E. PDE 3

> **Caution**
>
> This appendix is still undergoing changes, to be released on Thursday of week 9.

E.1. The 2D Wave Equation

Exercise E.1. Now consider the 2D wave equation

$$u_{tt} = c^2 \left(u_{xx} + u_{yy} \right).$$

We want to build a numerical solution to this new PDE. Just like with the 2D heat equation we propose the notation $U_{i,j}^n$ for the approximation of the function $u(t, x, y)$ at the point $t = t_n$, $x = x_i$, and $y = y_j$.

1. Give discretizations of the derivatives u_{tt}, u_{xx}, and u_{yy}.
2. Substitute your discretizations into the 2D wave equation, make the simplifying assumption that $\Delta x = \Delta y$, and solve for $U_{i,j}^{n+1}$. This is the finite difference scheme for the 2D wave equation.
3. Write code to implement the finite difference scheme from part 2 on the domain $(x, y) \in (0, 1) \times (0, 1)$ with homogeneous Dirichlet boundary conditions, initial condition $u(0, x, y) = \sin(2\pi(x - 0.5)) \sin(2\pi(y - 0.5))$, and zero initial velocity.
4. Draw the finite difference stencil for the 2D heat equation.

Exercise E.2. What is the region of stability for the finite difference scheme on the 2D wave equation? Produce several plots showing what happens when we are in the unstable region as well as when we are right on the edge of the stable region.

Exercise E.3. Solve the 2D wave equation on the unit square with u starting at rest and being driven by a wave coming in from one boundary.

E.2. The Travelling Wave Equation

Now we turn our attention to a new PDE: the transport equation

$$u_t + vu_x = 0.$$

In this equation $u(t, x)$ is the height of a wave at time t and spatial location x. The parameter v is the velocity of the wave. Imagine this as sending a single solitary wave pulsing down a taught rope or as sending a single pulse of light down a fibre optic cable.

Exercise E.4. Consider the PDE $u_t + vu_x = 0$. There is a very easy way to get an analytic solution to this equation that describes a travelling wave. If we have the initial condition $u(0, x) = f(x) = e^{-(x-4)^2}$ then we claim that $u(t, x) = f(x - vt)$ is an analytic solution to the PDE. More explicitly, we are claiming that

$$u(t, x) = e^{-(x-vt-4)^2}$$

is the analytic solution to the PDE. Let us prove this.

1. Take the t derivative of $u(t, x)$.
2. Take the x derivative of $u(t, x)$.
3. The PDE claims that $u_t + vu_x = 0$. Verify that this equal sign is indeed true.

Exercise E.5. Now we would like to visualize the solution to the PDE from the previous exercise. The Python code below gives an interactive visual of the solution. Experiment with different values of v and different initial conditions.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation, rc
from IPython.display import HTML

v = 1
f = lambda x: np.exp(-(x-4)**2)
u = lambda t, x: f(x - v*t)
x = np.linspace(0,10,101)
t = np.linspace(0,10,101)
```

```
fig, ax = plt.subplots()
plt.close()
ax.grid()
ax.set_xlabel('x')
ax.set_xlim(( 0, 10))
ax.set_ylim(( -0.1, 1))
frame, = ax.plot([], [], linewidth=2, linestyle='--')

def animator(N):
  ax.set_title(f"Time = {t[N]:.2f}")
  frame.set_data(x,???) # plot the correct time step for u(t,x)
  return (frame,)

PlotFrames = range(0,len(t),1)
anim = animation.FuncAnimation(fig,
                               animator,
                               frames=PlotFrames,
                               interval=100,
                              )

rc('animation', html='jshtml') # embed in the HTML for Google Colab
anim
```

Exercise E.6. Use the chain rule to prove that for any differentiable function $f(x)$ the function $u(t,x) = f(x-vt)$ is an analytic solution to the transport equation $u_t + vu_x = 0$ with initial condition $u(0,x) = f(x)$.

Thus the travelling wave equation $u_t + vu_x = 0$ has a very nice analytic solution which we can always find. Therefore there is no need to ever find a numerical solution – we can just write down the analytic solution if we are given the initial condition. As it turns out though, the numerical solutions exhibit some very interesting behaviour.

Exercise E.7. Consider the travelling wave equation $u_t + vu_x = 0$ with initial condition $u(0,x) = f(x)$ for some given function f and boundary condition $u(t,0) = 0$. To build a numerical solution we will again adopt the notation U_i^n for the approximation to $u(t,x)$ at the point $t = t_n$ and $x = x_i$.

(a) Write an approximation of u_t using U_i^{n+1} and U_i^n.

(b) Write an approximation of u_x using U_{i+1}^n and U_i^n.

(c) Substitute your answers from parts (a) and (b) into the travelling wave equation and solve for U_i^{n+1}. This is our first finite difference scheme for the travelling wave equation.

(d) Write Python code to get the finite difference approximation of the solution to the PDE. Plot your finite difference solution on top of the analytic solution for $f(x) = e^{-(x-4)^2}$. What do you notice? Can you stabilize this method by changing the values of Δt and Δx like with did with the heat and wave equations?

The finite difference scheme that you built in the previous exercise is called the downwind scheme for the travelling wave equation. Figure E.1 shows the finite difference stencil for this scheme. We call this scheme "downwind" since we expect the wave to travel from left to right and we can think of a fictitious wind blowing the solution from left to right. Notice that we are using information from "downwind" of the point at the new time step.

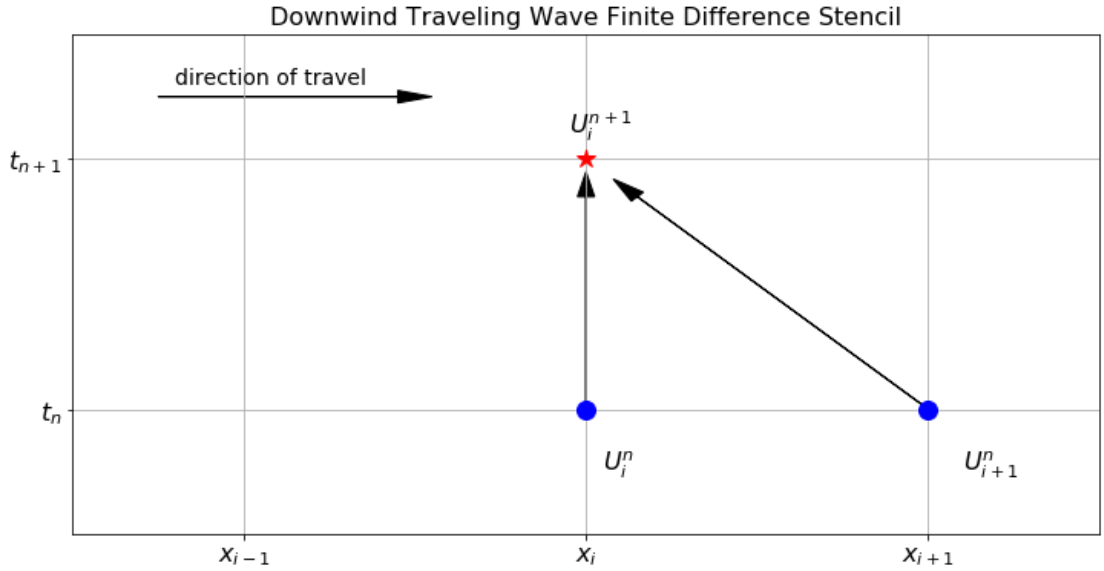

Figure E.1.: The finite difference stencil for the 1D downwind scheme on the traveling wave equation.

Exercise E.8. You should have noticed in the previous exercise that you cannot reasonably stabilize the finite difference scheme. Propose several reasons why this method appears to be unstable no matter what you use for the ratio $v\Delta t/\Delta x$.

Exercise E.9. One of the troubles with the finite difference scheme that we have built for the travelling wave equation is that we are using the information at our present spatial location and the next spatial location to the right to propagate the solution forward in time. The trouble here is that the wave is moving from left to right, so the interesting information about the next time step's solution is actually coming from the left, not the right. We call this "looking upwind" since you can think of a fictitious *wind* blowing from left to right, and we need to look "upwind" to see what is coming at us. If we write the spatial derivative as

$$u_x \approx \frac{U_i^n - U_{i-1}^n}{\Delta x}$$

we still have a first-order approximation of the derivative but we are now looking left instead of right for our spatial derivative. Make this modification in your finite difference code for the travelling wave equation (call it the "upwind method"). Approximate the solution to the same PDE as we worked with in the previous exercises. What do you notice now?

Figure E.2 shows the finite difference stencil for the upwind scheme. We call this scheme "up" since we expect the wave to travel from left to right and we can think of a fictitious wind blowing the solution from left to right. Notice that we are using information from "upwind" of the point at the new time step.

Exercise E.10. Complete the following sentences:

1. In the downwind finite difference scheme for the travelling wave equation, the approximate solution moves at the correct speed, but …
2. In the upwind finite difference scheme for the travelling wave equation, the approximate solution moves at the correct speed, but …

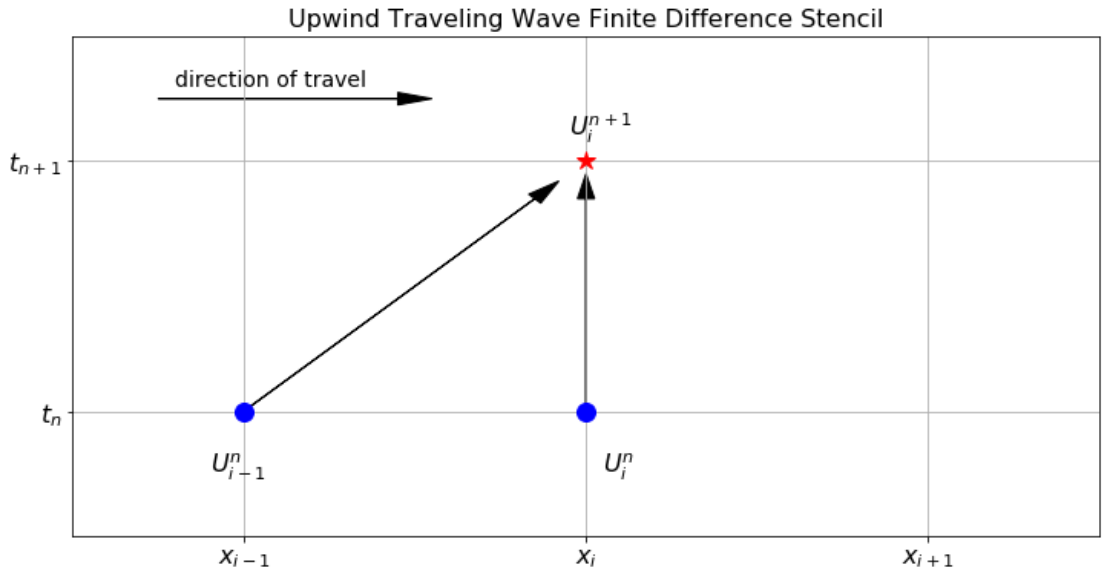

Figure E.2.: The finite difference stencil for the 1D downwind scheme on the traveling wave equation.

Exercise E.11. Neither the downwind nor the upwind solutions for the travelling wave equation are satisfactory. They completely miss the interesting dynamics of the analytic solution to the PDE. Some ideas for stabilizing the finite difference solution for the travelling wave equation are as follows. Implement each of these ideas and discuss pros and cons of each. Also draw a finite difference stencil for each of these methods.

1. Perhaps one of the issues is that we are using first-order methods to approximate u_t and u_x. What if we used a second-order approximation for these first derivatives

$$u_t \approx \frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t} \quad \text{and} \quad u_x \approx \frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}?$$

   Solve for U_i^{n+1} and implement this method. This is called the **leapfrog method.**

2. For this next method let us stick with the second-order approximation of u_x but we will do something clever for u_t. For the time derivative we originally used

$$u_t \approx \frac{U_i^{n+1} - U_i^n}{\Delta t}$$

   what happens if we replace U_i^n with the average value from the two surrounding spatial points

$$U_i^n \approx \frac{1}{2}\left(U_{i+1}^n + U_{i-1}^n\right)?$$

   This would make our approximation of the time derivative

$$u_t \approx \frac{U_i^{n+1} - \frac{1}{2}\left(U_{i+1}^n + U_{i-1}^n\right)}{\Delta t}.$$

Solve this modified finite difference equation for U_i^{n+1} and implement this method. This is called the **Lax-Friedrichs** method.

3. Finally we will do something very clever (and very counter intuitive). What if we inserted some artificial diffusion into the problem? You know from your work with the heat equation that diffusion spreads a solution out. The downwind scheme seemed to have the issue that it was *bunching up* at the beginning and end of the wave, so artificial diffusion might smooth this out. The **Lax-Wendroff method** does exactly that: take a regular Euler-type step in time

$$u_t \approx \frac{U_i^{n+1} - U_i^n}{\Delta t},$$

use a second-order centred difference scheme in space to approximate u_x

$$u_x \approx \frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x},$$

but add on the term

$$\left(\frac{v^2 \Delta t^2}{2\Delta x^2}\right)\left(U_{i-1}^n - 2U_i^n + U_{i+1}^n\right)$$

to the right-hand side of the equation. Notice that this new term is a scalar multiple of the second-order approximation of the second derivative u_{xx}. Solve this equation for U_i^{n+1} and implement the Lax-Wendroff method.

E.3. The Laplace and Poisson Equations

Exercise E.12. Consider the 1D heat equation $u_t = 1u_{xx}$ with boundary conditions $u(t, 0) = 0$ and $u(t, 1) = 1$ and initial condition $u(0, x) = 0$.

1. Describe the physical setup for this problem.
2. Recall that the solution to a differential equation reaches a steady state (or equilibrium) when the time rate of change is zero. Based on the physical system, what is the steady state heat profile for this PDE?
3. Use your 1D heat equation code to show the full time evolution of this PDE. Run the simulation long enough so that you see the steady state heat profile.

Exercise E.13. Now consider the forced 1D heat equation $u_t = u_{xx} + e^{-(x-0.5)^2}$ with the same boundary and initial conditions as the previous exercise. The exponential forcing function introduced in this equation is an external source of heat (like a flame held to the middle of the metal rod).

1. Conjecture what the steady state heat profile will look like for this particular setup. Be able to defend your answer.
2. Modify your 1D heat equation code to show the full time evolution of this PDE. Run the simulation long enough so that you see the steady state heat profile.

Exercise E.14. Next we will examine 2D steady state heat profiles. Consider the PDE $u_t = u_{xx} + u_{yy}$ with boundary conditions $u(t, 0, y) = u(t, x, 0) = u(t, x, 1) = 0$ and $u(t, 1, y) = 1$ with initial condition $u(0, x, y) = 0$.

1. Describe the physical setup for this problem.
2. Based on the physical system, describe the steady state heat profile for this PDE. Be sure that your steady state solution still satisfies the boundary conditions.
3. Use your 2D heat equation code to show the full time evolution of this PDE. Run the simulation long enough so that you see the steady state heat profile.

Exercise E.15. Now consider the forced 2D heat equation $u_t = u_{xx} + u_{yy} + 10e^{-(x-0.5)^2-(y-0.5)^2}$ with the same boundary and initial conditions as the previous exercise. The exponential forcing function introduced in this equation is an external source of heat (like a flame held to the middle of the metal sheet).

1. Conjecture what the steady state heat profile will look like for this particular setup. Be able to defend your answer.
2. Modify your 2D heat equation code to show the full time evolution of this PDE. Run the simulation long enough so that you see the steady state heat profile.

Up to this point we have studied PDEs that all depend on time. In many applications, however, we are not interested in the transient (time dependent) behaviour of a system. Instead we are often interested in the steady state solution when the forces in question are in static equilibrium. Two very famous time-independent PDEs are the Laplace Equation

$$u_{xx} + u_{yy} + u_{zz} = 0$$

and the Poisson equation

$$u_{xx} + u_{yy} + u_{zz} = f(x, y, z).$$

Notice that both the Laplace and Poisson equations are the equations that we get when we consider the limit $u_t \to 0$. In the limit when the time rate of change goes to zero we are actually just looking at the eventual steady state heat profile resulting from the initial and boundary conditions of the heat equation. In the previous exercises you already wrote code that will show the steady state profiles in a few setups. The trouble with the approach of letting the time-dependent simulation run for a *long time* is that the finite difference solution for the heat equation is known to have stability issues. Moreover, it may take a lot of computational time for the solution to reach the eventual steady state. In the remainder of this section we look at methods of solving for the steady state directly – without examining any of the transient behaviour. We will first examine a 1D version of the Laplace and Poisson equations.

Exercise E.16. Consider a 1-dimensional rod that is infinitely thin and has unit length. For the sake of simplicity assume the following:

- the specific heat of the rod is exactly 1 for the entire length of the rod,
- the temperature of the left end is held fixed at $u(0) = 0$,
- the temperature of the right end is held fixed at $u(1) = 1$, and
- the temperature has reached a steady state.

You can assume that the temperatures are *reference temperatures* instead of absolute temperatures, so a temperature of "0" might represent room temperature.

Since there are no external sources of heat we model the steady-state heat profile we must have $u_t = 0$ in the heat equation. Thus the heat equation collapses to $u_{xx} = 0$. This is exactly the one dimensional Laplace equation.

(a) To get an exact solution of the Laplace equation in this situation we simply need to integrate twice. Do the integration and write the analytic solution (there should be no surprises here).

(b) To get a numerical solution we first need to partition the domain into finitely many point. For the sake of simplicity let us say that we subdivide the interval into 5 equal sub intervals (so there are 6 points including the endpoints). Furthermore, we know that we can approximate u_{xx} as

$$u_{xx} \approx \frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}.$$

Thus we have 6 linear equations:

$$\begin{aligned}
U_0 &= 1 \quad \text{(left boundary condition)} \\
\frac{U_2 - 2U_1 + U_0}{\Delta x^2} &= 0 \\
\frac{U_3 - 2U_2 + U_1}{\Delta x^2} &= 0 \\
\frac{U_4 - 2U_3 + U_2}{\Delta x^2} &= 0 \\
\frac{U_5 - 2U_4 + U_3}{\Delta x^2} &= 0 \\
U_5 &= 0 \quad \text{(right boundary condition).}
\end{aligned}$$

Notice that there are really only four unknowns since the boundary conditions dictate two of the temperature values. Rearrange this system of equations into a matrix equation and solve for the unknowns U_1, U_2, U_3, and U_4. Your coefficient matrix should be 4×4.

(c) Compare your answers from parts (a) and (b).

(d) Write code to build the numerical solution with an arbitrary value for Δx (i.e. an arbitrary number of sub intervals). You should build the linear system automatically in your code.

Solving the 1D Laplace equation with Dirichlet boundary conditions is rather uninteresting since the answer will always be a linear function connecting the two boundary conditions. The Poisson equation $u_{xx} = f(x)$ is more interesting than the Laplace equation in 1D. The function $f(x)$ is called a forcing function. You can think of it this way: if u is the amount of force on a linear bridge, then f might be a function that gives the distribution of the forces on the bridge due to the cars sitting on the bridge. In terms of heat we can think of this as an external source of heat energy warming up the one-dimensional rod somewhere in the middle (like a flame being held to one place on the rod).

Exercise E.17. How would we analytically solve the Poisson equation $u_{xx} = f(x)$ in one spatial dimension? As a sample problem consider $x \in [0, 1]$, the forcing function $f(x) = 5\sin(2\pi x)$ and boundary conditions $u(0) = 2$ and $u(1) = 0.5$. Of course you need to check your answer by taking two derivatives and making sure that the second derivative exactly matches $f(x)$. Also be sure that your solution matches the boundary conditions exactly.

Exercise E.18. Now we can solve the Poisson equation from the previous problem numerically. Let us again build this with a partition that contains only 6 points just like we did with the Laplace equation a few exercise ago. We know the approximation for u_{xx} so we have the linear system

$$\begin{aligned} U_0 &= 2 \quad \text{(left boundary condition)} \\ \frac{U_2 - 2U_1 + U_0}{\Delta x^2} &= f(x_1) \\ \frac{U_3 - 2U_2 + U_1}{\Delta x^2} &= f(x_2) \\ \frac{U_4 - 2U_3 + U_2}{\Delta x^2} &= f(x_3) \\ \frac{U_5 - 2U_4 + U_3}{\Delta x^2} &= f(x_4) \\ U_5 &= 0.5 \quad \text{(right boundary condition)}. \end{aligned}$$

(a) Rearrange the system of equations as a matrix equation and then solve the system for U_1, U_2, U_3, and U_4. There are really only four equations so your matrix should be 4×4.

(b) Compare your solution from part (a) to the function values that you found in the previous exercise.

(c) Now generalize the process of solving the 1D Poisson equation for an arbitrary value of Δx. You will need to build the matrix and the right-hand side in your code. Test your code on new forcing functions and new boundary conditions.

Exercise E.19. The previous exercises only account for Dirichlet boundary conditions (fixed boundary conditions). We would now like to modify our Poisson solution to allow for a Neumann condition: where we know the derivative of u at one of the boundaries. The statement of the problem is as follows:

$$\text{Solve: } u_{xx} = f(x) \quad \text{on} \quad x \in (0,1) \quad \text{with} \quad u_x(0) = \alpha \quad \text{and} \quad u(1) = \beta.$$

The derivative condition on the boundary can be approximated by using a first-order approximation of the derivative, and as a consequence we have one new equation. Specifically, if we know that $u_x(0) = \alpha$ then we can approximate this condition as

$$\frac{U_1 - U_0}{\Delta x} = \alpha,$$

and we simply need to add this equation to the system that we were solving in the previous exercise. If we go back to our example of a partition with 6 points the system becomes

$$\begin{aligned}
\frac{U_1 - U_0}{\Delta x} &= \alpha \quad \text{(left boundary condition)} \\
\frac{U_2 - 2U_1 + U_0}{\Delta x^2} &= f(x_1) \\
\frac{U_3 - 2U_2 + U_1}{\Delta x^2} &= f(x_2) \\
\frac{U_4 - 2U_3 + U_2}{\Delta x^2} &= f(x_3) \\
\frac{U_5 - 2U_4 + U_3}{\Delta x^2} &= f(x_4) \\
U_5 &= \beta \quad \text{(right boundary condition).}
\end{aligned}$$

There are 5 equations this time.

(a) With a 6 point grid solve the Poisson equation $u_{xx} = 5\sin(2\pi x)$ with $u_x(0) = 0$ and $u(1) = 3$.

(b) Modify your code from part (a) to solve the same problem but with a much smaller value of Δx. You will need to build the matrix equation in your code.

Exercise E.20 (The 2D Poisson Equation)**.** We conclude this section, and chapter, by examining the two dimensional Poisson equations. As a sample problem, we want to solve the Poisson equation

$$u_{xx} + u_{yy} = f(x, y)$$

on the domain $(x, y) \in (0, 1) \times (0, 1)$ with homogeneous Dirichlet boundary conditions and forcing function

$$f(x, y) = -20\exp\left(-\frac{(x - 0.5)^2 + (y - 0.5)^2}{0.05}\right)$$

numerically.

We are going to start with a 6×6 grid of points and explicitly write down all of the equations. In Figure E.3 the red stars represent boundary points where the value of $u(x, y)$ is known and the blue interior points are the ones where $u(x, y)$ is yet unknown. It should be clear that we should have two indices for each point (one for the x position and one for the y position), but it should also be clear that this will cause problems when writing down the resulting system of equations as a matrix equation (stop and think carefully about this). Therefore, in Figure E.3 we propose an index, k, starting at the top left of the unknown nodes and reading left to right (just like we do with Python arrays).

(a) Start by discretizing the 2D Poisson equation $u_{xx} + u_{yy} = f(x, y)$. For simplicity we assume that $\Delta x = \Delta y$ so that we can combine like terms from the x derivative and the y derivative. Fill in the missing coefficients and indices below.

$$U_{i+1,j} + U_{i,j-1} - (\_\_)U_{\_\_,\_\_} + U_{\_\_,\_\_} + U_{\_\_,\_\_} = \Delta x^2 f(x_i, y_i)$$

(b) In Figure E.3 we see that there are 16 total equations resulting from the discretization of the Poisson equation. Your first task is to write all 16 of these equations. we will get you started:

$$\begin{aligned}
k = 0: &\quad U_{k=1} + U_{i=1,j=0} - 4U_{k=0} + U_{i=0,j=1} + U_{k=4} = \Delta x^2 f(x_1, y_1) \\
k = 1: &\quad U_{k=2} + U_{k=0} - 4U_{k=1} + U_{i=0,j=2} + U_{k=5} = \Delta x^2 f(x_1, y_2) \\
&\quad \vdots \\
k = 15: &\quad U_{i=4,j=5} + U_{k=14} - 4U_{k=15} + U_{k=11} + U_{i=5,j=4} = \Delta x^2 f(x_4, y_4)
\end{aligned}$$

In this particular example we have homogeneous Dirichlet boundary conditions so all of the boundary values are zero. If this was not the case then every boundary value would need to be moved to the right-hand sides of the equations.

(c) We now have a 16×16 matrix equation to write based on the equations from part (b). Each row and column of the matrix equation is indexed by k. The coefficient matrix A is started for you below. Write the whole thing out and fill in the blanks. Notice that this matrix has a much more complicated structure than the coefficient matrix in the 1D Poisson and Laplace equations.

$$A = \begin{pmatrix}
-4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & & \\
1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & \ddots & \\
0 & & & & & & & & & \\
\vdots & & & & & & & & & \\
 & & & & & & & & & -4
\end{pmatrix}$$

(d) In the coefficient matrix from part (c) notice that the small matrix

$$\begin{pmatrix}
-4 & 1 & 0 & 0 \\
1 & -4 & 1 & 0 \\
0 & 1 & -4 & 1 \\
0 & 0 & 1 & -4
\end{pmatrix}$$

shows up in blocks along the main diagonal. If you have a hard copy of the matrix go back and draw a box around these blocks in the coefficient matrix. Also notice that there are diagonal bands of 1^s. Discuss the following:

1. Why are the blocks 4×4?
2. How could you have predicted the location of the diagonal bands of 1^s?
3. What would the structure of the matrix look like if we partitioned the domain into a 10×10 grid of points instead of a 6×6 grid (including the boundary points)?
4. Why is it helpful to notice this structure?

(e) The right-hand side of the matrix equation resulting the your system of equations from part (b) is

$$b = \Delta x^2 \begin{pmatrix} f(x_1, y_1) \\ f(x_1, y_2) \\ f(x_1, y_3) \\ f(x_1, y_4) \\ f(x_2, y_1) \\ f(x_2, y_2) \\ \vdots \\ \\ f(x_4, y_y) \end{pmatrix}.$$

Notice the structure of this vector. Why is it structured this way? Why is it useful to notice this?

(f) Write Python code to solve the problem at hand. Recall that $f(x, y) = -20 \exp\left(-\frac{-(x-0.5)^2+(y-0.5)^2}{0.05}\right)$. Show a contour plot of your solution. This will take a little work changing the indices back from k to i and j. Think carefully about how you want to code this before you put fingers to keyboard. You might want to use the `np.block()` command to build the coefficient matrix efficiently or you can use loops with carefully chosen indices.

(g) (Challenge) Generalize your code to solve the Poisson equation with a much smaller value of $\Delta x = \Delta y$.

(h) One more significant observation should be made about the 2D Poisson equation on this square domain. Notice that the corner points of the domain (e.g. $i = 0, j = 0$ or $i = 5, j = 0$) are never included in the system of equations. What does this mean about trying to enforce boundary conditions that only apply at the corners?

E.4. Algorithm Summaries

Exercise E.21. Show the full mathematical details for building a first-order in time and second-order in space approximation method for the one-dimensional heat equation. Explain what the order of the error means in this context

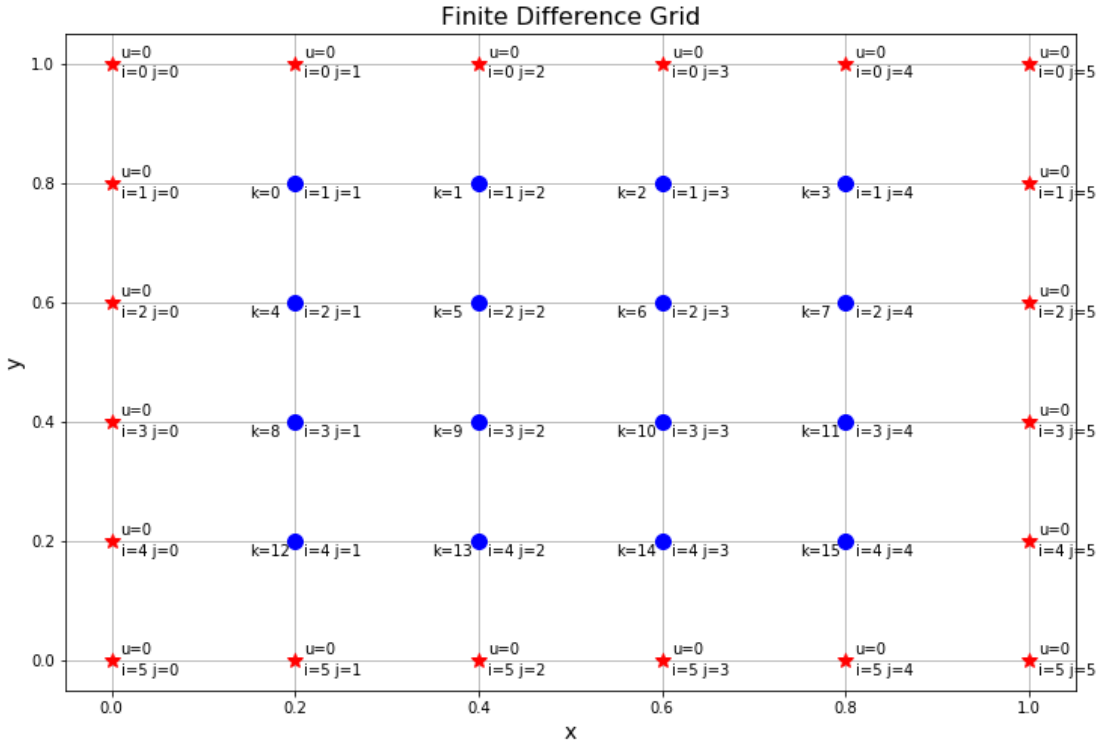

Figure E.3.: A finite difference grid for the Poisson equation with 6 grid points in each direction.

Exercise E.22. Show the full mathematical details for building a second-order in time and second-order in space approximation method for the one-dimensional wave equation. Explain what the order of the error means in this context

---

Exercise E.23. Show the full mathematical details for building a first-order in time and second-order in space approximation method for the two-dimensional heat equation. Explain what the order of the error means in this context

---

Exercise E.24. Show the full mathematical details for building a second-order in time and second-order in space approximation method for the two-dimensional wave equation. Explain what the order of the error means in this context

---

Exercise E.25. Explain in clear language what it means for a finite difference method to be stable versus unstable.

Exercise E.26. Show the full mathematical details for solving the 1D heat equation using the implicit and Crank-Nicolson methods.

Exercise E.27. Show the full mathematical details for building a downwind finite difference scheme for the travelling wave equation. Discuss the primary disadvantages of the downwind scheme.

Exercise E.28. Show the full mathematical details for building an upwind finite difference scheme for the travelling wave equation. Discuss the primary disadvantages of the upwind scheme.

Exercise E.29. Show the full mathematical details for numerically solving the 1D Laplace and Poisson equations.

Acton, Forman S. 1990. *Numerical Methods That Work*. 1St Edition edition. The Mathematical Association of America.

Burden, Richard L., and J. Douglas Faires. 2010. *Numerical Analysis*. 9th ed. Brooks Cole.

Butcher, J. C. 2016. *Numerical Methods for Ordinary Differential Equations*. Third edition. Wiley. https://yorsearch.york.ac.uk/permalink/f/1kq3a7l/44YORK_ALMA_DS51336126850001381.

Greenbaum, Anne, and Tim P. Chartier. 2012. *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*. Princeton University Press.

Kincaid, D. R., and E. W. Cheney. 2009. *Numerical Analysis: Mathematics of Scientific Computing*. Pure and Applied Undergraduate Texts. American Mathematical Society.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press. https://numerical.recipes/.