Simulation technischer Systeme mit Python

Prüfung im Wintersemester 2023

Bitte tragen Sie Nachnamen, Vornamen und Matrikelnummer, Hörsaal und Platznummer deutlich lesbar in die vorgesehenen Felder ein.

Name	Vorname	Matrikelnummer	Hörsaal	Platz-Nr.

Punkte	Aufgabe 1	Aufgabe 2	Aufgabe 3	Erreichte Pkte.	Note:	
					bestanden?	ja / nein

Ich versichere hiermit, dass ich mich geistig und körperlich in der Lage befinde, die Prüfung abzulegen (d. h. prüffähig bin).

☐ Ja

☐ Nein

_____
Unterschrift Studierende/r

Abschlussprüfung – WiSe 22/23

Hinweise zur Prüfung

Die Prüfung ist bestanden, wenn mindestens 50% der maximal erreichbaren Punkte erzielt werden. Zur Bearbeitung der Prüfung stehen 100 Minuten Zeit zur Verfügung.

In jeder Abbildung müssen alle Achsenbeschriftungen sinnvoll gewählt werden.

Hinweise zur Bearbeitung:

- Es sind alle nicht kommunikativen Hilfsmittel (z. B. Taschenrechner, Skript, Übungsunterlagen, Vorlesungsmitschrift) zugelassen. Chatbots wie ChatGPT sind nicht zugelassen. Die Arbeiten müssen selbstständig bearbeitet werden! Es dürfen keine großen Code-Blöcke per copy/paste eingefügt werden.
- Im Stud.IP-Ordner „Prüfungsaufgaben“ finden Sie außer dieser Aufgabenstellung zusätzliche Dateien, die für die Lösung der Aufgaben benötigt werden.
- Sorgen Sie am Ende der Prüfung eigenverantwortlich für das ordnungsgemäße Hochladen Ihrer Lösungen über den Stud.IP-Ordner „Abgabe Prüfungen“ im Prüfungskurs! Bitte benennen Sie Ihre Dateien nach folgendem Schema: „FAMILIENNAME_VORNAME_MATRIKELNUMMER_AufgabeX.py“, wobei X die Aufgabennummer bedeutet. Bitte laden sie jede Aufgabe einzeln und nicht als .zip hoch.
- Bitte beachten Sie, dass die **Bearbeitungszeit 100 Minuten** beträgt! Bitte sorgen Sie eigenverantwortlich dafür, dass Sie Ihre Lösungen rechtzeitig vor Ende der Abgabefrist **(16:50 Uhr)** über Stud.IP hochladen!
- Lesen Sie vor der Bearbeitung der Aufgaben die Aufgabenstellung aufmerksam durch!
- Für die ordnungsgemäße Abgabe Ihrer Arbeit sind Sie selbst verantwortlich.

Diese Aufgabenstellung umfasst 5 Seiten. Die maximal erreichbare Punktezahl beträgt 50 Punkte.

Aufgabe 1: Gleichungen Lösen und Plotten(15 Punkte)

Die Relative Dichte (rd) von Gyroid-Strukturen kann näherungsweise über folgenden Zusammenhang bestimmt werden:

$$rd = 3\,x - 4(x)^3 \; mit \; x = t/l$$

Dabei ist t die Wandstärke in mm und l die Länge einer Einheitszelle in mm. Für die Vorauslegung einer solchen Struktur, wollen Sie die Wandstärke t in Abhängigkeit der Einheitszellenlänge l für eine bestimmte Relative Dichte rd berechnen.

a) Stellen Sie zunächst die Gleichung so um, dass die Linke Seite Null ergibt und schreiben Sie eine Funktion **tl_ratio(x)**, die das Verhältnis aus Wandstärke und Einheitszellenlänge x für eine bestimmte Relative Dichte zurückgibt.

b) Verwenden Sie die Funktion **newton** aus der **scipy.optimize** Bibliothek um das Verhältnis t/l für eine Relative Dichte von $rd = 0.05$ zu bestimmen. Wählen Sie als Startwert $x_0 = 0.1$. Berechnen Sie nun die Wandstärke t für eine Einheitszellenlänge $l = 22.5$.

c) Schreiben Sie die Funktion **wall_thickness(rd, l)**, die die Wandstärke in Abhängigkeit der Einheitszellenlänge und der Relativen Dichte zurückgibt.

d) Plotten Sie die Wandstärke über der Einheitszellenlänge im Intervall $l = [5{,}40]$ und für die Relativen Dichten $rd = (0.25, 0.50, 0.75{,}1.00)$. Beschriften Sie die Achsen und weisen Sie jeder Kurve ihre Relative Dichte in der Legende zu. Nutzen Sie zu diesem Zweck einen formatierten String.

Aufgabe 2: Schwingungstilger (20 Punkte)

Ihr Vorgesetzter möchte komplexe Differentialgleichungen für elektrische Systeme auswerten. Er ist ein alter Theoretiker und ist sehr gut darin die DGLS auf dem Papier aufzustellen kennt sich aber überhaupt nicht mit Python aus. Er benötigt aus diesem Grund Ihre Hilfe um eine grobe Abschätzung der Impedanz numerisch zu berechnen. Da Ihr Vorgesetzter numerischen Lösungen generell misstraut, möchte er zunächst einen Test mit einem sehr einfachen Reihenschwingkreis. Der Reihenschwingkreis mit erzwungener Schwingung hat die Differentialgleichung:

$$2\pi f \hat{U} \sin(2\pi f \cdot t) = L\ddot{I} + R\dot{I} + \frac{I}{C}$$

Der Schwingkreis soll die folgenden Parameter haben (Achtung Einheiten!):

$$\hat{U} = 1V$$
$$R = 1\Omega$$
$$C = 1mF$$
$$L = 10mF$$

a) Nutzen Sie die **solve_ivp** Funktion aus der **scypi.integrate** Bibliothek um die Differentialgleichung im zwischen 0 und 0.3 Sekunden zu Lösen. Übergeben Sie die Zielfrequenz $f = 10$ Hz als Argument über **args**. Verwenden Sie **(0,0)** als Startwerte für Strom I und Stromänderung $\dot{I}$.

b) Überprüfen Sie 100 Frequenzen mit gleichmäßig logarithmischem Abstand von $f = 10$Hz bis $f = 100$Hz. Simulieren Sie 0.3s der Stromantwort des Systems und werten Sie die letzten 0.1s aus. Bilden Sie den Betrag der Impedanz Z für jede Frequenz über die Gleichung:

$$|Z| = \frac{2\hat{U}}{I_{max} - I_{min}}$$

c) Tragen Sie den Kehrwert der Impedanz $1/|Z|$ über der Frequenz in einer Grafik auf. Beschriften Sie alle Achsen.

Aufgabe 3: Objekt-Orientierte Programmierung (15 Punkte)

Als angehender Ingenieur müssen Sie immer wieder einfache, zweidimensionale Funktionen plotten. Da sie diese Plots in Zukunft nicht mehr händisch anlegen wollen, schreiben Sie sich eine Plot-Klasse:

a) Definieren Sie die Klasse **FunctionPlot.** In der magischen Methode __init__() sollen die Parameter **func**, **limits** und **args** übergeben werden und als geschützte Attribute abgelegt werden. Während **limits** mit dem Standardwert **(0, 1)** übergeben wird, wird **args** mit dem Standardwert **None** übergeben. Das geschützte Attribut label wird in der __init__ Methode mit dem Namen der übergebenen Funktion definiert.

b) Fügen Sie die Properties **x** und **y** zur Klasse hinzu. Die Property **x** soll ein eindimensionales Array mit 50 gleichverteilten Werten in den Grenzen von **self._limits** zurückgeben. Die Property **y** soll die Funktionswerte von **self._func** an den Stellen **x** zurückgeben. Wenn zusätzliche Funktionsargumente für **self._func** vorhanden sind, sollen diese bei der Funktionsauswertung berücksichtigt werden.

c) Schreiben Sie die Methode **plot(),** in der der Plot inklusive Legende mit Hilfe der Bibliothek **matplotlib.pyplot** angelegt wird.

d) Schreiben Sie die Methode **show()**, in der die klasseneigene Plotmethode aufgerufen wird und der Plot angezeigt wird.

e) Definieren Sie außerhalb der Klassendefinition die Funktion **f(x, rd=0.1)**:

$$f(x, rd = 0.1) = 3x - 4x^3 - rd$$

f) Instanziieren Sie ein Objekt der Klasse **FunctionPlot** mit der Funktion f und plotten Sie diese im Bereich (0, 0.1) mit der Methode **show()**

g) Übergeben Sie die neue relative Dichte **rd=0.01** als zusätzliches Argument. Wie verändert sich der Plot?