ***Integrator***, ***Gain*** und ***Sum*** das Blockschaltbild eines Schwingers 2. Ordnung auf (Vorlesung). Spiegeln Sie dabei die ***Gain***-Blöcke in der Rückführung. ***Format Flip Block*** Erkennen Sie, dass SIMULINK automatisch neue, eindeutige Namen für verdoppelte Blöcke vergibt.

Legen Sie eine weitere Leitung vom ***Signal Generator*** zum Eingang des P-T_2.

Öffnen Sie den ***Mux*** und setzen Sie die Anzahl seiner Eingänge auf fünf.

Verbinden Sie die Ausgänge von

- ***Signal Generator***,
- ***Transfer Fcn***,
- ***Zero-Pole***,
- ***State-Space*** und
- dem neu entstandenen P-T_2

mit den Eingängen des ***Mux***.

Schließen Sie das ***Scope*** und den ***To Workspace*** an den ***Mux*** an. Öffnen Sie ***To Workspace*** und geben Sie als Variablenname ***y*** ein.

Verbinden Sie die ***Clock*** mit einem zweiten ***To Workspace*** und geben als Variablenname ***t*** ein.

Öffnen Sie den ***Signal Generator*** und stellen Sie

- Rechtecke ***square*** mit einer
- Frequenz von 0.05 Hz und einer
- Amplitude von 1

ein.

Öffnen Sie den ***Transfer Fcn***-Block und geben Sie als

- Zählervektor ***num*** und als

- Nennervektor ***den***

ein. SIMULINK holt sich die Vektoren dann direkt von MATLAB.

Analog definieren Sie die Vektoren und Matrizen von

- ***Zero-Pole*** und
- ***State-Space***.

Auch im P-T_2 geben Sie in den ***Gain***-Blöcken direkt die entsprechenden MATLAB-Ausdrücke, beispielsweise ***2*d*w0***, an. Vergrößern Sie die Blöcke, bis ihr „Aufdruck" vollständig zu sehen ist.

Öffnen Sie das ***Scope***, klicken Sie den Parameterknopf (zweiter von rechts) und geben Sie auf der Achsen-Karte ***Axes*** die folgenden Parameter ein:

- ***Y max:*** 3
- ***Y min:*** -3
- ***Time Range:*** 20

Wählen Sie im Simulationsmenü ***Simulation Parameters Solver*** die folgenden Werte:

- ***Start time:*** 0
- ***Stop time:*** 20
- ***Type:*** Fixed-step, ode4 (Runge-Kutta)
- ***Fixed step size:*** 1

Starten Sie die Simulation ***Simulation Start*** und sehen Sie sich das Simulationsergebnis auf dem ***Scope*** an.

Wechseln Sie zurück in das MATLAB-Fenster und zeichnen Sie dort den von SIMULINK an MATLAB übergebenen Vektor ***y*** über der Zeit ***t*** als Einzelpunkte.

6.1 Welche Vor- und Nachteile haben ***Scope*** und ***plot***?

6.2 Warum sind die Flanken des Rechtecksignals nicht senkrecht und warum sieht die Sprungantwort so zackig aus?

Erniedrigen Sie die feste Schrittweite auf 0.01 und simulieren und plotten Sie erneut.

6.3 Besser? Welche Nachteile hat es, die Schrittweite immer weiter zu verkleinern? Was schlagen Sie statt dessen vor?

Erlauben Sie SIMULINK im Simulationsmenü, mit variabler Schrittweite zu integrieren:

- ***Type:*** Variable-step, ode45 (Dormand-Prince)
- ***Max step size:*** auto
- ***Min step size:*** auto

- ***Initial step size:*** auto
- ***Relative tolerance:*** 1e-6
- ***Absolute tolerance:*** 1e-6

Zeichnen Sie wieder unter MATLAB das Simulationsergebnis als Einzelpunkte (***y*** über ***t***).

6.4 Nach welchen Kriterien verändert SIMULINK die Schrittweite während der Simulation?

7 Gruppieren und Maskieren

Markieren Sie im Blockschaltbild der 6. Aufgabe den aus Einzelblöcken aufgebauten Schwinger 2. Ordnung. Achten Sie dabei darauf, dass Sie auch die Verbindungslinie vom Ausgang des P-T_2 zum ***Mux*** mitmarkieren, damit später klar ist, welches der Ausgang des Systems ist.

Kopieren Sie das P-T_2 in ein neues Fenster.

Markieren Sie alle Einzelblöcke des P-T_2 mit der linken Maustaste und gruppieren Sie sie in einen einzigen Subsystem-Block. ***Edit Create subsystem***

Maskieren Sie den Subsystem-Block. ***Edit Mask subsystem***

In die sich öffnende Dialogbox tragen Sie folgende Angaben ein:

- ***Icon Mask type:*** <p>Mein P-T2
- ***Icon Drawing commands:*** plot(step([k*w0^2],[1, 2*d*w0, w0^2]))
- ***Initialization Prompt:*** Eigenkreisfrequenz
- ***Initialization Variable:*** w0
- ***Initialization Add***
- ***Initialization Prompt:*** Dämpfungsfaktor
- ***Initialization Variable:*** d
- ***Initialization Add***
- ***Initialization Prompt:*** Verstärkungsfaktor
- ***Initialization Variable:*** k
- ***Documentation Block description:*** Schwinger 2. Ordnung
- ***Documentation Block help:*** Linearer Schwinger zweiter Ordnung. <p>Geben Sie Dämpfung, Eigenfrequenz und Verstärkung vor.

Doppelklicken Sie den jetzt maskierten Subsystem-Block und tragen Sie in die Dialogbox drei sinnvolle Werte für Dämpfung, Eigenfrequenz und Verstärkung ein.

7.1 Wer hat den Inhalt dieser Dialogbox definiert?

Rufen Sie ***Help*** in der Subsystems-Dialogbox auf.

7.2 Woher kommt der Help-Text und wie können Sie ihn nachträglich verändern?

7.3 Welche Aufgabe hat ***<p>***?

7.4 Was bewirken die ***Drawing commands***?

7.5 Ändert sich die Block-Ikone, wenn Sie die Dämpfung des P-T_2 variieren?

Ändern Sie den Blocknamen von ***Subsystem*** in ***P-T2***.

Überprüfen Sie die Funktion des P-T_2, indem Sie einen ***Signal Generator*** und ein ***Scope*** spendieren und für verschiedene Parameter simulieren.

Erzeugen Sie eine persönliche Bibliothek ***File New Library***, kopieren Sie das fertige P-T_2 als wiederverwendbaren Block in Ihre neue Bibliothek und speichern Sie diese unter dem Namen ***my_lib.mdl*** ab. ***save as***

7.6 Welchen Vorteil hat eine Bibliothek gegenüber einer einfachen Ansammlung von SIMULINK-Blöcken?

8 Mathematisches Pendel

Modellieren Sie die lineare Differentialgleichung eines mathematischen Pendels (Vorlesung) ohne Anregung mit zwei Integratoren und einem Verstärkungsfaktor (Vorzeichen beachten). Die Länge des Pendels sei dabei gerade 9.81 m.

Kopieren Sie das gesamte Modell innerhalb des Fensters und ergänzen Sie in der Kopie die lineare Rückführung durch einen Sinus-Funktions-Block aus der SIMULINK-Bibliothek, so dass ein nichtlineares Pendel entsteht. ***Math Trigonometric Function***

Vereinigen Sie beide Pendelausgänge in einem ***Mux***, an den Sie ein ***Scope*** anschließen.

Setzen Sie die Simulationsparameter auf ode45 und 20 s Simulationsdauer und automatische Schrittweitenkontrolle.

8.1 Was sehen Sie jetzt, wenn Sie die Simulation starten? Und warum nicht?

Initialisieren Sie die beiden Winkelintegratoren auf ***Integrator Initial condition:*** pi/10, lassen Sie die Winkelgeschwindigkeitsintegratoren auf null und simulieren Sie.

8.2 Was bedeutet dieses Initialisieren physikalisch?

8.3 Können Sie einen Unterschied zwischen den beiden Kurven feststellen?

8.4 Halten Sie die lineare Näherung bei diesem Winkel noch für vertretbar?

Wenn Sie möchten, können Sie den Kurvenverlauf glätten, indem Sie den ***Simulation Parameters Solver Output options Refine Factor*** z. B. auf 10 erhöhen.

Setzen Sie die Anfangswerte der Winkel auf ***pi/2***.

8.5 Wie gut finden Sie jetzt die lineare Näherung?

8.6 Wann ist eine lineare Näherung sinnvoll?

8.7 Welche der beiden Kurven gehört zum nichtlinearen Pendel und warum?

Setzen Sie die Anfangswerte der Winkel auf ***pi***.

8.8 Erklären Sie das interessante Simulationsergebnis physikalisch. Wie bewegt sich das Pendel während der Simulation?

Stören Sie jetzt das nichtlinearen Pendel geringfügig aus seiner labilen Ruhelage, indem Sie als Anfangswinkel ***pi - 0.01*** angeben.

8.9 Beschreiben Sie sein Verhalten mit Worten.

8.10 Und bei einem Anfangswinkel von ***3/2*pi***? Stellen Sie sich die Bewegung der Pendel mal anschaulich vor und erklären Sie.

Setzen Sie die Anfangswinkel auf null zurück und initialisieren Sie die Geschwindigkeitsintegratoren mit 1 rad/s.

8.11 Deuten Sie physikalisch.

8.12 Was erwarten Sie, wenn Sie jetzt (Anfangsgeschwindigkeit 1 rad/s) die Anfangswinkel auf ***pi*** setzen?

8.13 Und – hatten Sie Recht?

8.14 Noch mal die Quintessenz dieser Übung: Wann macht eine Linearisierung Sinn?

9 Trimmrechnung

Kopieren Sie das nichtlineare Pendel der 8. Aufgabe in ein neues Fenster. Wirklich nur das Pendel (zwei Integratoren, eine Funktion und ein negatives Vorzeichen).

Öffnen Sie die Verbindung von der ***-sin(u)***-Rückführung zum Eingang des Winkelgeschwindigkeitsintegrators und fügen Sie dort eine ***Sum*** ein.

Schließen Sie an den Summierpunkt sowohl die ***-sin(u)***-Rückführung, als auch einen neuen Blockeingang ***Signals & Systems In***-Port an.

Spendieren Sie einen Umrechnungsblock, der den Pendelwinkel von Radian in Grad umrechnet und leiten Sie das Ergebnis an einen Blockausgang ***Signals & Systems Out***-Port weiter. Hängen Sie ein ***Scope*** parallel zu dem ***Out***-Port.

Wechseln Sie in das MATLAB-Fenster zurück und trimmen Sie das Pendel statisch auf einen Winkel von 45 Grad. Der dazu nötige Befehl lautet: ***[x_trim, u_trim, y_trim, dx_trim] = trim ('loes9_mdl', [], [], [45], [], [], [1])***.

9.1 Halten Sie das Ergebnis für plausibel? Bilden Sie die Summe am Eingang des Winkelgeschwindigkeitsintegrators.

Lesen Sie die Syntax des Trimmbefehls unter ***help trim*** nach und vollziehen Sie jeden Parameter des obigen Befehls nach.

9.2 Wo steckt die zweite Trimmforderung, dass das Pendel auf 45 Grad stehenbleiben und nicht losschwingen soll?

Erzeugen Sie eine Simulationsoptionsstruktur, die den ausgetrimmten Zustandsvektor als Anfangswert beinhaltet: ***simopt = simset ('initialstate', x_trim)***, simulieren Sie von der MATLAB-Oberfläche aus: ***sim ('loes9_mdl', 20, simopt)*** und lesen Sie die Syntax der beiden Befehle nach. ***help simset, help sim***

9.3 Warum ergibt sich nicht der ausgetrimmte statische Zustand, sondern eine Schwingung?

Übergeben Sie auch den Eingangsgrößentrimmzustand direkt im Simulationsaufruf, indem Sie die Parameterliste erweitern: ***sim ('loes9_mdl', 20, simopt, 'u_trim')***.

9.4 Und - bleibt der Winkel bei 45 Grad?

Eine sinnvolle Alternative, den Trimmpunkt zu übergeben, ist das Vorsehen des Eingangsgrößentrimmzustands direkt im Blockschaltbild. Erweitern Sie dazu im Blockschaltbild den ***Sum***-Block auf drei Eingänge und spendieren Sie einen ***Constant***-Block, der auf die ***Sum*** wirkt. Diese ***Constant*** übernimmt jetzt den Trimmwert der Eingangsgröße, indem Sie ihr den ***Constant Value u_trim*** zuweisen.

Wenn Sie nun getrimmt haben, so dass ***u_trim*** definiert ist, können Sie direkt mit ***sim ('loes9_mdl', 20, simopt)*** simulieren.

Natürlich hat auch diese Methode einen kleinen Nachteil. Wenn Sie erneut trimmen wollen, hat die Konstante im Blockschaltbild schon den Wert ***u_trim*** der letzten Trimmrechnung, der dann während der aktuellen Trimmrechnung dazu führt, dass als neuer (zusätzlicher) Trimmwert ***u_trim*** nichts mehr nötig ist. Das neue ***u_trim*** wird folgerichtig zu null berechnet. Probieren Sie es aus. Bei der nachfolgenden Simulation wird dann aber dieses „falsche“ ***u_trim*** von null verwendet und führt zu einem nicht ausgetrimmten Anfangsszustand.

Fazit: Setzen Sie ***u_trim = 0***, bevor Sie trimmen.

Trimmen Sie ein senkrecht stehendes Pendel aus und simulieren Sie. Haben Sie auch den neuen Zustandsanfangsvektor in ***simopt*** aktualisiert?

9.5 Bleibt der Winkel bei 180 Grad?

Übergeben Sie einen minimal gestörten Anfangswert von ***x_trim + 1e-12*** und simulieren Sie 100 Sekunden lang.

9.6 Und?

10 Linearisierung

Öffnen Sie die Lösung der 9. Aufgabe (mathematisches Pendel).

Um dieses nichtlineare System um die Ruhelage (Winkel gleich null) herum zu linearisieren, geben Sie unter MATLAB den folgenden Befehl ein: ***[a, b, c, d] = linmod ('loes9_mdl')***.

10.1 Skizzieren Sie das Blockschaltbild des sich ergebenden Systems und überprüfen Sie seine Ähnlichkeit mit dem linearen mathematischen Pendel.

Linearisieren Sie um einen Winkel von 90 Grad herum, indem Sie den ***linmod***-Befehl um diesen Arbeitspunkt erweitern. ***help linmod*** Vorher müssen Sie natürlich ***x_trim*** an diesem Arbeitspunkt bestimmen (vgl. 9. Aufgabe).

10.2 Worin unterscheidet sich das Blockschaltbild des linearisierten Systems um 90 Grad herum von dem um 0 Grad?

10.3 Erläutern Sie physikalisch, um welche Art von Bewegung es sich bei dem um 90 Grad herum linearisierten System (im ersten Moment) handelt.

10.4 Vergleichen Sie das numerische Linearisierungsergebnis hier mit der Linearisierung über die Taylorreihe (Vorlesung).

11 Integrationsverfahren

Für die folgenden ersten Teilaufgaben soll das Euler-Verfahren mit einer konstanten großen Schrittweite von genau einer Sekunde verwendet werden. Stellen Sie die entsprechenden Simulationsparameter in einem neuen SIMULINK-Fenster ein.

Simulieren Sie die Sprungantwort eines kontinuierlichen Systems erster Ordnung mit einer Einheitsverstärkung und einer Zeitkonstanten von 10 Sekunden.

11.1 Macht sich das recht große Abtastintervall von einer Sekunde nachteilig bemerkbar?

11.2 Wo liegt der Pol des Systems qualitativ im Stabilitätsgebiet des Eulerverfahrens?

Simulieren Sie ein P-T_1 mit einer Zeitkonstanten von einer Sekunde.

11.3 Wo liegt dann der Pol und was bedeutet das für die Stabilität der Integration?

11.4 Wie gefällt Ihnen das Simulationsergebnis?

Stellen Sie die Zeitkonstante des P-T_1 so ein, dass sein Pol genau auf der „linken" Stabilitätsgrenze des Eulerverfahrens liegt.

11.5 Wie groß ist dann die Zeitkonstante?

11.6 Welches Simulationsergebnis erhalten Sie (Sinus, Dreieck, Rechteck, ...)?

Versuchen Sie, eine langsam aufklingende Schwingung zu erhalten, indem Sie den Pol des P-T_1 gerade eben in den instabilen Bereich von Euler schieben.

11.7 Wie groß haben Sie dazu die Zeitkonstante gewählt und wie lange mußten Sie simulieren?

Simulieren Sie einen reinen Integrator.

11.8 Und - klappt das?

11.9 Wo liegt der Integrator in Eulers Stabilitätsgebiet?

11.10 Versuchen Sie eine allgemeine Aussage über den Zusammenhang zwischen der Lage des Pols im Stabilitätsgebiet und der „Güte" der Simulation zu treffen. Werden Pole mitten im Stabilitätsgebiet besonders gut simuliert und Pole auf dem Stabilitätsrand besonders schlecht?

Simulieren Sie einen ungedämpften Schwinger zweiter Ordnung mit einer Eigenfrequenz von 1 rad/s mit Euler.

11.11 Wo liegen die Pole und warum wird das P-T_2 nicht vernünftig simuliert?

Probieren Sie's mit einer kleineren Eigenfrequenz von 0.1 rad/s.

11.12 Besser? Wo liegen jetzt die Pole im Vergleich zum schnelleren P-T_2?

11.13 Wie klein muß man die Eigenfrequenz eines mit Euler simulierten ungedämpften P-T_2 machen, damit es stabil simuliert wird?

11.14 Was halten Sie von Euler?

Simulieren Sie das ungedämpfte P-T_2 mit der Eigenfrequenz von 1 rad/s mit Runge-Kutta 4. Ordnung.

11.15 Wie erklären Sie sich dieses interessante Ergebnis?

11.16 Wie simuliert RK4 das P-T_2 mit einer Eigenfrequenz von 0.1 rad/s über einen Zeitraum von 10000 Sekunden?

11.17 Kann das Verfahren fünfter Ordnung (Dormand-Prince) das P-T_2 mit der Eigenfrequenz von 1 rad/s 10000 Sekunden lang stabil simulieren?

12 Steife Systeme

Modellieren Sie ein steifes System bestehend aus einer Reihenschaltung zweier P-T_1 mit Einheitsverstärkung und den beiden Zeitkonstanten $T_1 = 1$ Sekunden und $T_2 = 100$ Sekunden. Simulieren Sie eine Sprungantwort 1000 Sekunden lang mit einer festen Schrittweite von 1 Sekunde und benutzen Sie nacheinander alle Integrationsverfahren (Euler, ..., Dormand-Prince).

12.1 Können Sie einen signifikanten Unterschied zwischen den verschiedenen Verfahren feststellen?

Erhöhen Sie jetzt die Schrittweite auf 2 Sekunden und simulieren Sie wieder mit allen Verfahren.

12.2 Wo liegt der schnelle Pol jetzt im Stabilitätsgebiet des Euler-Verfahrens? Können Sie die beginnende Instabilität erkennen? Wie verhalten sich die anderen Verfahren?

Legen Sie den Pol gerade außerhalb Eulers Stabilitätsgebietes, indem Sie das Abtastintervall geringfügig auf 2.01 Sekunden erhöhen.

12.3 Wie reagiert Euler und was machen die anderen Verfahren?

12.4 Stellen Sie fest, bei welchen Schrittweiten bzw. Simulationszeiten die übrigen Verfahren aussteigen.

13 Begrenzungen

Modellieren Sie ein lineares System erster Ordnung aus einem Integrator, zwei Verstärkern und einem Summierer. Basteln Sie sich aus zwei ***Step***s und einem Summierer einen Impulsgeber, der einmalig von null auf eins und nach einer festgelegten Zeit wieder auf null zurückspringt.

Stellen Sie alle Zeitkonstanten und Verstärkungen so ein, dass das P-T_1 nach jeder Flanke praktisch stationär wird und sehen Sie sich das Ergebnis auf einem ***Scope*** an. Plotten Sie dabei sowohl das Eingangs- als auch das Ausgangssignal des P-T_1.

Stellen Sie sich jetzt vor, das P-T_1 sei das Modell eines Stellmotors, dessen Ausgangsgröße (Weg oder Winkel) mechanisch begrenzt ist. Schließen Sie einen ***Saturation***-Block (aus der ***Nonlinear***-Gruppe) an den Ausgang des P-T_1 an und begrenzen Sie auf einen sinnvollen Wert. Plotten Sie den begrenzten Wert *zusätzlich* auf dem ***Scope***.

13.1 Wieviel Zeit vergeht bei einem *realen* Steller, der in die Sättigung gegangen ist, nach der Rücknahme des Eingangssignals, bis der Steller anfängt, zurückzulaufen?

13.2 Wie verhält sich Ihr Stellermodell mit der reinen Ausgangsgrößenbegrenzung hinsichtlich obiger Frage?

13.3 Erklären Sie, wie es zu dieser Totzeit kommt.

Überlegen Sie sich jetzt, wie Sie das Problem mit der Totzeit in den Griff bekommen können und welche Blöcke Sie von SIMULINK gern hätten, um Ihre Lösung zu realisieren.

13.4 Welche Blöcke bräuchten Sie, um die Totzeit zu eliminieren?

Freundlicherweise erlaubt Ihnen SIMULINK, einen ***Integrator*** auch direkt zu begrenzen (auch andere Ingenieure haben sich diese Gedanken schon einmal gemacht ...). Kopieren Sie das P-T_1, öffnen Sie im kopierten Tiefpass den ***Integrator***, begrenzen Sie ihn auf die gleichen Werte wie im Saturation-Block des ersten Tiefpass und stellen Sie sein Ergebnis mit auf dem ***Scope*** dar.

13.5 Zufrieden?

Sie wollen jetzt zusätzlich zu der Positionsbegrenzung eine Geschwindigkeitsbegrenzung (Ratenlimitierung) in Ihr Stellermodell erster Ordnung einbauen, da der Steller aufgrund von Druck- und Durchflußbeschränkungen im Hydrauliksystem stationär nur mit einer bestimmten maximalen Geschwindigkeit laufen kann. Überlegen Sie sich, an welcher Stelle im P-T_1 die Geschwindigkeit auftritt, wenn die Ausgangsgröße die Position darstellt.

Kopieren Sie das P-T_1 ein drittes Mal, begrenzen Sie in der Kopie die Geschwindigkeit mit Hilfe eines ***Saturation***-Blockes auf einen plausiben Wert und überprüfen Sie das Ergebnis im ***Scope***.

13.6 Mit welcher Kurvenform verändert sich die Position, während sich der Steller in der Ratenbegrenzung befindet?

SIMULINK stellt Ihnen in der ***Nonlinear***-Gruppe einen Ratenbegrenzungsblock (***Rate Limiter***) zur Verfügung. Modellieren Sie ein weiteres ratenbegrenztes P-T_1, indem Sie den expliziten ***Rate Limiter*** vor den Eingang eines in der Geschwindigkeit nicht begrenzten P-T_1 setzen.

Vergleichen Sie das Simulationsergebnis mit dem vorher durch direkte Begrenzung der Geschwindigkeit generierten Signal.

Setzen Sie in einem fünften Modell den ***Rate Limiter*** hinter das P-T_1.

13.7 Warum sind die Zeitverläufe nicht identisch? Kann man denn in einem linearen System die Blöcke nicht beliebig in ihrer Reihenfolge vertauschen?

13.8 Welche Modellierung der Geschwindigkeitsbegrenzung eines Stellers halten Sie warum für am realistischsten?

13.9 Welchen Nachteil hat diese Art der Ratenlimitierung hinsichtlich der Modularität einzelner Blöcke?

14 Steuerstrang

Sie wollen einen kompletten Fly-by-wire-Steuerstrang eines Flugzeugs vom Stick des Piloten bis zum Ruder am anderen Ende des Fliegers simulieren. Sie haben daher die folgenden Blöcke in Reihe zu schalten:

- Stick
- Meßgeber
- Stickshaping-Kennlinie
- Fly-by-wire-Rechner
- Booster
- Hydraulik
- Lose
- Turbulenz
- Anschlag

Bevor Sie die einzelnen Blöcke zu einer Wirkungskette zusammenschalten, sollten Sie jeden Block für sich auf seine Funktion hin überprüfen. Schaffen Sie sich dazu ein kleines Testlabor, bestehend aus einem Eingabemedium (Maus, Signalgenerator, ...) dem Prüfling und einem Mehrkanaloszilloskop (***Mux*** und ***Scope***), mit dem Sie die Signale vor und hinter dem Probanten kontrollieren. (Sie können natürlich auch Kreuzkorrelationen, Leistungsdichteanalysen, ... mit den entsprechenden Blöcken durchführen.)

Stick

Sehen Sie zwei Möglichkeiten vor, Eingangssignale zu erzeugen: Entweder mit der Maus (***Mouse*** aus der Bibliothek ***jj_lib***; kopieren Sie dazu ***jj_lib.mdl*** und ***mouse_coordinates.m*** in Ihr Arbeitsverzeichnis), um als Pilot-in-the-loop willkürliche Kommandos eingeben zu können oder mittels eines ***Signal Generator***s. Beide Signalquellen sollen mit einem ***Manual Switch*** auswählbar gemacht werden. Schließen Sie den nicht benötigten Ausgang der ***Mouse*** mit einem ***Terminator*** ab.

Meßgeber

Der elektrische Meßgeber sei annähernd linear und habe leider ein nicht zu vernachlässigendes Meßrauschen. Addieren Sie daher zum ausgewählten Signal ein signifikantes Meßrauschen. ***Band-Limited White Noise***

Grundsätzlich sollten Sie bei der Wahl der Parameter aller Blöcke darauf achten, dass der Effekt des Blockes zwar sichtbar wird, aber nicht so groß ist, dass er andere Effekte völlig überdeckt. Idealerweise sollten im Ausgangssignal die Einflüsse möglichst aller Blöcke noch zu erkennen sein.

Stickshaping-Kennlinie

Piloten mögen Sticks, die mit wachsendem Ausschlag eine größere Verstärkung aufweisen. So sind die Flugzeugreaktionen bei kleinen Winkeln nicht zu heftig und bei großen Kommandos kommt es dem Piloten sowieso nicht mehr auf gefühlvolles Taktieren an. Modellieren Sie also eine Stickshaping-Kennlinie, die sich für positive und negative Ausschläge jeweils so aus zwei Geradenstücken zusammensetzt, dass obige Forderung erfüllt ist. ***Look Up Table***

Des weiteren haben mechanische Sticks die unangenehme Eigenschaft, dass der Punkt, zu dem die Federn den Stick zurücksetzen, wenn ihn der Pilot losläßt, leider nicht immer exakt der gleiche ist. Das Signal, dass der Stick dann in seiner Ruhelage liefert, läßt sich daher nicht auf genau null kalibrieren. Um diesen theoretischen Nullpunkt herum ist daher eine tote Zone nötig, in der der Stick bei Minimalausschlägen trotzdem noch ein Signal von null liefert. Sie können diese tote Zone entweder explizit kraft eines ***Dead Zone***-Blockes oder implizit in die Kennlinie einbauen. Die Kennlinie muß dann 8 Stützstellen haben.

Fly-by-wire-Rechner

Der Rechner hat die Aufgabe, beliebig komplexe Regelgesetze auf seine Eingangsgröße(n) anzuwenden, um so Steuerkommandos für die Steller zu erzeugen. In Ihren Fall wollen Sie ein 1:1-Regelgesetz realisieren; Sie wollen dem Piloten also direkt das Ruder an seinen Stick hängen. Dabei soll ein Winkelinkrement Stickausschlag einem Winkelinkrement des Ruders entsprechen. Leider haben Digitalrechner die unangenehme Eigenschaft, dass sie Signale aufgrund von A/D- und D/A-Wandlungen, Buslaufzeiten, Kommunikationsoverhead, ... verzögern, so dass Sie als die wesentliche Eigenschaft des Rechners eine Totzeit (Laufzeit) von 100 Millisekunden annehmen können. ***Transport Delay***

Booster

Der Booster steht hier stellvertretend für alle Signalverstärkungen und Umwandlungen auf dem Weg vom Rechnerausgang bis zum eigentlichen Stellglied. Modellieren Sie ihn als einen einfachen Tiefpaß mit einer „vernünftigen“ Zeitkonstante.

Hydraulik

Nehmen Sie an, der wesentliche Einfluß der Hydraulikversorgung bestehe darin, dass der Öldruck und damit der Durchfluß (Volumenstrom) im Steller begrenzt ist, was sich dort als eine Ratenbegrenzung auswirkt. Solche Stellratenbegrenzungen führen immer wieder zu spektakulären PIO-Situationen und Flugunfällen. Setzen Sie daher einen ***Rate Limiter*** hinter dem Booster.

Lose

Mechanische Verbindungen mit Gelenken, Umlenkungen, ... können Lose aufweisen. Auch wenn moderne Stellsysteme heute immer weniger mit Lose zu kämpfen haben, sollten Sie sich den Einfluß einer solchen ***Backlash*** einmal genauer ansehen und sich vielleicht einmal überlegen, wie Sie so etwas in Ihrer Lieblingsprogrammiersprache programmieren würden. Implementieren Sie einen ***Backlash*** hinter der Hydraulik.

Turbulenz

Wenn das Stellsystem Lose aufweist, haben es äußere Kräfte, die, durch Böen und Turbulenz hervorgerufen, auf die Stellflächen wirken, besonders leicht, die Flächen zu bewegen, da die Störkrafte in diesem Fall nicht gegen den Stellmotor arbeiten müssen. Addieren Sie daher hinter der Lose niederfrequentes Rauschen.

Anschlag

Ja, und dann hat ein Steller üblicherweise einen mechanischen oder virtuellen Anschlag. Ein Ruder kann sinnvollerweise nicht vollständig um seine Drehachse rotieren. Vielfach wird der Ruderwinkel flugzustandsabhängig auch programmtechnisch begrenzt. Schließen Sie daher den Steuerstrang am Ende mit einem ***Saturation***-Block ab.

Echtzeit

Damit Sie als Pilot mit Ihrem Mause-Stick einigermaßen realistische Eingaben machen können, stellen Sie das Integrationsintervall fest so ein, dass Sie ungefähr in Echtzeit simulieren. Sie erkennen das daran, dass der ***Scope***-Strahl zum Durchlauf durch ein 10-Sekundenfenster auch tatsächlich etwa 10 Sekunden braucht. Natürlich ist das einzustellende Integrationsintervall von der Rechengeschwindigkeit des jeweiligen

stellende Integrationsintervall von der Rechengeschwindigkeit des jeweiligen Rechners abhängig.

Noch Fragen, Ideen, Anregungen, Wünsche, ...?