1.4 Zusatzaufgaben

1.4.1 Rechnen mit Polynomen

Aufgabe 1.7. Gegeben ist das Polynom

$$p_1(s) = s^3 + 8s^2 + 19s + 12 \tag{1.21}$$

und $p_2(s)$ sei das charakteristische Polynom der Matrix

$$\mathbf{A} = \begin{bmatrix} -1 & 2 \\ -2 & -6 \end{bmatrix}. \tag{1.22}$$

Hinweis: Polynome können in MATLAB als Vektoren der absteigend geordneten Polynomkoeffizienten dargestellt werden, d. h. $p(s) = a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0$ wird als Vektor $\mathbf{p} = [a_n, a_{n-1}, \ldots, a_1, a_0]^T$ eingegeben. Zur Auswertung eines Polynoms an einer bestimmten Stelle s kann der Befehl `polyval()` verwendet werden.

Hinweis: Zur Bestimmung des charakteristischen Polynoms einer quadratischen Matrix kann der Befehl `poly()` verwendet werden.

Hinweis: Die Multiplikation zweier Polynome entspricht der Faltung ihrer Koeffizientenvektoren. Die (diskrete) Faltungsoperation kann mit dem Befehl `conv()` durchgeführt werden.

1. Berechnen Sie in MATLAB das Polynom $p_3(s) = p_1(s)p_2(s)$.
2. Schreiben Sie eine Funktion mit der Schnittstelle `pd = polydiff(p)`, welche als Argument `p` die Koeffizienten eines beliebigen Polynoms $p(s)$ erhält und als Rückgabewert `pd` die Koeffizienten der ersten Ableitung $\mathrm{d}p(s)/\mathrm{d}s$ zurückgibt. Vermeiden Sie dabei den Befehl `polyder()`.
3. Schreiben Sie eine Funktion mit der Schnittstelle `s0 = findzero(p,sstart)`, welche als Argument `p` die Koeffizienten eines beliebigen Polynoms $p(s)$ sowie einen Startwert `sstart` erhält. Die Funktion soll mittels des Newton-Verfahrens ausgehend vom Startwert `sstart` eine Nullstelle `s0` des Polynoms $p(s)$ suchen und zurückgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium. Sie können in Ihrem Algorithmus gegebenenfalls die Funktion `polydiff()` verwenden.
4. Testen Sie die Funktion `findzero()` anhand des Polynoms $p_3(s)$. Sie können dazu natürlich $p_3(s)$ zunächst grafisch darstellen.

5. Bestimmen Sie mit dem Befehl `roots()` die Nullstellen von $p_3(s)$.

6. Ist **A** eine Hurwitzmatrix? Nein, Satz 4.2 alle $\neq 0$ und gl. Vorzeichen

1.4.2 Integrationsalgorithmen

Beim expliziten Euler-Verfahren handelt es sich um die einfachste Einschrittmethode aus der Klasse der Runge-Kutta-Verfahren. Unter geeigneten Voraussetzungen konvergiert das explizite Euler-Verfahren mit Konvergenzordnung 1, was für viele Anwendungen zu ungenau ist.

Das Verfahren von Heun ist ein weiteres explizites Einschrittverfahren und gehört ebenfalls zur Klasse der Runge-Kutta-Verfahren. Es basiert auf einem Prädiktor- und einem Korrektor-Schritt, d.h. zuerst wird ein Schritt vor-prädiziert und im Anschluss korrigiert. Der Prädiktionsschritt

$$\mathbf{x}_{k+1}^P = \mathbf{x}_k + T_a \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{1.23}$$

wird mithilfe des expliziten Euler-Verfahrens berechnet. Anschließend wird dieser Schritt mittels der Trapezregel

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{2} T_a \Big(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}^P, \mathbf{u}_{k+1}) \Big) \tag{1.24}$$

also

$$\mathbf{x}_{k+1} = \frac{1}{2}\mathbf{x}_k + \frac{1}{2} \Big(\mathbf{x}_{k+1}^P + T_a \mathbf{f}(\mathbf{x}_{k+1}^P, \mathbf{u}_{k+1}) \Big), \tag{1.25}$$

korrigiert. Unter geeigneten Voraussetzungen konvergiert das Verfahren von Heun mit Konvergenzordnung 2, d.h. der globale Fehler des Verfahrens geht wie T_a^2 gegen Null.

Ein noch genaueres Einschrittverfahren aus der Klasse der expliziten Runge-Kutta-Methoden ist durch das sogenannte klassische Runge-Kutta-Verfahren in der Form

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{1.26a}$$

$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}_k + \tfrac{1}{2} T_a \mathbf{k}_1, \mathbf{u}_{k+1/2}) \tag{1.26b}$$

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{x}_k + \tfrac{1}{2} T_a \mathbf{k}_2, \mathbf{u}_{k+1/2}) \tag{1.26c}$$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{x}_k + T_a \mathbf{k}_3, \mathbf{u}_{k+1}) \tag{1.26d}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{6} T_a (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{1.26e}$$

gegeben. Hierbei bezeichnet $\mathbf{u}_{k+1/2}$ den Eingangsvektor $\mathbf{u}(t)$ zum Zeitpunkt $t = kT_a + \frac{1}{2}T_a$. Unter entsprechenden Voraussetzungen besitzt das klassische Runge-Kutta-Verfahren Konvergenzordnung 4.

Aufgabe 1.8. In Aufgabe 1.2 haben Sie bereits das explizite Euler-Verfahren für das dynamische System (1.7) verwendet. Implementieren Sie nun für das gleiche dynamische System das Verfahren von Heun und das klassische Runge-Kutta-Verfahren. Die Systemparameter sollen mit $m = 1\,\mathrm{kg}$, $c = 1\,\mathrm{N\,s/m}$, $k = 2\,\mathrm{N/m}$ unverändert bleiben. Wie in Aufgabe 1.2 soll ein Sprung aufgeschaltet werden. Auch der Integrationszeitraum bleibt mit $[0, 20]\,\mathrm{s}$ unverändert. Die Anfangswerte seien nun allerdings durch

$s_0 = 1\,\mathrm{m}$ und $v_0 = 0.5\,\mathrm{m/s}$ gegeben.

Vergleichen Sie die numerischen Ergebnisse des expliziten Euler-Verfahrens, des Verfahrens von Heun und des klassischen Runge-Kutta-Verfahrens jeweils mit der in Aufgabe 1.2 berechneten analytischen Lösung. Stellen Sie hierzu den Zeitverlauf der Zustände der numerischen, als auch der exakten Lösung in einer Grafik dar. Erstellen Sie jeweils eine Grafik entsprechend jeder numerischen Methode. Verkleinern Sie die Abtastzeit so lange bis die numerische Lösung des klassischen Runge-Kutta-Verfahrens und die analytische Lösung in der grafischen Darstellung hinreichend gut übereinstimmen. Was gilt bei gleicher Abtastzeit für die numerischen Lösungen des Verfahrens von Heun und des expliziten Euler-Verfahrens?

Aufgabe 1.9. Wir betrachten wieder das dynamische System (1.7) mit den gleichen Systemparametern, den gleichen Anfangswerten und gleichem Integrationszeitraum wie in Aufgabe 1.8. Der einzige Unterschied besteht darin, dass nun anstatt eines Sprungs eine Sinusschwingung der Form

$$u(t) = \sin(10\pi t/T), \quad t \in [0, 20]\,\mathrm{s}, \quad T = 20\,\mathrm{s}$$

aufgeschaltet werden soll. Berechnen Sie die Zeitverläufe der numerischen Lösungen mithilfe des expliziten Euler-Verfahrens, des Verfahrens von Heun und des klassischen Runge-Kutta-Verfahrens.

Die Berechnung der exakten Lösung ist für den oben angegeben zeitlichen Verlauf der Eingangsgröße recht mühsam. Eine Referenzlösung soll daher mit dem Befehl `lsim` aus der Control System Toolbox berechnet werden. Stellen Sie die Zeitverläufe der numerischen Lösungen denen, welche sie mit dem Befehl `lsim` berechnet haben, gegenüber. Erstellen Sie wieder jeweils eine separate Grafik entsprechend den Ergebnissen jeder numerischen Methode. Beschreiben Sie kurz wie sich die Wahl der Abtastzeit T_a auf die Güte der berechneten Lösungen auswirkt.

Abschließend soll der Begriff einer steifen Differentialgleichung bzw. eines steifen Differentialgleichungssystems $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ motiviert werden. Vereinfacht ausgedrückt handelt es sich bei steifen Differentialgleichungssystemen um solche Systeme, welche sich sehr viel einfacher mit impliziten als mit expliziten Zeitintegrationsmethoden lösen lassen. Es ist im Allgemeinen nicht möglich, den Begriff der Steifigkeit mathematisch eindeutig zu definieren. Typischerweise kann man jedoch sagen, dass bei steifen Differentialgleichungen Stabilitätsanforderung und weniger Genauigkeitsanforderungen die Größe der Zeitschrittweite (Abtastzeit) bestimmen. Oftmals klingen gewisse Komponenten der Lösung eines steifen Differentialgleichungssystems sehr viel schneller ab als andere.

Für lineare Systeme mit konstanten Koeffizienten kann man den sogenannten Steifigkeitsquotienten

$$S = \frac{\max_{j=1,\ldots,n} |\Re(\lambda_j)|}{\min_{j=1,\ldots,n} |\Re(\lambda_j)|} \tag{1.27}$$

mithilfe der Eigenwerte $\lambda_1, \ldots, \lambda_n$ der Dynamikmatrix A definieren. Gewöhnlicherweise gilt für den Steifigkeitskoeffizienten nicht-steifer Differentialgleichungssysteme $S \approx 1$, während der Steifigkeitskoeffizient steifer Systeme sehr große Werte annimmt.

Steife Differentialgleichungssysteme müssen in der Regel mit impliziten Zeitintegrationsmethoden gelöst werden. Das mit Abstand einfachste implizite Einschritt-Verfahren ist das implizite Euler-Verfahren, welches durch die Iterationsvorschrift

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_a \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) \tag{1.28}$$

definiert ist. In Abhängigkeit von $\mathbf{f}$ muss daher in jedem Zeitschritt ein lineares oder nichtlineares Gleichungssystem gelöst werden.

Aufgabe 1.10. Gegenstand von Aufgabe 1.10 ist wieder das dynamische System (1.7). Berechnen Sie den Steifigkeitsquotienten für die Systemparameter aus Aufgabe 1.8, d.h. mit $m = 1\,\mathrm{kg}$, $c = 1\,\mathrm{N\,s/m}$ und $k = 2\,\mathrm{N/m}$. Wie lautet der Wert des Steifigkeitsquotienten für den Fall $m = 1\,\mathrm{kg}$, $c = 1001\,\mathrm{N\,s/m}$ und $k = 1000\,\mathrm{N/m}$? Berechnen Sie den Zeitverlauf der Zustände mithilfe des expliziten Euler-Verfahrens für den Integrationszeitraum $[0, 20]\,s$. Wählen Sie die Anfangswerte wie in Aufgabe 1.8. Als externe Kraft soll wieder ein Sprung aufgeschaltet werden. Vergleichen Sie ihre Lösung mit der exakten Lösung. Wie groß darf die Abtastzeit ($20\,\mathrm{s}/T_a$ soll ganzzahlig sein) maximal sein, damit das explizite Euler-Verfahren ein qualitativ richtiges Ergebnis liefert? Stellen Sie den zeitlichen Verlauf der Zustände einmal für den gesamten Integrationszeitraum und einmal für einen kleinen Zeitraum zu Beginn der Simulation dar. Sie können dazu den Befehl `xlim` verwenden (z.B. `xlim([-2, 22])` und `xlim([-0.05, 1.0])`). Vergleichen Sie den zeitlichen Verlauf beider Lösungskomponenten.

Berechnen Sie schließlich den zeitlichen Verlauf der Zustände mithilfe des impliziten Euler-Verfahrens. Wie groß darf die Abtastzeit maximal sein, damit das implizite Euler-Verfahren ein qualitativ richtiges Ergebnis liefert?