Process model formulation and solution, 3E4

Tutorial 7

Kevin Dunn, dunnkg@mcmaster.ca **November 2010**

Tutorial objectives: Numerical differentiation and plotting.

Question 1 [2]

This question uses the experimental data from the class notes. A batch system was charged with reactant A at time $t = 0$ at a concentration of 1.0 mol/L. As the assumed first-order reaction evolved, the concentrations were recorded:

Time (minutes)	Concentration (mol/L)
0	1.00
5	0.84
10	0.72
20	0.57
30	0.47
45	0.37
60	0.30

The dynamic balance for this system can be shown to be $\frac{dC_\mathsf{A}(t)}{dt} = -kC_\mathsf{A}(t)$.

1. Reproduce the above table and add a column that provides an approximation of $\frac{dC_\mathsf{A}(t)}{dt}$ at each time step.
2. Add a fourth column that lists the order of your approximation listed in part 1.
3. Use a 2nd order Lagrange polynomial to approximate the $C_\mathsf{A}(t)$ data, and use this polynomial to estimate $C_\mathsf{A}{}'(t)$ at time $t = 15$ minutes.
4. Can you provide an $O(h^2)$ estimate of $\frac{dC_\mathsf{A}(t)}{dt}$ at $t = 0$? If so, please calculate it.

Solution

A table showing the derivative approximation, and the order:

Time (minutes)	Concentration (mol/L)	Type of approximation	Approximation	Order
0	1.00	Forward	$\frac{0.84-1.00}{5-0} = -0.032$	$O(h)$
5	0.84	Central	$\frac{0.72-1.00}{2(10-5)} = -0.028$	$O(h^2)$
10	0.72	Central	$\frac{0.57-1.00}{2(20-10)} = -0.0215$	$O(h^2)$
20	0.57	Central	$\frac{0.47-0.72}{2(30-20)} = -0.0125$	$O(h^2)$
30	0.47	Backward	$\frac{0.47-0.57}{30-20} = -0.01$	$O(h)$
45	0.37	Central	$\frac{0.30-0.47}{2(60-45)} = -0.005667$	$O(h^2)$
60	0.30	Backward	$\frac{0.30-0.37}{60-45)} = -0.004667$	$O(h)$

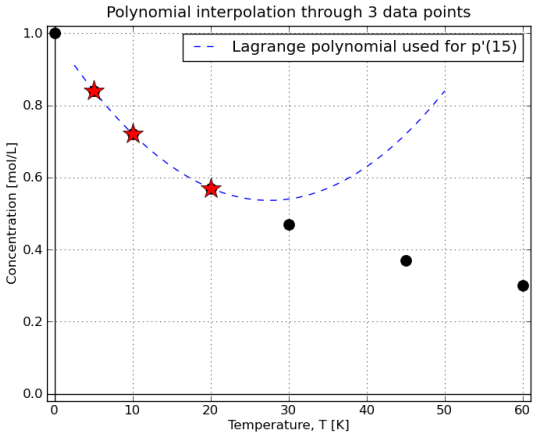
Note that the table uses an $O(h^2)$ estimate where possible; you should always prefer an $O(h^2)$ estimate over an $O(h)$ estimate, if it is possible to calculate one.

In general, one should fit a polynomial using the points closest to where you want to interpolate or use that polynomial. [There are exceptions of course: if one of the points was clearly noisy, relative to the others, you can skip that point and use another instead.] In this case one should fit the polynomial using points $t = (5, 10, 20)$, and their corresponding

concentrations, $C_\mathrm{A} = (0.84, 0.72, 0.57)$. Although in this case it didn't change the answer by much using points at $t = (0, 10, 20)$.

$$
\begin{aligned}
P_2(x) &= y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x) \\
P_2'(x) &= y_1\ell_1'(x) + y_2\ell_2'(x) + y_3\ell_3'(x) \\
&= 0.84 \cdot \frac{2x - 10 - 20}{(5-10)(5-20)} + 0.72 \cdot \frac{2x - 5 - 20}{(10-5)(10-20)} + 0.57 \cdot \frac{2x - 5 - 10}{(20-5)(20-10)} \\
P_2'(15) &= 0.84 \cdot 0 + 0.72 \cdot \frac{5}{-50} + 0.57 \cdot \frac{15}{150} = -0.15
\end{aligned}
$$

The polynomial, superimposed on the data points is shown below:



An $O(h^2)$ estimate can be found at $t = 0$ when using Richardson's extrapolation with $h = 10$ (full step), and $h = 5$ (half step).

- $h = 10$: $Q_1(h) = \frac{0.72 - 1.00}{10 - 0} = -0.028$
- $h = 5$: $Q_1(h/2) = \frac{0.84 - 1.00}{5 - 0} = -0.032$
- Combine these to calculate a $Q_2(h) = 2Q_1(h/2) - Q_1(h) = 2(-0.032) - (-0.028) = -0.036$

Question 2 [1.5]

A second order reaction in a constant-volume batch reactor has the following time-dependent behaviour:

$$C_\mathrm{A}(t) = \frac{C_\mathrm{A}(0)}{C_\mathrm{A}(0)kt + 1} \qquad \text{with} \qquad k = 0.002 \frac{\mathrm{m}^3}{\mathrm{mol.s}} \qquad \text{and} \qquad C_\mathrm{A}(0) = 2.5 \frac{\mathrm{mol}}{\mathrm{m}^3}$$

Starting with $h = 1.0$ and decreasing in powers of 10, which step size, $h = \Delta t$, provides the lowest error estimate of $\left.\frac{dC_\mathrm{A}}{dt}\right|_{t=100}$ at $t = 100$ seconds?

Present your results in tabular form with these 5 columns:

- Step size, $h = \Delta t$
- Forward difference approximation
- Error in the forward difference approximation
- Central difference approximation
- Error in the central difference approximation

Solution

Note: the last column was intended to be the error for the *central difference* approximation; an earlier version of the tutorial asked for the *backward difference* error. Either values will be accepted for the solution.

Given the above equation for $C_\mathsf{A}(t)$, the analytical derivative can be found as:

$$\frac{dC_\mathsf{A}(t)}{dt} = -k\frac{C_\mathsf{A}^2(0)}{\left(C_\mathsf{A}(0)kt+1\right)^2} = -kC_\mathsf{A}^2(t)$$

which is expected for a second-order reaction system. So the true derivative at $t = 100$ seconds is:

$$\frac{dC_\mathsf{A}(100)}{dt} = -0.002\frac{2.5^2}{(2.5^2 \times 0.002 \times 100 + 1)^2} = -0.005556$$

The required table is shown below. Two extra columns for the relative errors have been added, since that is arguably more useful than the absolute error (which was asked for in the question).

h	Forward diff	Fwd diff error	Fwd relative error[%]	Central diff	Central diff error	Central relative error[%]
1	-0.0055371	1.8457×10^{-5}	-0.33223	-0.0055556	6.1729×10^{-8}	-0.0011111
0.1	-0.0055537	1.8512×10^{-6}	-0.033322	-0.0055556	6.1728×10^{-10}	-1.1111×10^{-5}
0.01	-0.0055554	1.8518×10^{-7}	-0.0033332	-0.0055556	6.1648×10^{-12}	-1.1097×10^{-7}
1×10^{-3}	-0.0055555	1.8518×10^{-8}	-0.00033333	-0.0055556	1.4742×10^{-13}	$\mathbf{-2.6535 \times 10^{-9}}$
1×10^{-4}	-0.0055556	1.8496×10^{-9}	-3.3293×10^{-5}	-0.0055556	1.1466×10^{-12}	-2.0639×10^{-8}
1×10^{-5}	-0.0055556	1.6206×10^{-10}	-2.917×10^{-6}	-0.0055556	1.558×10^{-11}	-2.8043×10^{-7}
1×10^{-6}	-0.0055556	1.558×10^{-11}	$\mathbf{-2.8043 \times 10^{-7}}$	-0.0055556	1.558×10^{-11}	-2.8043×10^{-7}
1×10^{-7}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}
1×10^{-8}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}
1×10^{-9}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}	-0.0055556	4.5967×10^{-10}	-8.274×10^{-6}
1×10^{-10}	-0.0055578	2.2209×10^{-6}	-0.039976	-0.0055567	1.1107×10^{-6}	-0.019992
1×10^{-11}	-0.0055733	1.7764×10^{-5}	-0.31975	-0.0055622	6.6618×10^{-6}	-0.11991
1×10^{-12}	-0.0055511	4.4404×10^{-6}	-0.079928	-0.0054401	0.00011546	-2.0783

What's remarkable in the above table is how accurate the central different approximation is, even using very large values of h. The point here is that the central difference should be used instead of the forward or backward difference, wherever possible.

Furthermore, when using the central difference approximation with very small step sizes we start to accumulate round-off error, but no faster than the error we would have accumulated with the forward difference approximation.

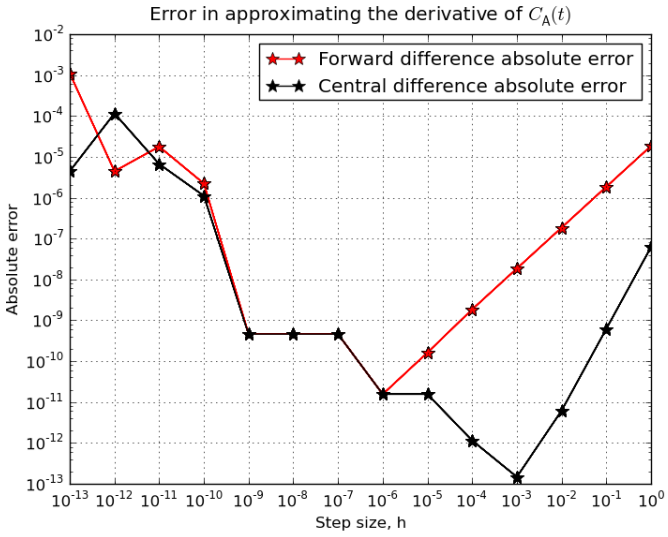
Bonus question [0.5]

Show the two error columns from question 2 in graphical form, using log-axes for both the x- and y-axis. Search for help over the internet if you don't know how to plot log-plots in MATLAB, or with Python's `matplotlib` library. Add a legend, grid lines and axis labels to the plot.

Solution

Note: A clear plot, showing the points, lines connecting the points (different colours or dashed/solid lines), labels, legend on a log-log axis are required to obtain full grade for this question.

A log-log plot with the required labels, for the absolute errors is shown here. The code to generate the plot is below.



MATLAB code

```
CA0 = 2.5;
rate_k = 0.002;

% The function and its first derivative
func = @(t, CA0, rate_k) CA0 ./ (CA0.*rate_k.*t + 1);
% an anonymous function of an anonymous function!
f = @(t)func(t, CA0, rate_k);
df = @(t, CA0, rate_k) - rate_k*CA0^2 / (CA0*rate_k*t + 1)^2;

h_values = 10 .^ [-0.0 : -1.0 : -13.0];
fprintf('\n %8s %10s %12s %16s %13s %16s %16s \n','h', ...
        'Fwd diff', 'Fwd diff error', 'Fwd rel error[%]', ...
        'Central diff', 'Ctl diff error', 'Ctl rel error[%]')

% time at which we are evaluating the derivative
t = 100.0;
true_derivative = df(t, CA0, rate_k);

% Initialize storage to plot the error vectors:
n = numel(h_values);
fwd_diff_approx = zeros(n, 1);
fwd_error = zeros(n, 1);
fwd_error_rel = zeros(n, 1);

central_diff_approx = zeros(n, 1);
central_error = zeros(n, 1);
central_error_rel = zeros(n, 1);

i=1;
for h = h_values
    fwd_diff_approx(i) = (f(t+h) - f(t))/h;
    fwd_error(i) = abs(fwd_diff_approx(i) - true_derivative);
    fwd_error_rel(i) = fwd_error(i)/true_derivative * 100.0;

    central_diff_approx(i) = (f(t+h) - f(t-h))/(2*h);
    central_error(i) = abs(central_diff_approx(i) - true_derivative);
    central_error_rel(i) = central_error(i) / true_derivative * 100.0;
```

```
    fprintf('\n %5f %10f %14g %16g %13f %16g %16g',...
            h, fwd_diff_approx(i), fwd_error(i), fwd_error_rel(i), ...
            central_diff_approx(i), central_error(i), central_error_rel(i))

    i = i + 1;
end


loglog(h_values, fwd_error, '-r*', 'markersize', 10)
hold on
loglog(h_values, central_error, '-k*', 'markersize', 10)
legend('Forward difference absolute error', ...
       'Central difference absolute error')
grid('on')
xlabel('Step size, h')
ylabel('Absolute error')
title('Error in approximating the derivative of C_A(t)')
% Save the plot
print('-dpng', '../images/tut7_qbonus_errors_MATLAB.png')
```

Python code

```
import numpy as np
from matplotlib.pylab import *

# Global constants
CA0 = 2.5
rate_k = 0.002

def f(time_t):
    return CA0 / (CA0*rate_k*time_t + 1)

def df(time_t):
    return - rate_k*CA0**2 / (CA0*rate_k*time_t + 1)**2

h_values = 10**(np.arange(-0.0, -14, -1.0))

print('{0:10} {1:14} {2:15} {3:15} {4:14} {5:15} {6:15}'\
    .format('h', 'Fwd diff', 'Fwd diff error', 'Fwd rel error[%]', \
    'Central diff', 'Ctl diff error', 'Ctl rel error[%]'))

t = 100.0 # time at which we are evaluating the derivative
true_derivative = df(t)

# Initialize storage to plot the error vectors:
n = h_values.shape
fwd_diff_approx = np.zeros(n)
fwd_error = np.zeros(n)
fwd_error_rel = np.zeros(n)

central_diff_approx = np.zeros(n)
central_error = np.zeros(n)
central_error_rel = np.zeros(n)

for i, h in enumerate(h_values):
    fwd_diff_approx[i] = (f(t+h) - f(t))/h
    fwd_error[i] = np.abs(fwd_diff_approx[i] - true_derivative)
    fwd_error_rel[i] = fwd_error[i]/true_derivative * 100.0

    central_diff_approx[i] = (f(t+h) - f(t-h))/(2*h)
```

```
    central_error[i] = np.abs(central_diff_approx[i] - true_derivative)
    central_error_rel[i] = central_error[i] / true_derivative * 100.0

    print('{0:10g}{1:14.5g}{2:15.5g}{3:15.5g}{4:14.5g}{5:15.5g}{6:15.5g}'\
          .format(h, fwd_diff_approx[i], fwd_error[i], fwd_error_rel[i],
          central_diff_approx[i], central_error[i], central_error_rel[i]))

fig = figure(1)
loglog(h_values, fwd_error, '-r*', ms=10)
legend(['Forward difference absolute error',
        'Central difference absolute error'])
loglog(h_values, central_error, '-k*', ms=10)
grid('on')
xlabel('Step size, h')
ylabel('Absolute error')
title('Error in approximating the derivative of $C_{\sf A}(t)$')
# Save the plot
fig.savefig('../images/tut7_qbonus_errors.png')
```

END