edX MITx CSE.0002x

Introduction to Computational Science and Engineering

Help sandipan_dey

Course Progress Dates Discussion MO Index

Course / 12 Stiffness and Implicit Methods for IVPs / 12.2 Explicit and Implicit Methods

‹ Previous ✓ Next ›

12.2.2 Example: Application of Backward Euler to Oscillating Combustion

Bookmark this page

MO2.3 **MO2.4** **MO2.7**

Let's return to the oscillating combustion problem we discussed earlier and this time apply the Backward Euler method. Because of the simplicity of this problem, Backward Euler is relatively easy (and in fact the computational cost of a single step of Backward Euler is essentially the same as a single step of Forward Euler for this simple problem). First, let's re-write the governing differential equation in Equation (12.1) using the notation that u is the state and $f(u,t)$ is the forcing so that for this problem,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u,t) \tag{12.24}$$

$$u = [F] \tag{12.25}$$

$$f(u,t) = -\frac{1}{\tau}u + I_F(t) \tag{12.26}$$

Applying the Forward Euler method to this problem gives the following evolution equation,

$$v^{n+1} = v^n + \Delta t\left[-\frac{1}{\tau}v^n + I_F(t^n)\right] \tag{12.27}$$

And applying the Backward Euler method gives the following evolution equation,

$$v^{n+1} = v^n + \Delta t\left[-\frac{1}{\tau}v^{n+1} + I_F(t^{n+1})\right] \tag{12.28}$$

Because of the simplicity of the $f(u,t)$, the dependence on v^{n+1} can easily be re-arranged to give the following result for v^{n+1}:

$$v^{n+1} = (1+\Delta t/\tau)^{-1}\left[v^n + \Delta t I_F(t^{n+1})\right] \tag{12.29}$$

To see how this looks in Python, compare the Forward Euler and Backward Euler integration loops for this oscillating combustor problem:

```
# Loop from from t=0 to t>=tF
for n in range(N-1):
    tn = t[n]
    vn = v[n]

    I_F = 0.5*A_F*(1-math.cos(2*pi*tn/T_F))

    # Update solution using Forward Euler
    v[n+1] = vn + dt*(-vn/tau + I_F)
```

```
# Loop from from t=0 to t>=tF
```

Discussions

All posts sorted by recent activity

```
for n in range(N-1):
    tn1 = t[n] + dt
    vn  = v[n]

    I_F = 0.5*A_F*(1-math.cos(2*pi*tn1/T_F))

    # Update solution using Backward Euler
    v[n+1] = (vn + dt*I_F)/(1+dt/tau)
```

The Backward Euler results are shown for $\Delta t =$ 0.01, 0.001, and 0.0001 seconds in Figure 12.7. As can be k

‹ Previous

Next ›

these timestep. And, the solutions at $\Delta t =$ 0.001 and 0.0001 seconds are nearly the same and hence indicative the numerical solution does not require a

© All Rights Reserved

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

Blog
Contact Us
Help Center
Security
Media Kit

GET IT ON Google play

Download on the App Store

▶ 0:00 / 0:00

Backward Euler method to our

oscillating