Assignment 10

1. Use the finite-difference method with $h = 0.2$ to approximate a solution to the boundary value problem

$$u^{(2)}(x) = 0.3u^{(1)}(x) + 0.1u(x) - x - 0.2$$
$$u(0) = 2$$
$$u(1) = 3$$

First, rewrite the ODE as $u^{(2)}(x) - 0.3u^{(1)}(x) - 0.1u(x) = -x - 0.2$. Next, we note that we can determine p, q and r as follows:

```
a2 =  1.0;
a1 = -0.3;
a0 = -0.1;
u_a = 2;
u_b = 3;
h = 0.2;
% This is a LODE with constant coefficients, so
% all three values 'p', 'q' and 'r' are also constant:
p =  2*a2 -   a1*h;
q = -4*a2 + 2*a0*h^2;
r =  2*a2 +   a1*h;

% This is the forcing function
g = @(x)(-x - 0.2);
% This is the corresponding matrix
A = [q r 0 0
     p q r 0
     0 p q r
     0 0 p q];
%                               2
% The target vector is 2*g(x ) h  for k = 1, 2, 3, n - 1.
%                           k
b = 2*g( [0.2 0.4 0.6 0.8]' )*h^2
% We have two Dirichlet boundary conditions, so we must update
% the first and last vector entries:
b(1)   = b(1)   - p*u_a;
b(end) = b(end) - r*u_b;
% Solve the system of linear equations
u = A \ b;
% Add back the values at 0 and 1:
u = [2; u; 3]
      u = 2.0000
          2.2041
          2.4135
          2.6210
          2.8192
          3.0000
```

2. Use the finite-difference method with $h = 0.2$ to approximate a solution to the boundary value problem

$$u^{(2)}(x) = 0.3u^{(1)}(x) + 0.1u(x) - x - 0.2$$
$$u^{(1)}(0) = 0$$
$$u(1) = 3$$

The set up is similar, but now we have

```
% This is the corresponding matrix
A = [q r 0 0
     p q r 0
     0 p q r
     0 0 p q];
%                              2
% The target vector is 2*g(x ) h  for k = 1, 2, 3, n - 1.
%                           k
b = 2*g( [0.2 0.4 0.6 0.8]' )*h^2
% We have one insulated boundary condition, so we must update
% the first row of the matrix:
A(1, 1) = A(1, 1) + (4.0/3.0)*p;
A(1, 2) = A(1, 2) - (1.0/3.0)*p;

% We have one two Dirichlet boundary conditions, so we must update
% the last vector entries:
b(end) = b(end) - r*u_b;
% Solve the system of linear equations
u = A \ b;
% Add back the values at x = 0 and x = 1, but at the left, we must
% have points that have a slope 0 at x = 0.
u = [(4.0/3.0)*u(1) - (1.0/3.0)*u(2); u; 3]
      u = 3.1341
          3.1323
          3.1267
          3.1090
          3.0700
          3.00000
```

3. Given the function $u(\mathbf{x}, t) = t\, x_1 x_2 - x_1 + 2x_2$, approximate the partial derivative with respect to time and the gradient at the point $t = 0.2$ and $\mathbf{x} = \begin{pmatrix} 0.3 \\ -0.5 \end{pmatrix}$ using a value of $\Delta t = h = 0.1$.

This can be done by hand, but here it is with Matlab:

```
u = @(x, t)( t*x(1)*x(2) - x(1) + 2*x(2) );
t = 0.2;
x = [0.3 -0.5]';
dt = 0.1;
h  = 0.1;
% The partial w.r.t. time
(u(x, t + dt) - u(x, t - dt))/(2*dt)
      ans = -0.15000
% The gradient, or the partials with respect to the first and second
% entries of the space argument
[(u(x + h*[1 0]', t) - u(x - h*[1 0]', t))/(2*h)
 (u(x + h*[0 1]', t) - u(x - h*[0 1]', t))/(2*h)]
      ans = -1.1000
             2.0600
```

Note, because the function is linear in each of the variables, these approximations are actually also the exact values. In Maple:

```
# Define a multi-variate function 'u'
u := (x, y, t ) -> t*x*y - x + 2*y:
# Calculate the partial derivative with respect to the third variable:
D[3](u)(0.3, -0.5, 0.2);
```

$$-0.15$$

```
# Calculate the gradient, or a vectors of the partials w.r.t. the first
# second variables
<D[1](u)(0.3, -0.5, 0.2), D[2](u)(0.3, -0.5, 0.2)>;
```

$$\begin{pmatrix} -1.10 \\ 2.06 \end{pmatrix}$$

4. Given the function in Question 3, approximate all three second partials: with respect to t, x_1, and x_2.

Because all these are linear in each of the variables, the concavity everywhere is zero, and this is shown by the approximations:

```
u = @(t, x)( t*x(1)*x(2) - x(1) + 2*x(2) );
t = 0.2;
x = [0.3 -0.5]';
dt = 0.1;
h  = 0.1;
% Approximate the second partial w.r.t. time
(u(t + dt, x) - 2*u(t, x) + u(t - dt, x))/(dt^2)
       ans = 2.2204e-14
% Approximate the second partial w.r.t. the first space variable
(u(t, x + h*[1 0]') - 2*u(t, x) + u(t, x - h*[1 0]'))/(h^2)
       ans = 2.2204e-14
% Approximate the second partial w.r.t. the second space variable
(u(t, x + h*[0 1]') - 2*u(t, x) + u(t, x - h*[0 1]'))/(h^2)
       ans = 0
```

5. In class, we did not discuss an explicit formula for $\frac{\partial^2}{\partial x \partial y} u(x, y)$. Come up with such a formula. Show that your formula works by calculating this second partial explicitly using the process shown in your calculus course for the function in $xe^x\sin(y)$ at $x = 1$ and $y = 2$, and then calculating your approximation using $h = 0.01$.

The partial with respect to the first variable is

$$\begin{aligned}
\frac{\partial^2}{\partial x \partial y} u(x, y) &= \frac{\partial}{\partial x}\frac{\partial}{\partial y} u(x, y) \\
&\approx \frac{\partial}{\partial x}\frac{u(x, y+h) - u(x, y-h)}{2h} \\
&\approx \frac{\frac{u(x+h, y+h) - u(x+h, y-h)}{2h} - \frac{u(x-h, y+h) - u(x-h, y-h)}{2h}}{2h} \\
&= \frac{u(x+h, y+h) - u(x+h, y-h) - u(x-h, y+h) + u(x-h, y-h)}{4h^2}
\end{aligned}$$

The second partial with respect to y and then to x for the given function is $e^x\cos(y) + xe^x\cos(y)$, and we evaluate:

```
% The bivariate function
u = @(x, y)( x*exp(x)*sin(y) );
% The second partial w.r.t. the second variable, and then the first
ddu = @(x, y)( exp(x)*cos(y) + x*exp(x)*cos(y) );
h = 0.01;
% Our approximation at x = 1 and y = 2
(u(1 + h, 2 + h) - u(1 + h, 2 - h) - u(1 - h, 2 + h) + u(1 - h, 2 - h))/(4*h^2)
       ans = -2.262446473823010
% The actual second partial at x = 1 and y = 2
ddu(1, 2)
       ans = -2.262408767513627
```

6. Demonstrate that the formula for $\frac{\partial^2}{\partial x \partial y} u(x, y)$ that you found in Question 5 is the same formula you would find if you were to approximate $\frac{\partial^2}{\partial y \partial x} u(x, y)$.

$$
\begin{aligned}
\frac{\partial^2}{\partial y \partial x} u(x, y) &= \frac{\partial}{\partial y} \frac{\partial}{\partial x} u(x, y) \\
&\approx \frac{\partial}{\partial y} \frac{u(x+h, y) - u(x-h, y)}{2h} \\
&\approx \frac{\frac{u(x+h, y+h) - u(x-h, y+h)}{2h} - \frac{u(x+h, y-h) - u(x-h, y-h)}{2h}}{2h} \\
&= \frac{u(x+h, y+h) - u(x-h, y+h) - u(x+h, y-h) + u(x-h, y-h)}{4h^2}
\end{aligned}
$$

You will see that this is simply a rearrangement of the terms in the numerator of Question 5.

7. Approximate a solution to the heat equation with four steps in time if the boundary conditions are $u_a(t) = 0$ for $t > 0$ and $u_b(t) = 2$ for $t > 0$ and the initial state is $u_0(x) = 1 - x$ if the interval in space is [0, 1] and $h = 0.2$. The coefficient $\alpha = 4$. You should use a Δt, as described in the course notes, to ensure convergence.

If $\frac{\alpha\Delta t}{h^2} < \frac{1}{2}$, then find Δt so that $\frac{\alpha\Delta t}{h^2} = \frac{1}{4}$ or $\Delta t = \frac{h^2}{4\alpha}$, so us $\Delta t = 0.0025$.

To do this by hand, we observe that our space-interval is [0, 1] and $h = 0.2$, so $n_x = 5$, and the six x values are $x_0 = 0$, $x_1 = 0.2$, $x_2 = 0.4$, $x_3 = 0.6$, $x_4 = 0.8$ and $x_5 = 1$. Because the $u_0(x) = 1 - x$, we can therefore calculate $u_{0,0} = u_0(x_0) = u_0(0) = 1$, $u_{1,0} = u_0(x_1) = u_0(0.2) = 0.8$, $u_{2,0} = 0.6$, $u_{3,0} = 0.4$, $u_{4,0} = 0.2$, $u_{5,0} = 0$.

Thus, we have:

k	x_k	$u_{k,0}$			
0	0	1			
1	0.2	0.8			
2	0.4	0.6			
3	0.6	0.4			
4	0.8	0.2			
5	1	0			

Now, the boundary values are not turned on until after the first step, so the formula is just

$$u_{k,\ell+1} \leftarrow u_{k,\ell} + \alpha\Delta t\,(u_{k-1,\ell} - 2u_{k,\ell} + u_{k+1,\ell})/h^2,$$

and in this case, $\alpha\Delta t/h^2 = 4\cdot 0.0025/0.2^2 = 0.25$, so the calculation is

$$u_{k,\ell+1} \leftarrow u_{k,\ell} + 0.25(u_{k-1,\ell} - 2u_{k,\ell} + u_{k+1,\ell}).$$

We start by calculating the four interior points with $k = 1, 2, 3, 4$, and in all cases, the sum in the parentheses is equal to zero, so there is no change in the u values! For example, $1 - 2\cdot 0.8 + 0.6 = 0$. The two boundary values, however, now are changed.

Thus, we have:

k	x_k	$u_{k,0}$	$u_{k,1}$		
0	0	1	0		
1	0.2	0.8	0.8		
2	0.4	0.6	0.6		
3	0.6	0.4	0.4		
4	0.8	0.2	0.2	$u_{4,2}$	
5	1	0	2		

Once again, we calculate the four interior points:

$$u_{1,2} \leftarrow u_{1,1} + 0.25(u_{0,1} - 2u_{1,1} + u_{2,1}) = 0.8 + 0.25(0.0 - 2\cdot 0.8 + 0.6) = 0.55$$
$$u_{2,2} \leftarrow u_{2,1} + 0.25(u_{1,1} - 2u_{2,1} + u_{3,1}) = 0.6 + 0.25(0.8 - 2\cdot 0.6 + 0.4) = 0.6$$
$$u_{3,2} \leftarrow u_{3,1} + 0.25(u_{2,1} - 2u_{3,1} + u_{4,1}) = 0.4 + 0.25(0.6 - 2\cdot 0.4 + 0.2) = 0.4$$
$$u_{4,2} \leftarrow u_{4,1} + 0.25(u_{3,1} - 2u_{4,1} + u_{5,1}) = 0.2 + 0.25(0.4 - 2\cdot 0.2 + 2.0) = 0.7$$

As you may expect, it is getting cooler at the one end, and warmer at the other. There is no change to the boundary values.

Thus, we have:

k	x_k	$u_{k,0}$	$u_{k,1}$	$u_{k,2}$	
0	0	1	0	0	
1	0.2	0.8	0.8	0.55	
2	0.4	0.6	0.6	0.6	$u_{2,3}$
3	0.6	0.4	0.4	0.4	
4	0.8	0.2	0.2	0.7	
5	1	0	2	2	

Once again, we calculate the four interior points:

$$u_{1,3} \leftarrow u_{1,2} + 0.25(u_{0,2} - 2u_{1,2} + u_{2,2}) = 0.55 + 0.25(0.0 \ - 2\cdot 0.55 + 0.6) = 0.425$$
$$u_{2,3} \leftarrow u_{2,2} + 0.25(u_{1,2} - 2u_{2,2} + u_{3,2}) = 0.6 \ + 0.25(0.55 - 2\cdot 0.6 \ + 0.4) = 0.5375$$
$$u_{3,3} \leftarrow u_{3,2} + 0.25(u_{2,2} - 2u_{3,2} + u_{4,2}) = 0.4 \ + 0.25(0.6 \ - 2\cdot 0.4 \ + 0.7) = 0.525$$
$$u_{4,3} \leftarrow u_{4,2} + 0.25(u_{3,2} - 2u_{4,2} + u_{5,2}) = 0.7 \ + 0.25(0.4 \ - 2\cdot 0.7 \ + 2.0) = 0.95$$

Thus, we have, again with the same boundary values:

k	x_k	$u_{k,0}$	$u_{k,1}$	$u_{k,2}$	$u_{k,3}$
0	0	1	0	0	0
1	0.2	0.8	0.8	0.55	0.55
2	0.4	0.6	0.6	0.6	0.6
3	0.6	0.4	0.4	0.4	0.4
4	0.8	0.2	0.2	0.7	0.7
5	1	0	2	2	2

```
% To do this in Matlab, first, set up the problem
alpha = 4;
a = 0;
b = 1;
u_init = @(x)( 1 - x );
u_a = @(t)( 0 );
u_b = @(t)( 2 );

% Second, set up the x-values, so 0.0  0.2  0.4  0.6  0.8  1.0
h = 0.2;
Nx = 5;          % The number of intervals in x
xs = 0:h:1;
% Third, set up the time values, so 0.0  0.0025  0.0050  0.0075  0.01
dt = 0.0025;
Nt = 4;          % The number of intervals in t
ts = 0:dt:(4*dt);

% Set up the grid as a 2-d array
U = zeros( Nx + 1, Nt + 1 );
```

```
% We will assign the initial values at time t = 0 (in red) by calling
% the function u    (x ) and then with each time step, we will call the
%                init  k
% boundary value functions u (t    ) and u (t    ) to determine the values
%                           a  ell       b  ell
% at the blue and cyan points, respectively.

%   x
%    k
% a = 0.0   |   0         0         0         0         0
%     0.2   |   0         0         0         0         0
%     0.4   |   0         0         0         0         0
%     0.6   |   0         0         0         0         0
%     0.8   |   0         0         0         0         0
% b = 1.0   |   0         0         0         0         0
%           +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%           ell

% Thus, call and initialize the values at t  = 0
%                                          0
for k = 1:(Nx + 1)
    U(k,1) = u_init(xs(k));
end

% Now the 2d-array looks as follow, where the initial function is 1 - x
%   x
%    k
% a = 0.0   |  1.0        0         0         0         0
%     0.2   |  0.8        0         0         0         0
%     0.4   |  0.6        0         0         0         0
%     0.6   |  0.4        0         0         0         0
%     0.8   |  0.2        0         0         0         0
% b = 1.0   |  0.0        0         0         0         0
%           +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%           ell
```

```
for ell = 1:Nt
    for k = 2:Nx
        % Estimate the temperature at the four interior points
        %     The previous value
        %                       alpha dt
        %         plus the ratio ----------
        %                          2
        %                         h
        %         multiplied by the specified linear combination of
        %         the previous three values in space
        U(k, ell+1) = U(k, ell) + alpha*dt/h^2*(     ...
            U(k-1, ell) - 2*U(k, ell) + U(k+1, ell)  ...
        );

        % Evaluate the left-hand boundary (at a = 0) at
        % corresponding time
        U(1, ell+1) = u_a(ts(ell+1));

        % Similarly, evaluate the right-hand boundary (at b = 1)
        % at the corresponding time
        U(Nx+1, ell+1) = u_b(ts(ell+1));
    end
end
% After the first loop, because the initial state is in the steady
% state, there is no change, except for the two boundary values:
%   x
%    k
% a = 0.0   |  1.0       0.0       0         0         0
%     0.2   |  0.8       0.8       0         0         0
%     0.4   |  0.6       0.6       0         0         0
%     0.6   |  0.4       0.4       0         0         0
%     0.8   |  0.2       0.2       0         0         0
% b = 1.0   |  0.0       2.0       0         0         0
%           +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%           ell

% After the second loop, the left side at 'a' cools, while
% the right side at 'b' heats up:
%   x
%    k
% a = 0.0   |  1.0       0.0      0.0        0         0
%     0.2   |  0.8       0.8      0.55       0         0
%     0.4   |  0.6       0.6      0.6        0         0
%     0.6   |  0.4       0.4      0.4        0         0
%     0.8   |  0.2       0.2      0.7        0         0
% b = 1.0   |  0.0       2.0      2.0        0         0
%           +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%           ell
```

```
% After the third loop, the left side at 'a' continues to cool, while
% the right side at 'b' continues to heats up, however, the cooling
% and heating is continuing to propagate:
%   x
%    k
% a = 0.0    |  1.0       0.0       0.0       0.0       0
%     0.2    |  0.8       0.8       0.55      0.425     0
%     0.4    |  0.6       0.6       0.6       0.5375    0
%     0.6    |  0.4       0.4       0.4       0.525     0
%     0.8    |  0.2       0.2       0.7       0.95      0
% b = 1.0    |  0.0       2.0       2.0       0.0       0
%            +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%            ell

% After the fourth loop, heat propagation continues:
%   x
%    k
% a = 0.0    |  1.0       0.0       0.0       0.0      0
%     0.2    |  0.8       0.8       0.55      0.425    0.246875
%     0.4    |  0.6       0.6       0.6       0.5375   0.50625
%     0.6    |  0.4       0.4       0.4       0.525    0.634375
%     0.8    |  0.2       0.2       0.7       0.95     1.10625
% b = 1.0    |  0.0       2.0       2.0       2.0      2.0
%            +-------------------------------------------------
%          t    0      0.0025    0.0050    0.0075    0.0100
%            ell
```

Acknowledgement: Andy Liu for pointing out an index was incorrect in the solution to Question 2. Harsh Patel for asking me to add an expanded worked-out answer for Question 7. Dhyey Patel for noting that I accidentally referred to $f^{(1)}(x)$ instead of $u^{(1)}(x)$ in Question 1.