Institut für Mechanik und Adaptronik

Prof. Dr.-Ing. M. Sinapius

Simulation technischer Systeme mit Python

Dr. Ing. H. Schlums
M. Sc. H. Traub
M. Sc. K. Karimian
M. Sc. P. Scholle

Simulation technischer Systeme mit Python

Take Home Exam im Wintersemester 2021/22

Hinweise zur Bearbeitung:

- Es sind alle nicht kommunikativen Hilfsmittel (z. B. Taschenrechner, Skript, Bücher, Vorlesungsmitschrift) zugelassen. Die Arbeiten müssen selbstständig bearbeitet werden! Es dürfen keine großen Code-Blöcke per copy/paste eingefügt werden.
- Sorgen Sie am Ende der Prüfung eigenverantwortlich für das ordnungsgemäße Hochladen Ihrer Lösungen über den Stud.IP-Ordner „Abgabe Take Home Exam“ im Prüfungskurs! Bitte benennen Sie Ihre Dateien nach folgendem Schema: „FAMILIENNAME_VORNAME_MATRIKELNUMMER_AufgabeX.py“, wobei X die Aufgabennummer bedeutet.
- Bitte beachten Sie, dass die **Bearbeitungszeit 100 Minuten** beträgt! Bitte sorgen Sie eigenverantwortlich dafür, dass Sie Ihre Lösungen rechtzeitig vor Ende der Abgabefrist **(12:20 Uhr)** über Stud.IP hochladen!
- Bestätigen Sie mit Ihrer Unterschrift auf der „Erklärung nach Abschluss der Online-Prüfung“ (separates Dokument), dass Sie für die Bearbeitung keine unerlaubten Hilfsmittel genutzt haben! Dort ist auch der Prüfungsbeginn **(10:30 Uhr)** sowie das Prüfungsende **(12:10 Uhr)** einzutragen. Digitalisieren Sie die Erklärung bitte anschließend und benennen Sie die Datei nach folgendem Schema: „FAMILIENNAME_VORNAME_MATRIKELNUMMER_Erklärung.pdf“! Legen Sie diese Datei ist im Ordner „Abgabe Erklärung“ ab!
- Sollten Sie auf technische Schwierigkeiten stoßen, informieren Sie uns umgehend per E-Mail an: **h.schlums@tu-braunschweig.de**
- Lesen Sie vor der Bearbeitung der Aufgaben die Aufgabenstellung aufmerksam durch!
- Für die ordnungsgemäße Abgabe Ihrer Arbeit sind Sie selbst verantwortlich.

Diese Aufgabenstellung umfasst 6 Seiten. Die maximal erreichbare Punktezahl beträgt 50 Punkte.

Erreichte Punktzahl / Gesamtpunktezahl:	/ 50	Note:

Institut für Mechanik und Adaptronik

Prof. Dr.-Ing. M. Sinapius

Simulation technischer Systeme mit Python

Dr. Ing. H. Schlums
M. Sc. H. Traub
M. Sc. K. Karimian
M. Sc. P. Scholle

Abschlussprüfung – WiSe 22

Nachname :

Vorname :

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. Bitte denken Sie daran die „Erklärung Online-Prüfung" auszufüllen und nach der Prüfung hochzuladen. Die Abgabe erfolgt über den StudIP Ordner „Abgabe Take Home Exam".

Hilfsmittel: ***Die Prüfung muss selbstständig angefertigt werden. Es dürfen keine großen Code-Blöcke per copy/paste eingefügt werden. Alle weiteren Hilfsmittel sind erlaubt***

Aufgabe:	**1**	**2**	**3**	**Bestanden**
Punkte:				**Ja** ☐
Korrektor:				**Nein** ☐
				Note:

Aufgabe 1: Curve Fitting (10 Punkte)

Eine Forschungsgruppe aus der Biomechanik hat ein Kompressionsexperiment an einer Muskelzelle durchgeführt. In der Datei **experimentalData.csv** sind die ermittelten Kraft-Dehnungs-Daten zur Verfügung gestellt, welche durch eine Modellfunktion angenähert werden sollen. Gehen Sie dabei folgendermaßen vor:

a) Lesen Sie die Daten **experimentalData.csv** in Python ein, wobei die Dehnungs- und Kraftwerte in Vektoren gleicher Länge gespeichert werden sollen.
b) Fitten Sie mit Hilfe von **numpy.polyfit** ein Polynom 4. Ordnung an die Messdaten an. Der normierte Fehler der Annäherung lässt sich anschließend durch die Formel

$$e = \frac{LSE}{N * F_{max}}$$

berechnen. Hierbei steht LSE für den Least Squares Error, der von der Funktion **numpy.polyfit** zurückgegeben werden kann, N für die Anzahl der Messpunkte und Fmax für die gemessene Maximalkraft. Geben Sie den Fehler e in Prozent als Gleitkommazahl in der Konsole im folgenden Format aus:
„Das Polynom 4. Ordnung besitzt einen Fehler von XX.XX %“.
c) Plotten Sie die Messdaten als graue und das gefittete Model als schwarze Kurve in einem Fenster mit Gitternetzlinien. Durch eine Legende sollen sowohl die experimentell ermittelten Daten als auch das Modell gekennzeichnet sein. Zusätzlich soll die x-Achse nur Werte im Intervall [0, 0.6] abbilden. Achten Sie auf eine sinnvolle Wahl der Achsenbeschriftungen.

Aufgabe 2: Ein vereinfachtes Modell der Harzaushärtung (20 Punkte)

Die Aushärtung von Epoxidharzen kann durch Differentialgleichungen beschrieben werden. Ein zentraler Parameter dieser Reaktion ist der Aushärtungsgrad α, der beschreibt wie weit eine Reaktion fortgeschritten ist. Dabei entspricht α=1 einer vollständigen Aushärtung. Da es sich um eine exotherme Reaktion handelt, steigt die Temperatur T während der Aushärtung. Folgende Differentialgleichungen beschreiben das Außhärteverhalten:

$$\frac{d\alpha}{\mathrm{dt}} = A_1 e^{-\frac{E_1}{RT}} \alpha^m (1-\alpha)^n$$

$$\frac{dT}{dt} = \frac{1}{c_P}\left(Q_m \frac{d\alpha}{\mathrm{dt}} + \dot{Q}_s\right)$$

dabei sind $A_1 = 400 s^{-1}, E_1 = 18700 \frac{J}{mol}$, $m = 1.5, n = 1.7$ und $Q_m = 84000 \frac{J}{kg}$ die Materialparameter des Epoxidharzes, $R = 8.3 \frac{J}{mol \cdot K}$ ist die allgemeine Gaskonstante, $c_P = 1100 \frac{J}{kg\, K}$ die spezifische Wärmekapazität des Epoxids und $\dot{Q}_s = -142 \frac{J}{kg\, s}$ ein als konstant angenommener spezifischer Wärmeabfluss aus dem System.

a) Lösen Sie die Differentialgleichung mithilfe von scipy. Nutzen Sie als Anfangsbedingung eine Temperatur von T=280K und einen Aushärtungsgrad zu Beginn von α=0.01. Berechnen Sie die ersten 300 Sekunden der Reaktion und wählen Sie den Runge Kutta 3(2)-solver zur Lösung.
b) Plotten Sie das Ergebnis in Form eines subplots mit drei übereinander liegenden plots. Im obersten plot soll dabei die Temperatur T, im mittleren der Aushärtungsgrad α und im unteren die Aushärtungsrate dα/dt dargestellt werden. Plotten Sie die Temperatur als rote Linie, den Aushärtegrad als blaue Linie, und die Aushärtungsrate als grüne Linie. Aktivieren Sie bei allen plots das grid.
c) Die Harzaushärtung ist stark von der Temperatur abhängig. Ermitteln Sie diesen Einfluss indem Sie die Starttemperatur der Aushärtungsreaktion verändern. Variieren Sie die Temperatur im Bereich zwischen 270K und 350K mit einem Intervall von 10 K. Bestimmen Sie die Zeit nach der die Reaktion vollständig abgelaufen ist. Die Reaktion gilt als abgeschlossen wenn ein Aushärtegrad von 99% erreicht wurde. Plotten Sie die Zeit bis zur vollständigen Aushärtung über der Starttemperatur. Einzelne Datenpunkte sollen als X dargestellt und mit einer gestrichelten Linie verbunden werden.

Aufgabe 3: Objekt-Orientierte Programmierung (20 Punkte)

Das Flächenträgheitsmoment ist ein entscheidender Parameter für die Durchbiegung eines Balkens. Während sich das Flächenträgheitsmoment einfacher Geometrien analytisch bestimmen lässt, sind bei komplexen Geometrien numerische Methoden erforderlich. In dieser Aufgabe soll eine Klasse „**Inertia**“ erzeugt werden, die die Flächenträgheitsmomente $\boldsymbol{I_y}$ und $\boldsymbol{I_z}$ beliebiger Geometrien auf Basis einer Bitmap berechnet. In der Bitmap repräsentiert der Wert **1** (weiß) die Geometrie, der Wert **0** (Schwarz) den Hohlraum. Die Klasse soll verwendet werden, um die Flächenträgheitsmomente zweier unterschiedlicher Geometrien zu vergleichen. Die Geometrien sind in Abbildung 3 dargestellt.

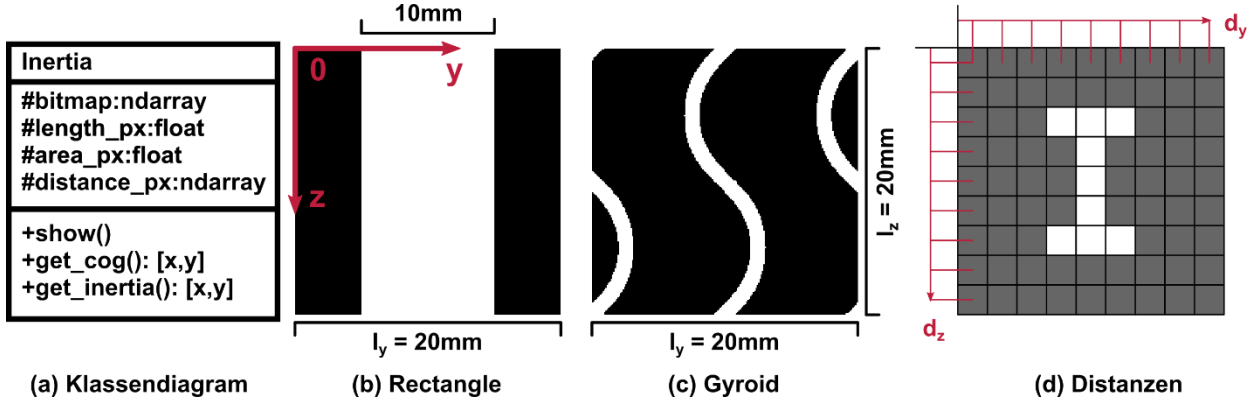

Abbildung 1: (a) Klassendiagram, (b) Rechteck, (c) Gyroid und (d) Schwerpunktberechnung

Hinweis: Die Klasse Inertia soll ausschließlich für quadratische Bitmaps mit quadratischen Pixeln Programmiert werden. Das bedeutet für Abbildung 1 (b) und (c) $l_y = l_z$ *und für Abbildung 3 (d)* $d_y = d_z$*. Das Flächenträgheitsmoment* I_y *bestimmt sich aus der Summe der Flächenträgheitsmomente der einzelnen Pixel* I_{yi} *und der Steineranteile* $z_i^2 \cdot A_i$*. Dabei ist* z_i *der z-Abstand eines Pixels zum Schwerpunkt. Der Schwerpunkt ist mit* $(y_{cog} = 10\,, z_{cog} = 10)$ *gegeben.*

$$I_y = \Sigma\, \mathrm{I_{y_i}} + \Sigma\, z_i^2 \cdot A_i \; I_{y_i} = \frac{b_i \cdot h_i^3}{12}$$

$$I_{y_i} = \frac{b_i \cdot h_i^3}{12}$$

$$b_i = h_i$$

1. Definieren Sie die Klasse **Inertia**. In der magischen Methode **__init__(self, fname, size)** sollen die geschützten Attribute (protected attributes) **_bitmap, _length_px, _area_px** und **_distance_px** initialisiert werden. Nutzen Sie den befehl **numpy.genfromtxt(fname)** um eine CSV Datei zu importieren, die am Dateipfad **fname** gespeichert ist und weisen Sie die Matrix dem Attribut **_bitmap** zu. Bestimmen Sie die Länge und Fläche eines Pixels (**_length_px, _area_px)** aus der Dimension der Matrix und der Kantenläge des Quadrats, die durch den Parameter **size** übergeben wird. Bestimmen Sie nun den y- und z-Abstand jedes Pixelmittelpunkts vom Ursprung im Vektor **_distance_px** ($d_{px} = d_y = d_z$ in Abbildung 3 (d)). Verwenden Sie dafür den befehl **numpy.linspace()**
2. Definieren Sie die öffentliche Methode (public method) **show()** und verwenden Sie den Befehl **imshow()** aus der **pyplot** Bibliothek um die **_bitmap** anzuzeigen.
3. Schreiben Sie die öffentliche Methode **get_inertia().** Die Methode soll als Rückgabewert eine Liste mit den Flächenträgheitsmomenten $\boldsymbol{I_y}$ und $\boldsymbol{I_z}$ zurückgeben. Machen sie sich das Attribut **_distance_px** (d_{px}) und die Methode **get_cog()** bei der Berechnung der Momente zunutze.
4. Instanziieren sie außerhalb der Klassendefinition zwei Objekte der Klasse **Inertia**, die jeweils das Flächenträgheitsmoment der Bitmap **„rect.csv“** und der Bitmap **„gyroid.csv“** ermitteln. Verwenden Sie die Methode **show()** um die Bitmaps zu visualisieren. Nutzen Sie die Methoden **get_cog()** und **get_inertia()** um die Schwerpunkte und Flächenträgheitsmomente der Geometrien zu ermitteln. Geben sie die Ergebnisse in einem formatierten String aus. Welche Geometrie hat jeweils das höhere Flächenträgheitsmoment in **y** und **z** Richtung? Vergleichen Sie das Ergebnis von „rect.csv“ mit einer analytischen Berechnung nach Abbildung 3a.