MATLAB/ Simulink | Numerische Integration

Beschreibung

(a) `runScript.m`

(b) `Automobilfederung.m`

In dieser Hausaufgabe werden Sie das vorgegebene Modell einer Automobilfederung vervollständigen, um anschließend einen Integrator für das Modell zu implementieren. Die Programmstruktur ist bereits vorgegeben. Machen Sie sich daher zunächst mit dieser vertraut. Datei a (runScript) dient zum Starten des Programms. Dabei wird zunächst das Eingangssignal für das zu simulierende System erstellt:

```
u = @(t) [(t <= 1) (t > 1)]*[0; 0.3];
```

Listing 1: Eingangssignal

Es wird damit das Auffahren auf einen Bordstein zum Zeitpunkt `t = 1s` simuliert. Anschließend wird die Klasse `Automobilfederung` instanziiert, wobei die numerischen Werte der Modellparameter vorgegeben sind. Neben dem eigentlichen Modell `(rhs)` verfügt das Objekt über eine `sim`-Methode, welche das Model integriert. Ziel ist es für das vorgegebene Modell zunächst die Zustandsmatrizen A und B zu realisieren, um anschließend einen fixed step Integrator nach dem klassischen Runge Kutta Verfahren zu implementieren. Machen Sie sich dazu mit dem Abschnitt **Runge-Kutta Verfahren** aus **Simulation.pdf**, welche Sie in lec05 finden, vertraut. Vervollständigen Sie in der Klasse b den Code an den entsprechend gekennzeichneten Stellen:

```
% ========= YOUR CODE HERE =========
```

Untersuchen Sie anschließend anhand der gegebenen Visualisierung, bis zu welcher Integrationsschrittweite eine stabile Simulation gewährleistet ist. Zur Beantwortung soll ihre Abgabe im Abschnitt

```
%% simulate model
```

der `runScript.m` Datei lediglich einen einfachen Aufruf der Simulationsmethode und der Visualisierungsmethode beinhalten.

```
model.sim('t0', 0, 'tfinal', 3, 'y0', [0 0 0 0], 'stepsize', stepsize);
model.visualizeResults();
```

Listing 2: Abgabe

Beim Simulationsaufruf soll genau die `stepsize` übergeben werden, welche gerade noch eine stabile Integration sicherstellt.

Bewertungskriterien

Sie können mit dieser Abgabe 15 Punkte erreichen. Sie erhalten zunächst für die Umsetzung, welche die Funktionalität des Programms sicherstellt 60% der maximal erreichbaren Punkte. Darüber hinaus, entfallen 40% auf Stil und performance-orientierte Implementierung. Halten Sie sich dazu, wie in der Vorlesung besprochen, an ein Minimum an Clean-Code Regeln.