Solution.

**GRADERS:** Please scale all final scores to 100 points, rounded up. I.e. 70 out of 74 points should be entered in Canvas as a 95

*A reminder from the course syllabus:* Mathematically rigorous solutions are expected; strive for clarity and elegance. You may collaborate on the problems, but your write-up must be your own independent work. Transcribed solutions and copied MATLAB code are both unacceptable. *You may not consult solutions from previous sections of this class.* Unless it is specified that a particular calculation must be performed 'by hand,' you are encouraged to use MATLAB’s Symbolic Math Toolbox (or Mathematica/Wolfram Alpha/Maple) to compute and simplify tedious integrals and derivatives on the problem sets. As always, you must document your calculations clearly.

A total of 74 points is distributed among the following problems

1. [12 pts (2 ea)]
For each of the following equations, specify whether each is (a) an ODE or a PDE; (b) determine its order; (c) specify whether it is linear or nonlinear. For those that are linear, specify whether they are (d) homogeneous or inhomogeneous, and (e) whether they have constant or variable coefficients. For those that are 1st or 2nd order ODE, (f) find a general solution.

   \[
   \begin{align*}
   (1.1) & \quad \frac{dy}{dx} - \frac{2}{x} y = 3x \cdot \ln x \\
   (1.2) & \quad \frac{\partial v}{\partial t} + 3t^2 \frac{\partial v}{\partial x} - 5 \frac{\partial v}{\partial x} = 0 \\
   (1.3) & \quad \frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left[ u^2 \frac{\partial u}{\partial x} \right] = 0 \\
   (1.4) & \quad \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \\
   (1.5) & \quad \frac{d^2 y}{dx^2} - 2(\mu + 1) \frac{dy}{dx} = 0 \\
   (1.6) & \quad \frac{d^2}{dx^2} \left[ \rho(x) \frac{d^2 u}{dx^2} \right] = \sin(x)
   \end{align*}
   \]

Solution.

(1.1) ODE, first order, linear, inhomogeneous, variable coefficient.

The \(3x \cdot \ln x\) term on the right, which does not involve \(y\), makes the equation inhomogeneous.

The \(-2/x\) factor in front of the \(y\) is the variable coefficient.

With integrating factor \(\mu(x) = e^{-\int 2/x dx} = e^{-2 \ln x} = e^{\ln(\frac{1}{x})} = \frac{1}{x}\), the general solution is

\[
y(x) = \left( \int \frac{1}{x^2} \cdot 3x \cdot \ln x \, dx + c \right) \cdot x^2 = \frac{3x^2 \cdot (\ln x)^2}{2} + c \cdot x^2
\]

for some constant \(c\).
(1.2) PDE, first order, linear, homogeneous, variable coefficient.
The \((3t^2 - 5)\) factor in front of the \(\frac{\partial v}{\partial x}\) is the variable coefficient.

(1.3) PDE, second order, nonlinear.
Using the product rule for partial derivatives, we can write this equation in the equivalent form
\[
\frac{\partial u}{\partial t} - 2u \left( \frac{\partial u}{\partial x} \right)^2 - u^2 \left( \frac{\partial^2 u}{\partial x^2} \right) = 0.
\]
The second and third terms on the left hand side make this equation nonlinear.

(1.4) PDE, third order, nonlinear.
The \(u(\frac{\partial u}{\partial x})\) term makes this equation nonlinear. This is a version of the famous Korteweg-de Vries (KdV) equation that describes shallow water waves.

(1.5) ODE, second order, linear, homogeneous, constant coefficients.
The characteristic polynomial can be written as
\[
r^2 - 2(\mu + 1)r = r(r - 2(\mu + 1)),
\]
and a general solution is \(y(x) = c_1 + c_2 e^{2(\mu + 1)t}\) for constants \(c_1\) and \(c_2\).

(1.6) ODE, fourth order, linear, inhomogeneous, variable coefficient.
Using the product rule for partial derivatives, we can write this equation in the equivalent form
\[
\frac{d^2 \rho}{dx^2} \frac{d^2 u}{dx^2} + 2 \frac{d \rho}{dx} \frac{d^3 u}{dx^3} + \rho(x) \frac{d^4 u}{dx^4} = \sin(x),
\]

hence we can see that it is fourth order. This equation, attributed to Euler, describes the deflection of a one-dimensional beam with a static load of \(\sin(x)\); \(\rho(x)\) describes the elasticity of the material that constitutes the beam.

---

2. [21 pts (7 ea)]
Consider the temperature function
\[
u(x, t) = e^{-\kappa \theta^2 t/(\rho c)} \sin(\theta x)
\]
for constant \(\kappa\), \(\rho\), \(c\), and \(\theta\).

(a) Show that this function \(u(x, t)\) is a solution of the homogeneous heat equation
\[
\rho c \frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2},
\]
for \(0 < x < \ell\) and all \(t\).

(b) For which values of \(\theta\) will \(u\) satisfy homogeneous Dirichlet boundary conditions at \(x = 0\) and \(x = \ell\)?

(c) Suppose \(\kappa = 2.37\) W/(cm K), \(\rho = 2.70\) g/cm\(^3\), and \(c = 0.897\) J/(g K) (approximate values for aluminum found on Wikipedia), and that the bar has length \(\ell = 10\) cm. Let \(\theta\) be such that \(u(x, t)\) satisfies homogeneous Dirichlet boundary conditions as in part (b) and \(u(x, t) \geq 0\) for all \(x\) and \(t\).

Use MATLAB to plot the solution \(u(x, t)\) for \(0 \leq x \leq \ell\) and time \(0 \leq t \leq 20\) sec.

You may choose to do this in one of the following ways: (1) Plot the solution for \(0 \leq x \leq \ell\) at times \(t = 0, 4, 8, \ldots, 20\) sec., superimposing all six plots on the same axis (helpful commands: \texttt{linspace, plot, hold on}); (2) Create a three-dimensional plot of the data using \texttt{surf, mesh, or waterfall}. In either case, be sure to produce an attractive, well-labeled plot.

Solution.
(a) We compute
\[ \frac{\partial u}{\partial t} = -\kappa \theta^2 / (\rho c) e^{-\kappa \theta^2 t / (\rho c)} \sin(\theta x) \]
\[ \frac{\partial u}{\partial x} = \theta e^{-\kappa \theta^2 t / (\rho c)} \cos(\theta x) \]
\[ \frac{\partial^2 u}{\partial x^2} = -\theta^2 e^{-\kappa \theta^2 t / (\rho c)} \sin(\theta x). \]

With these formulas in hand it is easy to verify that
\[ \rho c \frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}. \]

(b) We wish to find the values of \( \theta \) that give homogeneous Dirichlet boundary conditions, i.e., \( u(0, t) = u(\ell, t) = 0 \) for all \( t \). Since \( e^{-\kappa \theta^2 t / (\rho c)} \) is positive for all \( t \), we can only get the homogeneous Dirichlet conditions when \( \sin(\theta x) = 0 \). For any \( \theta \), \( \sin(\theta \cdot 0) = 0 \), so the condition at \( x = 0 \) is automatically satisfied. To get \( \sin(\theta \ell) = 0 \), we need \( \theta \ell \) to be an integer multiple of \( \pi \), that is,
\[ \theta \ell = \pi n, \quad n = 0, \pm 1, \pm 2, \ldots, \]
or equivalently
\[ \theta = \frac{\pi n}{\ell}, \quad n = 0, \pm 1, \pm 2, \ldots. \]

(Notice that if \( n = 0 \) we have the trivial solution \( u(x, t) = 0 \) for all \( x, t \). If \( n = 1 \), we have a solution for which \( u(x, t) \geq 0 \) for all \( x, t \). For other values of \( n \) the solution will be negative for some \( x, t \). If our temperature is measured in Kelvin this could be a problem! However, this heat equation takes the same form if we shift to Celsius units, so we needn’t be so troubled by the negative values of temperature.)

Since \( n = 0 \) is trivial, we shall take \( n = 1 \) (\( \theta = \pi / \ell \)) to obtain
\[ u(x, t) = e^{-\kappa \pi^2 t / (\ell^2 \rho c)} \sin(\pi x / \ell) \]
\[ = e^{-2.37 \pi^2 t / (100 \cdot 2.70 - 0.897)} \sin(\pi x / 10). \]

(c) Solutions are shown in the attached plots. Any of these style is acceptable. The MATLAB code that generated these plots follows.

[GRADERS: please make a note if students did not include their MATLAB code, but do not take off points for this first time. You should do so in the future, though!]
MATLAB code:

c = .897;
kappa = 2.37;
rho = 2.70;
l = 10;
theta = pi/l;

% first style: snapshots at t = 0, 4, 8, ..., 20

t = 0:4:20;
x = linspace(0,l,100);

figure(1), clf
for j=1:length(t)
    u = exp(-kappa*theta^2*t(j)/(rho*c))*sin(theta*x); % compute u(:,t(j))
    plot(x,u,'k-','linewidth',2), hold on
    text(4.75, max(u)-.03, sprintf('t = %d', t(j)))
end
axis([0 10 0 1.1])
set(gca,'fontsize',14)
xlabel('x')
ylabel('u(x,t)')
print -depsc2 checksol1

% generate data for 3-d plots

x = linspace(0,l,100);
t = linspace(0,20,50);
U = zeros(length(t), length(x));
for j=1:length(t)
    U(j,:) = exp(-kappa*theta^2*t(j)/(rho*c))*sin(theta*x);
end

% mesh plot
figure(2), clf
mesh(x,t,U,'edgecolor','k')
view(-55,20)
set(gca,'fontsize',14)
xlabel('x'), ylabel('t'), zlabel('u(x,t)')
print -depsc2 checksol2

% surf plot
figure(3), clf
surf(x,t,U)
Recall the 1D steady-state heat equation with constant diffusivity over the interval $[0, 1]$

$$-\frac{\partial^2 u}{\partial x^2} = f$$

$$u(0) = u(1) = 0.$$ 

Recall from class the finite difference approximation to this problem: given a set of points $x_0, \ldots, x_{N+1}$, solved for the solution $u(x_i)$ at each point by approximating $\frac{\partial^2 u}{\partial x^2}$ with

$$u''(x_i) \approx \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2}, \quad i = 1, \ldots, N$$

(where $h$ is the spacing between points $x_{i+1}$ and $x_i$) along with the conditions that

$$u(x_0) = u(x_{N+1}) = 0.$$ 

We will modify this finite difference approximation to accomodate instead the Neumann boundary condition of $u'(1) = 0$ at $x = 1$.

(a) We would like to enforce that $u'(x_{N+1}) = 0$, but if we approximate $u'(x_{N+1})$ with a central difference

$$u'(x_{N+1}) \approx \frac{u(x_{N+\frac{1}{2}}) - u(x_{N-\frac{1}{2}})}{h},$$

we end up with an equation involving $u(x_{N+\frac{1}{2}})$, which does not lie inside the interval $[0, 1]$. Instead, we can define a backward difference approximation to the derivative

$$u'(x_{N+1}) \approx \frac{u(x_{N+1}) - u(x_N)}{h} = 0$$

and set this to zero instead. Write out the expression for $u''(x)$ in terms of $u(x_i)$ and use the backward difference approximation for $u'(x_{N+1})$ to eliminate $u(x_{N+1})$.

(b) Determine the exact solution to $-u''(x) = 1$ for $u(0) = 0, u'(1) = 0$ (hint: the solution is a quadratic function).
(c) Create a MATLAB script that constructs the matrix system $Au = f$ resulting from the finite difference equations when $f = 1$. Plot the computed solution values $u(x_i)$ for $i = 0, \ldots, N+1$ for $N = 16, 32, 64, 128$. On a separate plot, compute the maximum error $e_h$ for a given $h$

$$e_h = \max_{0 \leq i \leq N+1} |u(x_i) - u_i|$$

and plot $\log(h)$ against $\log(e_h)$.

Solution.

(a) We can rewrite the finite difference approximation to $-u''(x) = f(x)$ at point $x_N$ as

$$-\frac{u(x_{N-1}) - 2u(x_N) + u(x_{N+1})}{h^2} = f(x_N).$$

Note that

$$\frac{u(x_{N-1}) - 2u(x_N) + u(x_{N+1})}{h^2} = \frac{u(x_{N-1}) - u(x_N)}{h^2} + \frac{u(x_{N+1}) - u(x_N)}{h^2}$$

Using the backwards difference approximation to $u'(x_{N+1})$, we have

$$\frac{u(x_{N+1}) - u(x_N)}{h^2} = 0$$

which simplifies our finite difference equation at $x_N$ to

$$-\frac{u(x_{N-1}) - u(x_N)}{h^2} = f(x_N).$$

(Note that the boundary condition also implies $x_N = x_{N+1}$).

(b) We can integrate the differential equation twice to get the boundary conditions.

$$\int_0^x -u''(s)ds = \int_0^x 1ds$$

where $s$ is a dummy variable for integration. By the fundamental theorem of calculus, this gives

$$-u'(x) + u'(0) = x.$$ 

Since we don’t know the value of $u'(0)$, we consider it an unknown constant $C_1$ that we have to determine using our boundary conditions. Repeating the process again gives

$$\int_0^x (-u'(s) + C_1)ds = \int_0^x xds$$

which results in

$$-u(x) + C_1 x + u(0) = \frac{x^2}{2}.$$ 

We could set $u(0)$ to be a constant $C_2$ to be determined by the boundary conditions as well; however, since we know $u(0) = 0$ from the boundary conditions, we can go ahead and zero it out. The end result gives

$$u(x) = -\frac{x^2}{2} + C_1 x$$

The above form of the equation and the boundary condition $u'(1) = 0$ give the condition that

$$u'(1) = -1 + C_1 = 0$$.
implying $C_1 = 1$, and

$$u(x) = -\frac{x^2}{2} + x = \frac{1}{2}x(2 - x).$$

Alternatively, since the problem specifies the solution is a quadratic, it is possible to simply specify

$$u(x) = ax^2 + bx + c$$

and use the differential equation and boundary conditions to determine the constants.

c) Since the finite difference equations must be satisfied at each point $x_i$, they lead to a series of $N$ equations with $N$ unknowns (the values of $u(x_i)$ for $i = 1, \ldots, N$). The matrix system resulting from these equations for homogeneous boundary conditions

$$u(0) = u(1) = 0$$

is

$$\frac{-1}{h} \begin{bmatrix} -2 & 1 \\ 1 & -2 & 1 \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \\ & & & 1 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix},$$

where $u_i \approx u(x_i)$. Since we have the boundary condition $u'(1) = 0$ instead, this changes our finite difference equation at point $x_N$, which corresponds to the final row of our matrix. Thus, our new matrix system for a no-flux boundary condition at $x = 1$ will be

$$\frac{-1}{h^2} \begin{bmatrix} -2 & 1 \\ 1 & -2 & 1 \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \\ & & & 1 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix}.$$
Figure 1: Finite difference solutions for various $N$.

Figure 2: Error between the exact solution and finite difference solution at points $x_i$. 

\[ A = -2 \cdot \text{diag}(\text{ones}(N,1)) + \text{diag}(\text{ones}(N-1,1),1) + \text{diag}(\text{ones}(N-1,1),-1); \]
\[ A(N,N-1:N) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \quad \text{% modify last row of matrix for no-flux boundary condition} \]
\[ A = -A/h^2; \]
\[ b = \text{ones}(N,1); \quad \% f(x) = 1 \]
\[ u = A \backslash b; \]

```matlab
figure(1)
x = [0; x; 1];
plot(x, [0; u; u(N)], '.-', 'color', C(i,:), 'linewidth', 3);
hold on \% append value at x(N+1) = x(N)

err(i) = max(abs(uexact(x) - [0; u; u(N)]));

i = i+1;
end
default('Finite difference solutions compared to the exact solution', 'fontsize', 14)
plot(x, uexact(x), 'k-')
xlabel('x'); ylabel('u(x)')
legend('N = 16', 'N = 32', 'N = 64', 'N = 128', 'Exact')
print(gcf, '-dpng', 'p3c_sol') \% print out graphs to file
```

```matlab
figure(2)
default('Error between finite difference and exact solutions', 'fontsize', 14)
plot(log(Nlist), log(err), 'o-'); hold on
xlabel('log(N)')
ylabel('log(error)')
print(gcf, '-dpng', 'p3c_error') \% print out graphs to file
```

4. [14 pts (7 ea)]

**This problem is pledged!** You may not discuss this with anyone but your instructor. You may not consult any source other than approved textbooks, CAAM 336 lecture notes or your in-class notes to help you with the problem.

Consider the finite difference method applied to a simple, steady state, heat equation with mixed boundary conditions. The equation is given by

\[ -\partial_{xx} u(x) + 4\partial_x u(x) = f(x), \quad 0 < x < 1 \]
\[ u(0) = 1 \]
\[ u'(1) = 0. \]

We know that the second derivative can be approximated on uniform mesh \( \{x_0, x_1, \ldots, x_{N+1}\} \), to second order accuracy, using the central difference

\[ [\partial_{xx} u](x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} \]

(a) Write the approximation of the first derivative using the central difference. Use Taylor Series arguments to show that this approximation is a second order approximation of the first derivative at (internal) mesh point \( x_i \).

(b) Write a matrix problem \( AU = F \) which represents the finite difference approximation of equation (1) using the mixed boundary conditions. NOTE: Do not forget to include the discretization for the vectors \( F \) and \( U \).
Solution.

(a) Using a uniform mesh, of size $h$, and Taylor expansions about the point $x_i$ we have

$$
u(x_{i+1}) = u(x_i) + hu^{(1)}(x_i) + \frac{h^2}{2!}u^{(2)}(x_i) + \frac{h^3}{3!}u^{(3)}(x_i) + \ldots$$

$$u(x_{i-1}) = u(x_i) - hu^{(1)}(x_i) + \frac{h^2}{2!}u^{(2)}(x_i) - \frac{h^3}{3!}u^{(3)}(x_i) + \ldots$$

Subtracting these expansions and solving the obtained equation for $u^{(1)}(x_i)$ gives

$$u^{(1)}(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{2h} + O(h^2)$$

(b) To determine the matrix $A$ we need to look at the system of equations which arise from the given discretization techniques. Assuming we have a mesh $x_0 = 0$, $x_1, x_2, \ldots, x_N$, $x_{N+1} = 1$ we first consider the equations at an interior mesh point $x_i$, $1 < i < N$. We have:

$$-\partial_{xx} u_i + 4\partial_x u_i = -h^{-2}(u_{i+1} - 2u_i + u_{i-1}) + 2h^{-1}(u_{i+1} - u_{i-1})$$

Using the Dirichlet boundary condition $u(0) = 1$ we have a modified equation at $x_1$ given by (take $i = 1$ and use $u_0 = 0$)

$$-\partial_{xx} u_1 + 4\partial_x u_1 = -h^{-2}(u_2 - 2u_1 + 1) + 2h^{-1}(u_2 - 1)$$

Approach 1. Using backward difference approximation, the Neumann boundary condition can be written as (take $i = N$ and $u_{N+1} = u_N$)

$$-\partial_{xx} u_N + 4\partial_x u_N = h^{-2}((1 + 2h)u_N - (1 + 2h)u_{N-1})$$

and the matrix problem can be written in the following form

$$
\begin{bmatrix}
2h^{-2} & -h^{-2} + 2h^{-1} & 0 & 0 & 0 & \ldots & 0 \\
-h^{-2} + 2h^{-1} & -h^{-2} + 2h^{-1} & 0 & 0 & \ldots & 0 \\
0 & -h^{-2} + 2h^{-1} & -h^{-2} + 2h^{-1} & 0 & \ldots & 0 \\
0 & 0 & -h^{-2} + 2h^{-1} & -h^{-2} + 2h^{-1} & \ldots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \ldots & 0 & -h^{-2} - 2h^{-1} & 2h^{-2} & -h^{-2} + 2h^{-1} \\
0 & 0 & \ldots & 0 & 0 & -h^{-2}(1 + 2h) & h^{-2}(1 + 2h)
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5 \\
\vdots \\
u_N
\end{bmatrix} =
\begin{bmatrix}
f_1 + 2h^{-1} + h^{-2} \\
f_2 \\
f_3 \\
f_4 \\
\vdots \\
f_N
\end{bmatrix}
$$

Approach 2. Using central difference approximation, the Neumann boundary condition $u'(1) = 0$ yields a modified equation at $x_{N+1}$ given by (take $i = N + 1$ and $\partial_x u_{N+1} = \frac{u(x_{N+2}) - u(x_{N+1})}{2h} = 0$ to conclude that $u_{N+2} = u_N$)

$$-\partial_{xx} u_{N+1} + 4\partial_x u_{N+1} = 2h^{-2}(u_{N+1} - u_N)$$
and the matrix problem can be written in the following form

$$
\begin{bmatrix}
2h^{-2} & -h^{-2} + 2h^{-1} & 0 & 0 & 0 & \ldots & 0 \\
-h^{-2} - 2h^{-1} & 2h^{-2} & -h^{-2} + 2h^{-1} & 0 & 0 & \ldots & 0 \\
0 & -h^{-2} - 2h^{-1} & 2h^{-2} & -h^{-2} + 2h^{-1} & 0 & \ldots & 0 \\
0 & 0 & -h^{-2} - 2h^{-1} & 2h^{-2} & -h^{-2} + 2h^{-1} & \ldots & 0 \\
0 & 0 & \ldots & 0 & -h^{-2} - 2h^{-1} & 2h^{-2} & -h^{-2} + 2h^{-1} \\
0 & 0 & \ldots & 0 & 0 & -2h^{-2} & 2h^{-2} \\
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_{N+1}
\end{bmatrix}
= 
\begin{bmatrix}
f_1 + 2h^{-1} + h^{-2} \\
f_2 \\
f_3 \\
f_4 \\
\vdots \\
f_{N+1}
\end{bmatrix}$$