4.2.5 Problem Set: Robertson's Autocatalytic Reaction

Discussions

All posts sorted by recent activity

Now we will consider the use of an implicit method, specifically Backward Euler, to solve an initial value problem in which $\underline{f}$ depends nonlinearly on $\underline{u}$. The following IVP is a model of an autocatalytic chemical reaction:

$$\frac{\mathrm{d}Y_0}{\mathrm{d}t} = -k_0 Y_0 + k_2 Y_1 Y_2 \tag{4.37}$$

$$\frac{\mathrm{d}Y_1}{\mathrm{d}t} = k_0 Y_0 - k_1 Y_1^2 - k_2 Y_1 Y_2 \tag{4.38}$$

$$\frac{\mathrm{d}Y_2}{\mathrm{d}t} = k_1 Y_1^2 \tag{4.39}$$

where Y_i is the mass fraction of species i and k_i are constants. This IVP was first presented by Robertson in 1966 as an example of a stiff system and hence is known now as Robertson's problem. We will use the following values,

$$k_0 = 4.0\text{E-}2\,\mathrm{s}^{-1}, \quad k_1 = 3.0\text{E}7\,\mathrm{s}^{-1}, \quad k_2 = 1.0\text{E}4\,\mathrm{s}^{-1} \tag{4.40}$$

Following our standard IVP notation, the state vector we choose as,

$$\underline{u} = [Y_0, Y_1, Y_2]^T \tag{4.41}$$

For the initial condition at $t = 0$, we will use,

$$\underline{u}(0) = [1.0, 0.0, 0.0]^T \tag{4.42}$$

Thus, only Y_0 is present initially.

Complete the following steps:

1. Implement the `calc_all` method in `solve_robertson.py` to calculate $\underline{f}$ and $\partial \underline{f}/\partial \underline{u}$ given $\underline{u}$. See the docstring for details.

   Further, complete the implementation of `test_calc_all` which will test that your implementation of `calc_all` is correct. Specifically, within `test_calc_all`, call `calc_all` to determine $\underline{f}$ and $\partial \underline{f}/\partial \underline{u}$ for $\underline{u} = [0.5, 0.05, 0.45]^T$ (and use the same values of k_0, k_1, and k_2 given above). The correct values of `f` and `f_u` are already included in `test_calc_all` and are named `f_correct` and `f_u_correct`.

The main block of `solve_robertson.py` calls `test_calc_all`. When, `test_calc_all` is working, uncomment the next line in the main block beginning `time_FE = ...`, in preparation for the next part.

2. Implement the `plot_Y` method in `solve_robertson.py` to produce plots of the

< Previous R Next >

solve_robertson.py, the desired plot of the Forward Euler method (using $\Delta t = 1.0\text{E-}4$

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play
Download on the App Store

`myNewton` in `mynonlinsolver.py`. A tester is provided which can be checked by running