Approximating Solutions to the Heat Equation

1 Introduction

In this exercise we will consider the two-dimensional heat equation and solve it with an implicit scheme. We will show that this implicit scheme is first-order accurate in space, and second-order accurate in time. We then consider the internal temperature of a potato along a two-dimensional plane. We will approximate the potato's dimensions as a rectangle and simulate the potato's temperature as it boils in a pot of water. Finally, we will use the data from our MATLAB simulation of the potato to plot its internal temperature distribution and determine when the potato has fully cooked.

2 Part a

We consider the heat transfer in a rectangular domain $\Omega = [x_l;x_r]\times[y_b;y_t]$

$$\begin{cases} \dfrac{\partial T}{\partial t} = \lambda \Delta T + f, & \text{(x,y)}\in \Omega \\ T(t,x,y) = T_{bc}(t,x,y), & \text{(x,y)}\in \partial\Omega \\ T(t_{start},x,y) = T_{start}(x,y), & \text{(x,y)}\in \Omega \end{cases} \tag{1}$$

where λ is the thermal diffusivity of the material, f, T_{bc} and T_{start} are given functions of the source term, boundary conditions, and initial conditions, respectively. Note that ΔT refers to the Laplace operator, which is the gradient operator squared. In other words $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ We will discretize (1) using the implicit scheme as follows: Using method of lines, we choose a grid resolution N_x and discretize space as $x_1 = x_l$, $x_r = x_{N_x}$ Δ x $= \frac{x_r - x_l}{N_x - 1}$, similarly $y_b = y_1$, $y_t = y_{N_y}$ $\Delta y = \frac{y_t - y_b}{N_y - 1}$. For time we will see that we can discretize in any way we like, since the implicit scheme we use is unconditionally stable in time, so we can simply say $\Delta t = \frac{t_{final} - t_{start}}{N_t - 1}$. Now using the backwards finite difference formula for time and the central finite difference formula for space (1) becomes:

$$\begin{cases} \dfrac{\partial T}{\partial t}(t_{n+1},x_i,j_j) \approx \dfrac{T(t_{n+1},x_i,y_j) - T(t_n,x_i,y_j)}{\Delta t} \approx \dfrac{T^{n+1}_{i,j} - T^n_{i,j}}{\Delta t} \\ \dfrac{\partial^2 T}{\partial x^2}(t_{n+1},x_i,y_j) \approx \dfrac{T(t_{n+1},x_{i+1},y_j) - 2T(t_{n+1},x_i,y_j) + T(t_{n+1},x_{i-1},y_j)}{\Delta x^2} \approx \dfrac{T^{n+1}_{i+1,j} - 2T^{n+1}_{i,j} + T^{n+1}_{i-1,j}}{\Delta x^2} \\ \dfrac{\partial^2 T}{\partial y^2}(t_{n+1},x_i,y_j) \approx \dfrac{T(t_{n+1},x_i,y_{j+1}) - 2T(t_{n+1},x_i,y_j) + T(t_{n+1},x_i,y_{j-1})}{\Delta y^2} \approx \dfrac{T^{n+1}_{i,j+1} - 2T^{n+1}_{i,j} + T^{n+1}_{i,j-1}}{\Delta y^2} \end{cases} \tag{2}$$

and thus obtaining:

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} = D(\frac{T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1} + T_{i-1,j}^{n+1}}{\Delta x^2} + \frac{T_{i,j+1}^{n+1} - 2T_{i,j}^{n+1} + T_{i,j-1}^{n+1}}{\Delta y^2}) + f(t_{n+1}, x_i, y_j) \tag{3}$$

Since we are attempting to solve for T at time t = t_{n+1} using the solution for T at t_{n+1}, we cannot explicitly solve this equation. We must therefore set this up as a system of equations. There are $N_x \times N_y$ grid points to solve for, and therefore we will have $N_x N_y$ equations in our system to solve simultaneously. If we rearrange (2) such that all unknowns are on the left-hand side and all known terms on the right-hand side, we obtain the expression:

$$T_{i,j-1}^{n+1}(-\frac{D\Delta t}{\Delta y^2}) + T_{i-1,j}^{n+1}(-\frac{D\Delta t}{\Delta x^2}) + T_{i,j}^{n+1}(1 + 2\frac{D\Delta t}{\Delta x^2} + 2\frac{D\Delta t}{\Delta y^2}) + T_{i,j+1}^{n+1}(-\frac{D\Delta t}{\Delta y^2}) = T_{i,j}^{n} + \Delta t f(t_{n+1}, x_i, y_j) \tag{4}$$

Our system of equations will be much more easily solved as a matrix equation A·T_{n+1} = r. We need a column vector of grid points at t = t_{n+1}, T_{n+1}, and grid points at t = t_n, to use for the column vector r. We therefore must organize our 2D grid of points as follows:

$T_s = T_{i,j}$ where the index s = (j-1)N_x + i for all (i,j) $\in \Omega$. This will organize the column vectors as descending values of grid points starting from the bottom-left grid point and proceeding to the right along the grid points in the domain until the top-right grid point is the last value in the column vectors. This applies to both column vectors, but at different times of course. The matrix, A, and the equation of each value of the r vector, is a little more nuanced than the simple T_{n+1} vector. We know that at internal grid points (4) holds. However, we must apply boundary conditions to all points along the outside edges. For internal points, we will have:

$$\begin{aligned}
r &= T_s^n + \Delta t f(t_{n+1}, x_i, y_j) \\
A_{s,s} &= (1 + 2\frac{D\Delta t}{\Delta x^2} + 2\frac{D\Delta t}{\Delta y^2}) \\
A_{s,s-1} &= A_{s,s+1} = -\frac{D\Delta t}{\Delta x^2} \\
A_{s,s-N_x} &= A_{s,s+N_x} = -\frac{D\Delta t}{\Delta y^2} \\
A_{s,k} &= 0 \text{ for all other positions of the } s^{th} \text{ row of matrix A}
\end{aligned} \tag{5}$$

In this way, matrix A's s^{th} row values correspond appropriately with the grid points to the left, right, above, below, and center of grid point T_s, and we have described one equation of the form (4). We do this for all values of s where i and j are internal grid points (i,j) such that, $1 < i < N_x$ and $1 < j < N_y$.

For all boundary points we have:

$$\begin{aligned}
r &= T_{bc}(t_{n+1}, x_i, y_j) \\
A_{s,s} &= 1 \\
A_{s,k} &= 0 \text{ for all other positions of the } s^{th} \text{ row of matrix A}
\end{aligned} \tag{6}$$

For all rows s of the matrix equation where $i = 1$ or $j = 1$ or $i = N_x$ or $j = N_y$ or any combination of these values, corresponding to points on the boundary of the domain.

3 Part b

We now consider a heat equation with domain $[-1, 1] \times [-0.5, 1.7]$, $\lambda = 0.75$, and exact solution $T_{exact} = sin(x)cos(y)exp(-t)$. By plugging T_{exact} into the diffusion equation, we calculate T_{start}, T_{bc}, and f(t,x,y) as:

$$\begin{aligned} f(t,x,y) &= (2\lambda - 1)exp(-t_n))cos(y_j)sin(x_i) \\ T_{bc} &= T_{exact}(t_n, x_i, y_j) = sin(x_i)cos(y_j)exp(-t_n) \\ T_{start} &= T_{exact}(t_{start}, x_i, y_j) = sin(x_i)cos(y_j)exp(-t_{start}) \end{aligned} \tag{7}$$

We take $\Delta t = \Delta x/2$, and grid resolutions (N_x,N_y) = (25,30),(50,60),(100,120), with t_{start} = 0, $t_{final} = 1$, and solve with a MATLAB implementation of the implicit method as described previously. We then calculate the error of the implicit method for each grid resolution as the maximum deviation from the exact solution at t_final:

$$E_{trial} = max(|T_{exact}(t_{final}, x_i, y_j) - T_{i,j}^{t_{final}}|)$$

We then calculate the order of accuracy as:

$$\left| \frac{log\left(\frac{error(trial_m)}{error(trial_{m+1})}\right)}{log\left(\frac{N_x(trial_{m+1})}{N_x(trial_m)}\right)} \right|$$

Where $N_x(trial_m)$ is the grid resolution in the x dimension for the m^{th} trial. In this case, the denominator always corresponds to log(2).

The result we obtained indicates that the implicit scheme is first-order accurate in space:

(Nx, Ny)	Error	Order k
(25, 30)	0.000584	-
(50, 60)	0.000278	1.068439
(100, 120)	0.000136	1.031632

Table 1: Accuracy of the implicit diffusion scheme implemented in MATLAB

4 Part c

We now turn our attention to the problem of a boiling potato. The parameters of this diffusion equation are:

$$\begin{aligned} \Omega &= [-2,2] \times [-2.5, 2.5]\text{cm} \\ \lambda &= 1.5 \times 10^{-3} cm^2/\text{s} \\ T_{start}(x,y) &= 20^\circ\text{C} \\ T_{bc}(t,x,y) &= \min\left(20 + 80\frac{t}{60}, 100\right)^\circ \text{C} \\ f(t,x,y) &= 0 \end{aligned} \tag{8}$$

We will solve for $0 \leq t \leq 1500$ seconds, using grid resolution $N_x = 80$, $N_y = 100$, $\Delta t = 5$ seconds. We obtain the graph of the temperature of the center of the potato:

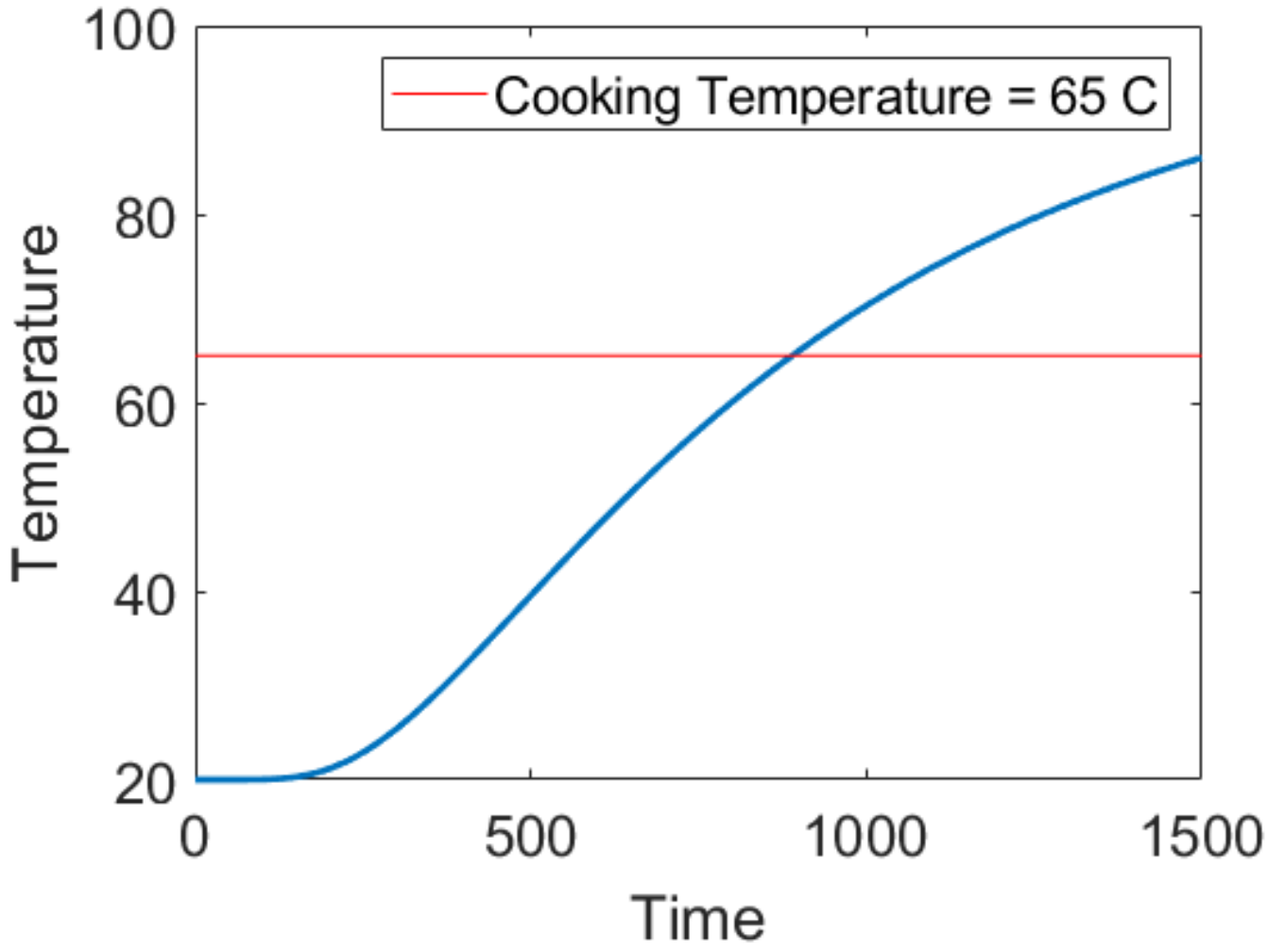

Figure 1: Temperature over time at the potato's center.

We found that the potato began to cook (reach 65 degrees celsius at its center) at approximately 890 seconds and we found that it had fully cooked 300 seconds later, at 1190 seconds. This corresponds to slightly less than 20 minutes to fully cook the potato. We also obtained the temperature distribution within the potato at time t = 0, 200, 400, 600 seconds:

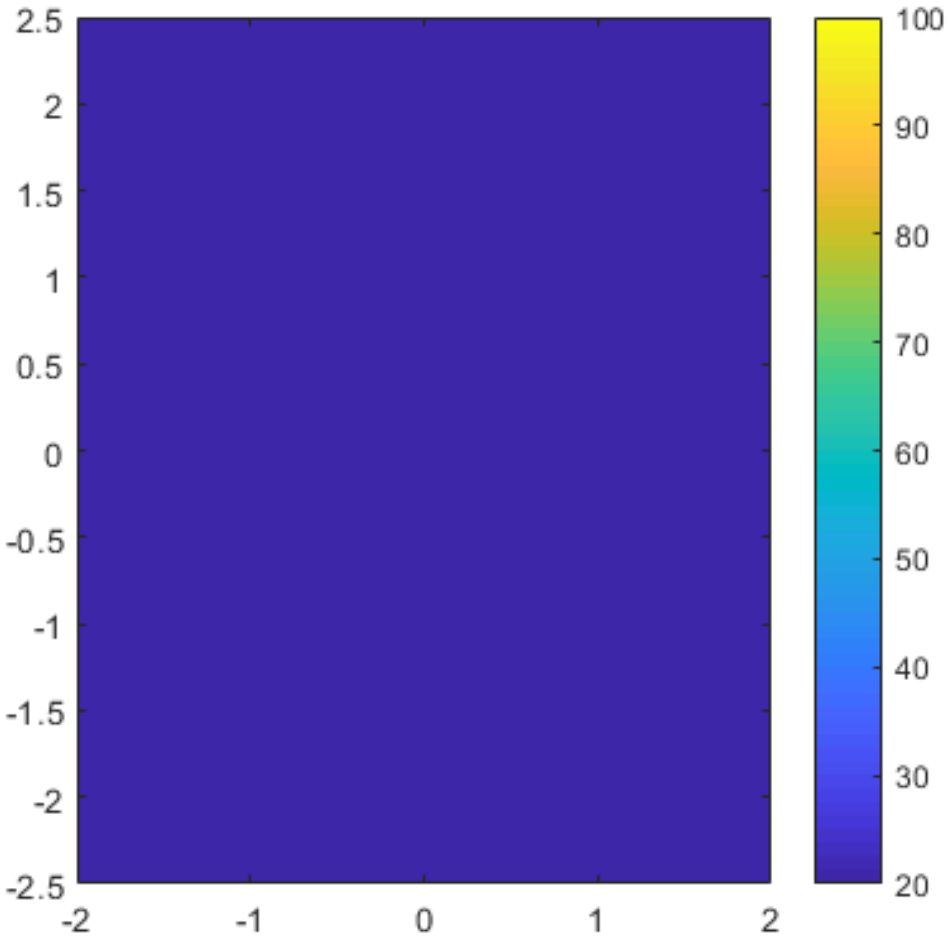

Figure 2: Temperature distribution at time t = 0 seconds.

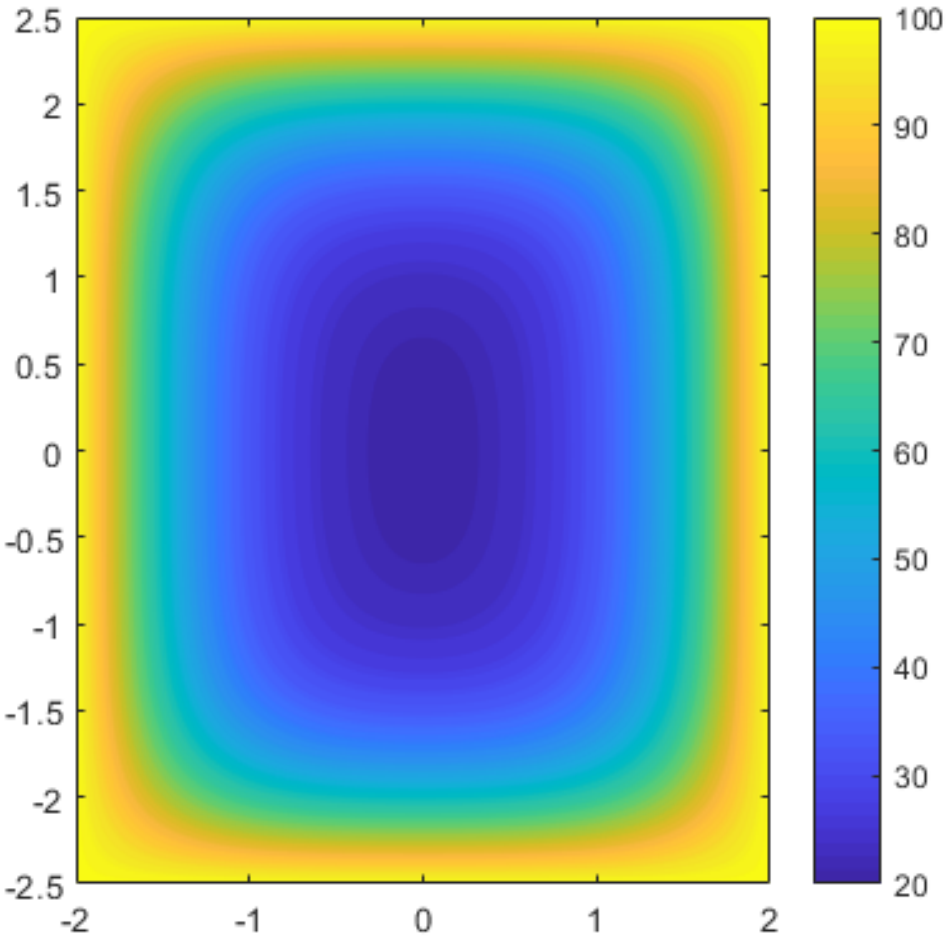

Figure 3: Temperature distribution at time t = 200 seconds.

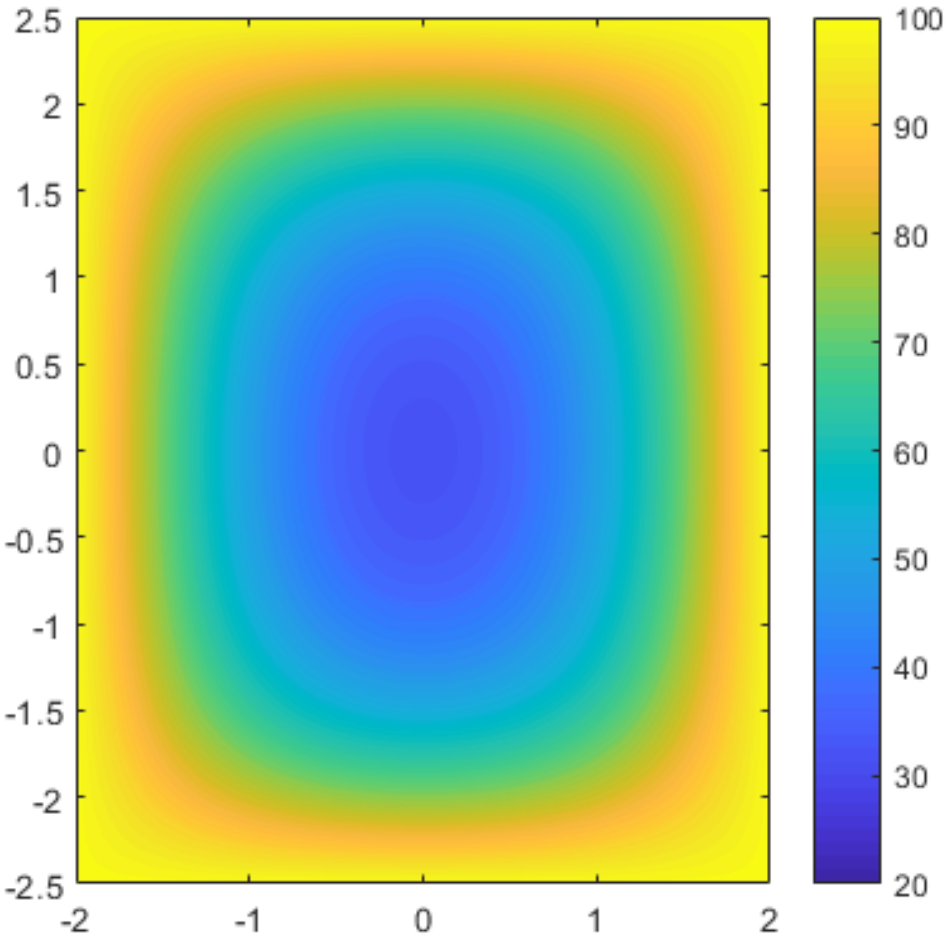

Figure 4: Temperature distribution at time t = 400 seconds.

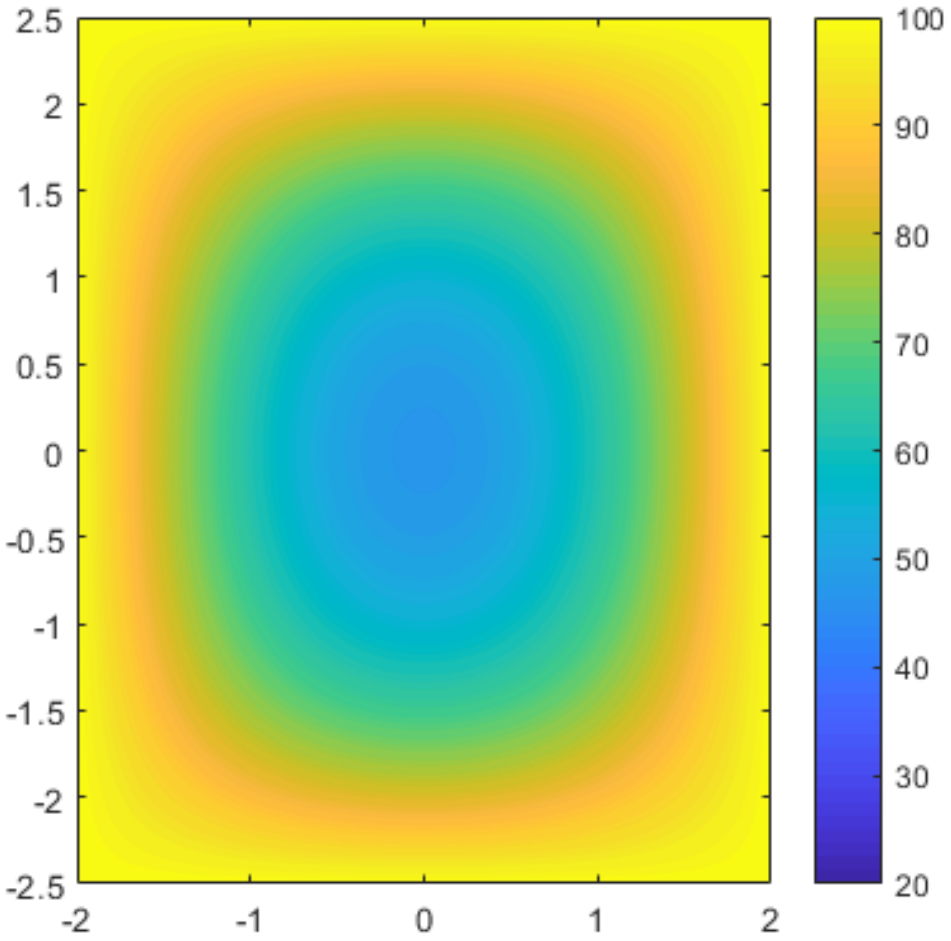

Figure 5: Temperature distribution at time t = 600 seconds.

5 Conclusion

The implicit scheme is useful for solving a diffusion equation primarily because you can choose a time-step at will, since it is unconditionally stable for any time-step. It is first-order accurate in space and can be used to model real world scenarios such as heat transfer within a potato. Also, it takes a surprisingly long time to cook a potato. That's probably why I have never bothered doing it.