Solution set 7:

Higher-order methods and challenges

Exercise 7.1 **(a)** In the limit of the method we know that the error behaves as $\mathcal{O}\left(h^{\frac{1}{2}}\right)$. Writing the error $E_{after} \sim h_{after}^{\frac{1}{2}}$ and $E_{before} \sim h_{before}^{\frac{1}{2}}$ with some constant relating the before and after grids $h_{before} = c \cdot h_{after}$, we have $\frac{E_{before}}{E_{after}} = \frac{h_{before}^{\frac{1}{2}}}{h_{after}^{\frac{1}{2}}} = \sqrt{c}$, so $\frac{E_{before}}{E_{after}} = 10 \Rightarrow c = 100$. In other words, increasing the accuracy by a factor of 10 requires increasing the resolution with a factor of 100. **(b)** The amount of grid-points in each dimension is proportional to $\frac{1}{h}$, and the memory footprint is directly proportional to the number of grid-points. Assume $h \sim k$. We can write the change in memory as $\frac{M_{after}}{M_{before}} = \frac{\left(\frac{1}{h_{after}}\right)^4}{\left(\frac{1}{h_{before}}\right)^4} = \left(\frac{h_{before}}{h_{after}}\right)^4 \sim 10^8$. **(c)** Yes, this is a huge problem. Even if we do not need to save all the data generated in the integration, the footprint of a single solution state will be a factor of 10^6 larger. Say your 3d problem has a memory footprint of a roughly a gigabyte, easily residing on your laptop, trying to improve the accuracy of your solution by an order of magnitude would suddenly require a large scale cluster for storage of each solution state.

Exercise 7.2 **(a)** Here we derive the coefficients for the centered 3 point stencil

$$\frac{d}{dx}u(x) = au(x-h) + bu(x) + cu(x+h) \tag{1}$$

Do Taylor expansion in x to get

$$\begin{aligned}\frac{d}{dx}u(x) =& a\left(u(x) - hu'(x) + \frac{1}{2}h^2u''(x) - \frac{1}{6}h^3u'''(x) + \mathcal{O}\left(h^4\right)\right) + bu(x) \\ &+ c\left(u(x) + hu'(x) + \frac{1}{2}h^2u''(x) + \frac{1}{6}h^3u'''(x) + \mathcal{O}\left(h^4\right)\right)\end{aligned}$$

Collecting the coefficients for derivatives in x yields

$$\frac{d}{dx}u(x) = (a+b+c)\,u(x) + (c-a)\,hu'(x) + \left(\frac{1}{2}a + \frac{1}{2}c\right)h^2u''(x) + \left(\frac{1}{6}c - \frac{1}{6}a\right)h^3u'''(x) + \mathcal{O}\left(h^4\right) \tag{2}$$

We choose (a,b,c) so that $(a+b+c) = 0$, $(c-a) = \frac{1}{h}$, $\left(\frac{1}{2}a + \frac{1}{2}c\right) = 0$, i.e., $(a,b,c) = \left(-\frac{1}{2h}, 0, \frac{1}{2h}\right)$. Inserting in eq. 3 we have

$$\frac{d}{dx}u(x) = u'(x) + \frac{1}{6}h^2u'''(x) + \mathcal{O}\left(h^3\right) \tag{3}$$

We can then write the stencil that approximate the partial derivative of x of our solution as

$$\frac{d}{dx}u(x_i) \approx \frac{1}{2h}\left(U_{i+1} - U_{i-1}\right) \tag{4}$$

To derive the coefficients and order of accuracy of a centered 5-point stencil, follow the same procedure to arrive at

$$\frac{d}{dx}u(x_i) \approx \frac{1}{3h}\left(\frac{1}{4}U_{i-2} - 2U_{i-1} + 2U_{i+1} - \frac{1}{4}U_{i+2}\right) \tag{5}$$

(b) From eq. 3 we see directly that the approximation is $\mathcal{O}\left(h^2\right)$, the 5 point stencil in Eq. 5 is $\mathcal{O}\left(h^4\right)$. **(c)** See matlab/Octave code attached. **(d)** Using the attached Matlab code we arrive at the results presented in table 1. **(e)** Notice how more grid-points are needed to satisfy the accuracy requirements when using the $\mathcal{O}\left(h^2\right)$ method compared to the $\mathcal{O}\left(h^4\right)$ spacial discretization. So this means that for this problem the higher order method is always better? Not necessarily. Using the $\mathcal{O}\left(h^4\right)$ stencil to approximate the x-derivative at each stencil point is twice as expensive as using the $\mathcal{O}\left(h^2\right)$ stencil. In table 1c we account for this by multiplying the results in table 1b by two and then dividing by the numbers in table 1a by this. Notice that the lower order method is actually computationally less expensive when integrating a short interval with low accuracy,

$E_{max} \setminus T_{end}$	$\frac{1}{4}wave$	$5waves$
10^{-1}	12	54
10^{-3}	120	536

(a) Using the $\mathcal{O}\left(h^2\right)$ stencil eq. 4

$E_{max} \setminus T_{end}$	$\frac{1}{4}wave$	$5waves$
10^{-1}	8	19
10^{-3}	32	67

(b) Using the $\mathcal{O}\left(h^4\right)$ stencil eq. 5

Table 1a / 2 * 1b	$\frac{1}{4}wave$	$5waves$
10^{-1}	0.75	1.42
10^{-3}	1.88	4.00

(c) The relative computational complexity

Table 1: The number of grid-points in x needed to obtain an error less than E_{max} when solving the PDE over T_{end} waves. The error was measured in the 1-norm.

conversely, when integrating a long interval or high accuracy is required, the computational complexity of the higher order method is smaller. The important thing to note here is that using higher order methods can be substantially cheaper both computationally and memory wise when high accuracy or long time integration is of interest.

```
clear all,clc
% Code to solve the advection equation presented in exercise set 7
% for the course Numerical methods for conservation laws.
% The code solves the equation to Tend repeatedly in the while
% loop untill the accuracy satisfies emax.

% Find elements needed to produce error less than
Tend = [0.25,5];
emax = [10^(-1);10^(-3)];

% Do magic
elements = zeros(2,2);
for j = 1:2 % Loop over Tend
    [Uref,Xref] = getRef(1000,Tend(j));
    for e = 1:2 % Loop over emax
        E = 1;
        elements(e,j) = 5;
        while E>emax(e) && elements(e,j) < 800
            elements(e,j) = elements(e,j) + 1;
            % Setup problem
            [U,X,nX,nT,h,k,a] = setup(elements(e,j),Tend(j));
            % Get matrix to approximate derivative
            A = order2(nX,h);
            % Integrate using 4th order RK
            for i = 1:nT
                U = RK4(U,A,k,a);
            end
            E = norm(U-interp1(Xref,Uref,X)',1)/nX;
        end
    end
end
```

```
function [ Uref,Xref ] = getRef(elements,Tend)
name = ['Uref',num2str(elements),'_',num2str(Tend),'.mat'];
if exist(name, 'file') == 2 % Check if reference solution exist
    load(name);
else % Else calculate reference solution
    [Uref,Xref,nX,nT,h,k,a] = setup(elements,Tend);
    C = order4(nX,h);
    wh = waitbar(0,'Calculating reference solution');
    for i = 1:nT
        Uref = RK4(Uref,C,k,a);
        waitbar(i/nT)
    end
    close(wh)
    save(name,'Uref','Xref');
end
end
```

```matlab
function [U,X,nX,nT,h,k,a] = setup(elements,Tend)
% Define and return discretization and initial condition
a = -2*pi;
h = 2*pi/elements;
X = 0:h:(2*pi-h);
nX = numel(X);
k = h/5;
nT = round(Tend/k);
k = Tend/nT;
U = exp(sin(X))';
end
```

```matlab
function [ U ] = RK4(U,A,k,a)
% Use the classical 4th order Runge Kutta method to integrate
% over a timestep k
Y1 = U;
Y2 = U + 0.5*k*a*A*Y1;
Y3 = U + 0.5*k*a*A*Y2;
Y4 = U + k*a*A*Y3;
U = U + (a*k/6)*(A*Y1+2*A*Y2+2*A*Y3+A*Y4);
end
```

```matlab
function [ B ] = order2(nX,h)
% Set up matrix to approximate the derivative with a second order stencil
B = diag((1/(2*h))*ones(1,nX-1),1) + diag(-1/(2*h)*ones(1,nX-1),-1);
B(1,nX) = -1/(2*h);
B(nX,1) = -B(1,nX);
end
```

```matlab
function [ C ] = order4(nX,h)
% Set up matrix to approximate the derivative with a fourth order stencil
c1          = 1/(12*h);
c2          = -2/(3*h);
c3          = 0;
c4          = 2/(3*h);
c5          = -1/(12*h);
% Stencil
C = diag(c1*ones(1,nX-2),-2) + diag(c2*ones(1,nX-1),-1);
C = C + diag(c4*ones(1,nX-1),1) + diag(c5*ones(1,nX-2),2);
% Periodic boundary conditions
C(1,nX)     = c2;
C(1,nX-1)   = c1;
C(2,nX)     = c1;
C(nX,1)     = c4;
C(nX-1,1)   = c5;
C(nX,2)     = c5;
end
```