example- Second-order ODE with step input

This example is meant to illustrate why using a backward-looking Euler approximation to derivatives is better (i.e., stable) than using a forward-looking Euler approximation (can be unstable) to derivatives [e.g., equations (2.69) and (2.71) of the text]

First, consider the general second-order ODE

\[
\frac{d^2 y(t)}{dt^2} + (s_1 + s_2) \frac{dy(t)}{dt} + s_1s_2 y(t) = s_1s_2 x(t)
\]

where

\[
y(0) = 0, \quad \left. \frac{dy(t)}{dt} \right|_{t=0} = 0, \quad \text{and} \quad x(t) = u(t).
\]

**Analytic Solution**

Assume \( y(t) = Ae^{s_1t} \), then the characteristic equation for the homogeneous or unforced solution, i.e., \( \frac{d^2 y_u(t)}{dt^2} + (s_1 + s_2) \frac{dy_u(t)}{dt} + s_1s_2 y_u(t) = 0 \), is

\[
s^2 + (s_1 + s_2)s + s_1s_2 = (s + s_1)(s + s_2) = 0 \quad \text{with roots} \ -s_1 \quad \text{and} \ -s_2.
\]

This yields the unforced solution

\[
y_u(t) = A_1 e^{-s_1t} + A_2 e^{-s_2t}.
\]

The inhomogeneous or forced solution is \( y_f(t) = y_{ss} = y(t \to \infty) \). At steady-state derivatives go to zero and the ODE becomes

\[
0 + (s_1 + s_2)0 + s_1s_2 y(t \to \infty) = s_1s_2 u(t \to \infty) = s_1s_2 \quad \text{and} \quad y(t \to \infty) = 1.
\]

The total solution is the sum of the unforced and forced solutions

\[
y(t) = y_u(t) + y_f(t) = A_1 e^{-s_1t} + A_2 e^{-s_2t} + 1.
\]
Next, apply the initial conditions to solve for the constants $A_1$ and $A_2$.

From $y(0) = 0$, we get the equation $y(0) = 0 = A_1 + A_2 + 1 \Rightarrow A_1 + A_2 = -1$.

From $\frac{dy(t)}{dt} \bigg|_{t=0} = 0$, we get the equation

$$
\frac{dy(t)}{dt} \bigg|_{t=0} = 0 = (-s_1 A_1 e^{-s_1 t} - s_2 A_2 e^{-s_2 t} + 0) \bigg|_{t=0} \Rightarrow 0 = -s_1 A_1 - s_2 A_2.
$$

Solving the two equations for the two unknowns yields

$$
A_1 = \frac{-s_2}{s_2 - s_1} \text{ and } A_2 = \frac{s_1}{s_2 - s_1}.
$$

Therefore, the final analytic solution is

$$
y(t) = \left(\frac{-s_2}{s_2 - s_1}\right) e^{-s_1 t} + \left(\frac{s_1}{s_2 - s_1}\right) e^{-s_2 t} + 1 \quad t \geq 0.
$$

**Forward-looking Euler Numerical Solution (Text method)**

Applying equations (2.69) and (2.71) of the text to the general second-order ODE yields the difference equation

$$
\frac{y[n+2] - 2y[n+1] + y[n]}{T^2} + (s_1 + s_2) \frac{y[n+1] - y[n]}{T} + s_1 s_2 y[n] = s_1 s_2 x[n] = s_1 s_2 u[n]
$$

which simplifies to

$$
y[n+2] + \left[ -2 + (s_1 + s_2)T \right] y[n+1] + \left[ 1 - (s_1 + s_2)T + s_1 s_2 T^2 \right] y[n] = s_1 s_2 T^2 u[n].
$$

Exploiting time-invariance (i.e., let $n \to n - 2$) yields

$$
y[n] + \left[ -2 + (s_1 + s_2)T \right] y[n-1] + \left[ 1 - (s_1 + s_2)T + s_1 s_2 T^2 \right] y[n-2] = s_1 s_2 T^2 u[n-2]
$$

for $n \geq 1$. The initial conditions are $y[0] = y[-1] = x[-1] = 0$ and $x[0] = 1$. 


Backward-looking Euler Numerical Solution

Applying the backward-looking Euler approximations for derivatives

\[
\frac{dy(t)}{dt}\bigg|_{t=nT} = \frac{y[n] - y[n-1]}{T} \quad \text{and} \quad \frac{d^2 y(t)}{dt^2}\bigg|_{t=nT} = \frac{y[n] - 2y[n-1] + y[n-2]}{T^2}
\]

to the general second-order ODE, yields the difference equation

\[
\frac{y[n] - 2y[n-1] + y[n-2]}{T^2} + (s_1 + s_2)\frac{y[n] - y[n-1]}{T} + s_1 s_2 y[n] = s_1 s_2 u[n].
\]

This can be simplified first to

\[
\left[1 + (s_1 + s_2)T + s_1 s_2 T^2\right]y[n] + \left[-2 - (s_1 + s_2)T\right]y[n-1] + y[n-2] = s_1 s_2 T^2 u[n].
\]

Then, the difference equation can be put in the standard form

\[
y[n] + \left(-2 - (s_1 + s_2)T\right)\frac{y[n-1]}{1 + (s_1 + s_2)T + s_1 s_2 T^2} + \left(\frac{1}{1 + (s_1 + s_2)T + s_1 s_2 T^2}\right)y[n-2] = \left(\frac{s_1 s_2 T^2}{1 + (s_1 + s_2)T + s_1 s_2 T^2}\right)u[n]
\]

valid for \( n \geq 1 \).

The necessary initial conditions for \( y[n] \) are

\[
y(0) \Rightarrow y[0] = 0
\]

and

\[
\frac{dy(t)}{dt}\bigg|_{t=0} = 0 \Rightarrow \frac{y[0] - y[0-1]}{T} = \frac{0 - y[-1]}{T} = 0 \Rightarrow y[-1] = 0.
\]

Since there are no past terms for \( x[n] \), it does not require initial conditions.
Specific example - Second-order ODE with step input
When \( s_1 = 1 \) and \( s_2 = 21 \) are selected, the second-order ODE is

\[
\frac{d^2 y(t)}{dt^2} + 22 \frac{dy(t)}{dt} + 21 y(t) = 21 x(t)
\]

with \( y(0) = 0, \frac{dy(t)}{dt} \bigg|_{t=0} = 0 \), and \( x(t) = u(t) \).

The analytic solution is then

\[
y(t) = -1.05 e^{-t} + 0.05 e^{-21t} + 1, \quad t \geq 0.
\]

With \( T = 0.1 \) s, the forward-looking Euler numerical solution (text method)

\[
y[n] + \left[-2 + (s_1 + s_2)T\right]y[n-1] + \left[1 - (s_1 + s_2)T + s_1 s_2 T^2\right]y[n-2] = s_1 s_2 T^2 u[n]
\]

becomes

\[
y[n] + 0.2 y[n-1] - 0.99 y[n-2] = 0.21 u[n-2]
\]

for \( n \geq 1 \). The initial conditions are \( y[0] = y[-1] = x[-1] = 0 \) and \( x[0] = 1 \).

With \( T = 0.1 \) s, the backward-looking Euler numerical solution

\[
y[n] + \left(\frac{-2 - (s_1 + s_2)T}{1 + (s_1 + s_2)T + s_1 s_2 T^2}\right)y[n-1] + \left(\frac{1}{1 + (s_1 + s_2)T + s_1 s_2 T^2}\right)y[n-2] = \frac{s_1 s_2 T^2}{1 + (s_1 + s_2)T + s_1 s_2 T^2} u[n]
\]

becomes

\[
y[n] - 1.23167 y[n-1] + 0.293255 y[n-2] = 0.0615836 u[n] \quad \text{for} \quad n \geq 1.
\]

The initial conditions for \( y[n] \) are \( y[-1] = y[0] = 0 \). Since there are no past terms for \( x[n] \), it does not require initial conditions.

The corresponding MATLAB m-files to implement these solutions and the corresponding results are attached on the following pages.
% Numerical ODE Solution Example (chap2_2ODE_euler_soln_fwd.m)
%
% For a second-order ordinary differential equation (ODE)
% \[ \frac{d^2y}{dt^2} + (s_1+s_2)\frac{dy}{dt} + (s_1s_2) = \frac{u(t)}{s_1s_2} \]
% where \( y(0) = dy(0)/dt = 0 \) which has an analytic solution-
% \[ y(t) = A_1e^{(-s_1t)} + A_2e^{(-s_2t)} + 1 \]
% where \( A_1 = -s_2/(s_2-s_1) \) and \( A_2 = s_1/(s_2-s_1) \),
% find approximate numerical solution by using a forward-difference
% Euler's approximation for derivatives to change it into a
% second-order difference equation which can be solved
% recursively. Compare numerical results with exact solution.
%
close all; clear; clc;
% *** define ODE variables ***
s1 = 1; s2 = 21;
tstop = 3*max(1/s1,1/s2); % How far to go in time in seconds
% *** Forward-difference Euler approximation ***
T = 0.1; % Time step for numerical approximation
a1 = -2+(s1+s2)*T;
a2 = 1-(s1+s2)*T+s1*s2*T*T;
a=[a1, a2]; % a = \([a_1 a_2]\) coeff. vector
b=[0,0,s1*s2*T*T]; % b = \([b_0,b_1,b_2]\) coeff. vector
n=1:1:round(tstop/T); % Define index vector for recur()
x=ones(1,length(n)); % unit step input
x0=[0,1]; y0=[0,0]; % initial conditions
y=recur(a,b,n,x,x0,y0); % yields output for n=1,2,3,...
yapprox=[0,y]; n=[0,n]; % tack on values at t=nT=0
% *** Analytic solution ****
A1 = -s2/(s2-s1); A2 = s1/(s2-s1); t=0:0.01:tstop;
yexact=1+A1*exp(-s1*t)+A2*exp(-s2*t);
plot(t,yexact,'r',n*T,yapprox,'b.';
legend(' y_{exact}','y_{fwd,approx} w/ T = ',num2str(T),' s','Location','NW'); axis([0 tstop -0.1 2.1])
ylabel('y(t)','fontsize',16,'fontname','times')
xlabel('t (s)','fontsize',16,'fontname','times')
title({'Second-order Differential Eqn example';...'Forward-difference Euler approximation (Text)'},'fontsize',16,'fontname','times')
text(0.1,1.52,['d^{2}y/dt^2 + ',num2str(s1+s2),'dy/dt + ',num2str(s1*s2),'y = ',num2str(s1*s2),'u(t)'],'fontsize',14,'fontname','times')
text(0.1,1.3,['y(t) = ',num2str(A1),'e^{-',num2str(s1),'t} + ',num2str(A2),'e^{-',num2str(s2),'t} + 1 \ t \geq 0'],'fontsize',14,'fontname','times')
set(findobj('type','line'),'linewidth',1.5)
set(findobj('type','line'),'markersize',12)
set(findobj('type','axes'),'linewidth',2)
% Numerical ODE Solution Example (chap2_2ODE_euler_soln_bwd.m)
%
% For a second-order ordinary differential equation (ODE)
% \( \frac{d^2y}{dt^2} + (s_1+s_2)\frac{dy}{dt} + (s_1s_2) = \frac{u(t)}{(s_1s_2)} \)
% where \( y(0) = \frac{dy(0)}{dt} = 0 \) which has an analytic solution-
% \( y(t) = \frac{A_1}{s_2-s_1}e^{-(s_1-t)} + \frac{A_2}{s_2-s_1}e^{-(s_2-t)} + 1 \)
% where \( A_1 = -\frac{s_2}{s_2-s_1} \) and \( A_2 = \frac{s_1}{s_2-s_1} \),
% find approximate numerical solution by using a backward-difference
% Euler's approximation for derivatives to change it into a
% second-order difference equation which can be solved
% recursively.  Compare numerical results with exact solution.

close all; clear; clc;
% *** define ODE variables ***
s1 = 1; s2 = 21;
tstop = 3*max(1/s1,1/s2);  % How far to go in time in seconds
% *** Backward-difference Euler approximation ***
T = 0.1;  % Time step for numerical approximation
a1 = (-2-(s1+s2)*T)/(1 + (s1+s2)*T + s1*s2*T*T);
a2 = 1/(1 + (s1+s2)*T + s1*s2*T*T);
a=[a1, a2];  % coefficient vector for \( y[] \) terms
b=[(s1*s2*T*T)/(1 + (s1+s2)*T + s1*s2*T*T)];  % \([b0]\) coeff. for \( x[n]\)

n=1:1:round(tstop/T);  % Define index vector for recur()
x=ones(1,length(n));  % unit step input
x0=[]; y0=[0,0];  % initial conditions

y=recur(a,b,n,x,x0,y0);  % yields output for \( n=1,2,3,... \)
yapprox=[0,y];  % tack on values at \( t=nT\) = \( 0 \)
% *** Analytic solution ****
A1 = -s2/(s2-s1); A2 = s1/(s2-s1); t=0:0.01:tstop;
yexact=1+A1*exp(-s1*t)+A2*exp(-s2*t);

plot(t,yexact,'r',n*T,yapprox,'b.'
legend(' \( y_{\text{exact}} \)', [' \( y_{\text{bwd,approx}} \) w/ \( T = \),num2str(T),', ' s\)'],...
'Location','NW'),
axis([0 tstop -0.1 2.1])
ylabel('\( y(t) \)','fontsize',16,'fontname','times')
xlabel('\( t \) (s)','fontsize',16,'fontname','times')
title({'Second-order Differential Eqn example';...
'Backward-difference Euler approximation'},'fontsize',16,'fontname','times')
text(0.1,1.3,['\( y(t) = \),num2str(A1),'\( e^{-\text{\( s_1 \}\( t \)}} \),num2str(A2),'\( e^{-\text{\( s_2 \}\( t \)}} \) + 1 \( t \geq 0 \)'],$
'fontsize',14,'fontname','times')
set(findobj('type','line'),'linewidth',1.5)
set(findobj('type','line'),'markersize',12)
set(findobj('type','axes'),'linewidth',2)
Second-order Differential Eqn example
Forward-difference Euler approximation (Text)

\[ \frac{d^2y}{dt^2} + 22\frac{dy}{dt} + 21y = 21u(t) \]
\[ y(t) = -1.05e^{-1t} + 0.05e^{-21t} + 1 \quad t \geq 0 \]

Second-order Differential Eqn example
Backward-difference Euler approximation

\[ \frac{d^2y}{dt^2} + 22\frac{dy}{dt} + 21y = 21u(t) \]
\[ y(t) = -1.05e^{-1t} + 0.05e^{-21t} + 1 \quad t \geq 0 \]

- Obviously, the **forward**-looking Euler numerical solution (text method) is unstable. Whereas, the **backward**-looking Euler numerical solution is stable.