CMIS18 - HW1

Isabela Blucher

April 27, 2018

Lecture 11 - Slide 39

For this exercise we have been given a small 2D Toy example, where we know that the conditions $\frac{\partial u}{\partial x} = 0$ and $\frac{\partial u}{\partial y} = 0$ are respectively satisfied on all vertical and horizontal boundaries. We use a 4-by-4 grid for the nodes in our domain, and a fifth row and column, added to all boundaries, as our ghost nodes. We have the following grid for our problem.

(0, 5)	(1, 5)	(2, 5)	(3, 5)	(4, 5)	(5, 5)
(0, 4)	(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)
(0, 3)	(1, 3)	(2, 3)	(3, 3)	(4, 3)	(5, 3)
(0, 2)	(1, 2)	(2, 2)	(3, 2)	(4, 2)	(5, 2)
(0, 1)	(1, 1)	(2, 1)	(3, 1)	(4, 1)	(5, 1)
(0, 0)	(1, 0)	(2, 0)	(3, 0)	(4, 0)	(5, 0)

In this image there is no spacing between the nodes, but we still consider the existence of a Δx and Δy.

Derive update formulas for ghost nodes

The image above shows the ghost nodes colored in red. To derive update formulas for all of them we can use our boundary constraints and central difference approximation. We want to find a formula for a ghost node based on the u values of the domain nodes.

Let's start with the left column of ghost nodes ($u_{0,j}$ for $j = 1, .., 4$). Since we know that for vertical boundary conditions we have $\frac{\partial u}{\partial x} = 0$, we can use that to find

$$\frac{\partial u_{\frac{1}{2},j}}{\partial x} = 0$$

From the slides in Lecture 11, we can see that a central difference approximation would imply

$$\frac{\partial u_{\frac{1}{2},j}}{\partial x} = \frac{u_{1,j} - u_{0,j}}{\Delta x} = 0 \implies u_{1,j} = u_{0,j}$$

If we repeat the process of applying a central difference approximation on the boundaries for the right column, top and bottom rows we get the following formulas.

$$\frac{\partial u_{\frac{9}{2},j}}{\partial x} = \frac{u_{5,j} - u_{4,j}}{\Delta x} = 0 \implies u_{4,j} = u_{5,j}$$

$$\frac{\partial u_{i,\frac{1}{2}}}{\partial y} = \frac{u_{i,1} - u_{i,0}}{\Delta y} = 0 \implies u_{i,1} = u_{i,0}$$

$$\frac{\partial u_{i,\frac{9}{2}}}{\partial y} = \frac{u_{i,5} - u_{i,4}}{\Delta y} = 0 \implies u_{i,4} = u_{i,5}$$

The four formulas above are used for $i, j \in \{1, 2, 3, 4\}$, which means we don't assign values to the corner ghost nodes, and for all the other ghost nodes we use the boundary domain value adjacent to it.

Derive update formulas for all domain nodes

For the given problem, we have $u(x, y) : \mathbb{R}^2 \to \mathbb{R}$ and the PDE $\nabla^2 u - \kappa^2 u = f(x, y)$, where we know $\kappa = 2$ and $f(x, y) = x + y$. By the slides, we know that we can approximate the term $\nabla^2 u$ by a 2nd order central difference, and then get the following formula

$$\frac{1}{\Delta x^2}u_{i-1,j} + \frac{1}{\Delta y^2}u_{i,j-1} + (-2^2 - \frac{2}{\Delta x^2} - \frac{2}{\Delta y^2})u_{i,j} + \frac{1}{\Delta y^2}u_{i,j+1} + \frac{1}{\Delta y^2}u_{i+1,j} = f(i, j)$$

Putting the $u_{i,j}$ term on the left hand-side we can find an equation that updates the u value for the node located at (i, j), making it dependent on its neighbor nodes (four adjacent nodes, two vertical and two horizontal), and also on the source term f.

$$u_{i,j} = \frac{(i + j) - \frac{1}{\Delta x^2}u_{i-1,j} - \frac{1}{\Delta y^2}u_{i,j-1} - \frac{1}{\Delta y^2}u_{i,j+1} - \frac{1}{\Delta x^2}u_{i+1,j}}{(-4 - \frac{2}{\Delta x^2} - \frac{2}{\Delta y^2})}$$

We can use this formula to update all our domain nodes, which means i, j in the formula above is a value in the set $\{1, 2, 3, 4\}$. We can observe that the u value will also be dependent on the coordinate of the current node being updated, due to $f(i, j) = i + j$, and the distance between horizontal and vertical nodes $\Delta x, \Delta y$.r

Lecture 11 - Slide 44

Explain how the approximation equations from the governing equations and boundary conditions are mapped into a matrix using index sets.

The index sets method allows us to preallocate a matrix that will receive the mapping results from both the domain nodes (governing equations) and the ghost nodes (boundary conditions).

To explain the mapping process, it is important to have in mind the node coefficients in the equation

$$\frac{1}{\Delta x^2}u_{i-1,j} + \frac{1}{\Delta y^2}u_{i,j-1} + (-2^2 - \frac{2}{\Delta x^2} - \frac{2}{\Delta y^2})u_{i,j} + \frac{1}{\Delta y^2}u_{i,j+1} + \frac{1}{\Delta y^2}u_{i+1,j} = f(i, j)$$

.

Let's start by explaining how the domain nodes were mapped into a matrix using index sets. We use the memory layout from the slides, where a pair of grid indexes (i, j) are mapped to a single vector index k. We also consider the stencil shape from the slides, where we have a central $u_{i,j}$ and four adjacent nodes (top, bottom, left right). We start the mapping by applying the formula $k = jN + i$ to the coordinates (i, j) of the central node. After that, we insert the node coefficient value $c_{i,j} = -\kappa^2 - \frac{2}{\Delta x^2} - \frac{2}{\Delta y^2}$ into the coordinates (k, k) of the matrix. Having done that, it is possible to now map coefficient values of the adjacent nodes of the stencil. Table 1 shows how the mapping of all the stencil coordinates work.

Grid coordinates	Coefficient value	Mapped coordinates
(i, j)	$-\kappa^2 - \frac{2}{\Delta x^2} - \frac{2}{\Delta y^2}$	$(jN + i, jN + i) = (k, k)$
$(i - 1, j)$	$\frac{1}{\Delta x^2}$	$(jN + i, jN + i - 1) = (k, k - 1)$
$(i + 1, j)$	$\frac{1}{\Delta x^2}$	$(jN + i, jN + i + 1) = (k, k + 1)$
$(i, j - 1)$	$\frac{1}{\Delta y^2}$	$(jN + i, (j - 1)N + i) = (k, k - N)$
$(i, j + 1)$	$\frac{1}{\Delta y^2}$	$(jN + i, (j + 1)N + i) = (k, k + N)$

Table 1: Formulas for the mapping of the 2D grid coordinates to a matrix

After having applied the formulas in the table above to all domain nodes, we start by mapping the ghost nodes. Again, it is important to have in mind the update formula, where we can see the coefficient values

$$(+1)u_{0,j} + (-1)u_{1,j} = 0$$

We can observe that the coefficients used here are +1 and -1. With that, we can use the same memory layout from the domain nodes applied here. We used fixed values of i and j, depending on what boundary is being updated. Table 2 shows how the mapping works, what coefficient value is being assigned to the new position, and what coordinate is fixed (which set of ghost nodes is being updated).

Grid coordinates	Fixed coordinate	Coefficient value	Mapped coordinates
(i, j)	j = 1	+1	(k, k), where $k = i$
(i, j + 1)	j = 1	-1	$(k, k + N)$, where $k = i$
(i, j)	j = N	+1	(k, k) where $k = (N - 1)N + i$
(i, j - 1)	j = N	-1	$(k, k - N)$ where $k = (N - 1)N + i$
(i, j)	i = 1	+1	(k, k) where $k = (j - 1)N + 1$
(i + 1, j)	i = 1	-1	$(k, k + 1)$ where $k = (j - 1)N + 1$
(i, j)	i = N	+1	(k, k) where $k = (j - 1)N + N$
(i - 1, j)	i = N	-1	$(k, k - 1)$ where $k = (j - 1)N + N$

Table 2: Formulas for the mapping of the ghost nodes from grid coordinates to matrix

After all that has been done, the rest of the ghost nodes, the corner ones, are mapped into the rows where we have all zero values. Since there is no update equation, and they are never accessed for computation of a central difference approximation, we can use an arbitrary coefficient value.

Lecture 11 - Slide 45

Examine the fill pattern and the eigenvalues of the matrix in the toy example

From the 2D toy problem (n = 4, $f(x, y) = x + y$, $\kappa = 2$ and $\Delta x = \Delta y = 1$), we have a filled coefficient matrix with its structure shown in the left-hand side of Figure 1.

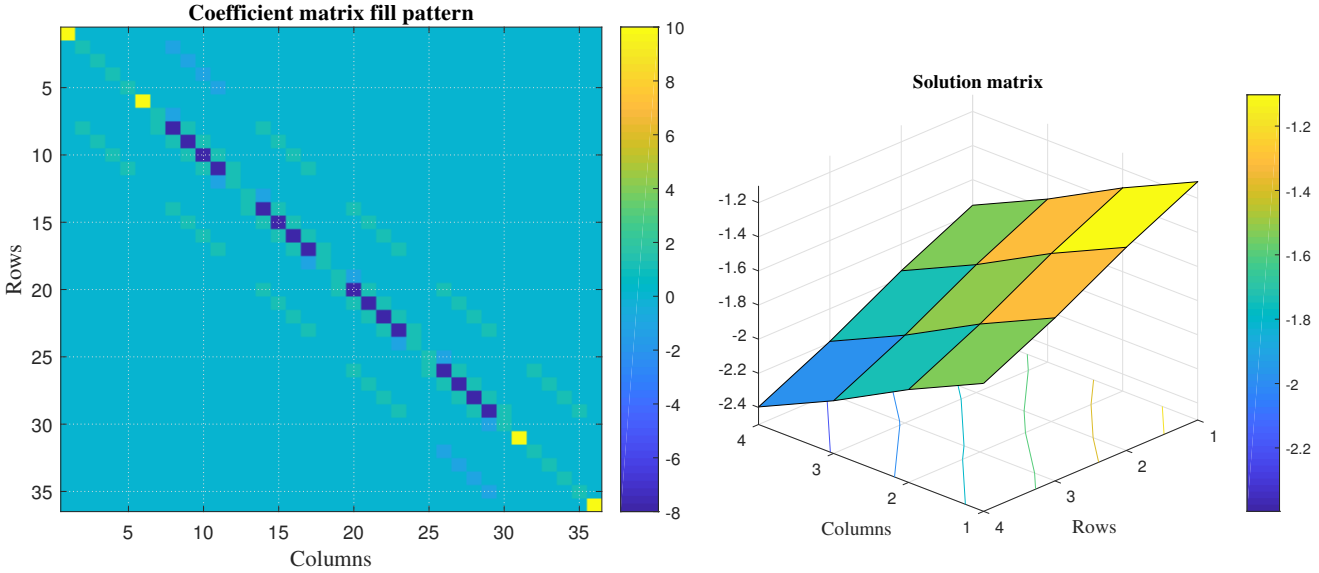


Figure 1: Plots of fill pattern and solution surface for 2D toy problem with parameters N = 4 and $\kappa = 2$

The yellow pixels are in the rows that represent corner ghost nodes. If they were not included our matrix would have zero rows and thus the matrix would be singular, so we set them with an arbitrary value of 10. The light blue pixels are in the rows that represent the ghost nodes that are not in the corners. The ones in the top and bottom of the outer diagonal lines are the ones with fixed j values and the ones mixed in the inner diagonal are the ones with fixed i values. The other pixels that are not equal to zero are the ones colored in green and dark purple. These rows represent our domain nodes. We can also see that the central stencil nodes ($u_{i,j}$) are all mapped into the dark purple pixels and located only in the inner diagonal.

The fill structure seems very symmetrical with this diagonal pattern, though the actual matrix is not. We can verify this by looking at the eigenvalues of A and seeing that some of them are negative, which means that it is not positive semidefinite, and therefore not symmetric.

When it comes to analyzing the eigenvalues, they are all real values so the speculation is that the linear system can be solved for u. We confirm this speculation by actually computing u for the 2D toy problem. The right-hand side of Figure 1 illustrates u after it has been reshaped into a matrix with only the domain nodes in it. By visual inspection of the surface plot, we don't see much of a ripple effect, which can be explained by a high value of $\kappa = 2$.

We can plot the matrix fill pattern and the u surface for different parameter values, to try and see if our model is experimentally sound (verification process).

For $\kappa = 0.001$ and $N = 10$, we get the following pattern and solution surface.

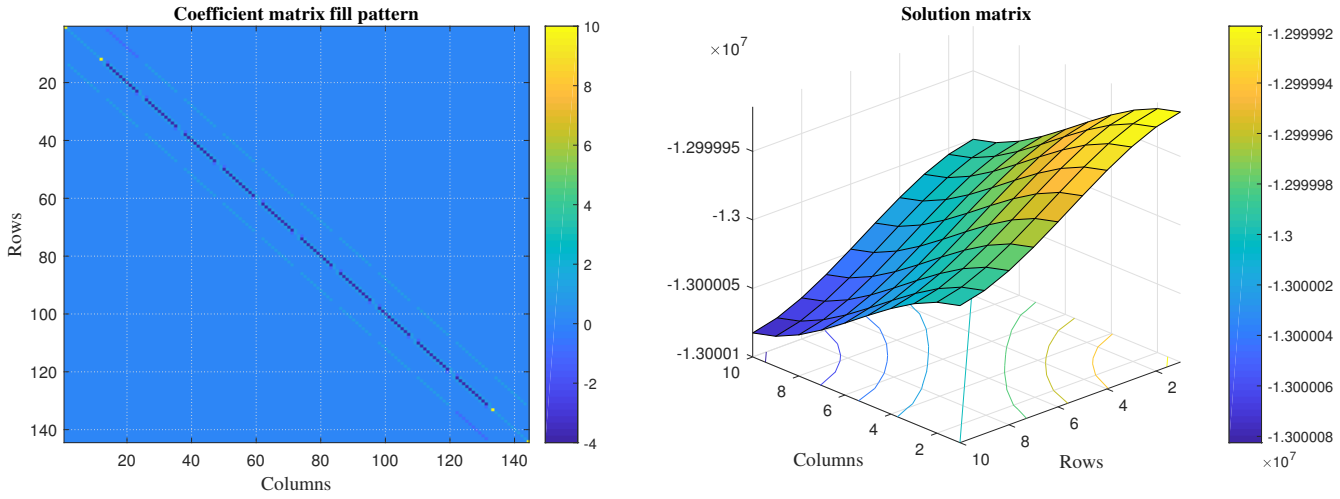


Figure 2: Plots of fill pattern and solution surface for 2D toy problem with parameters N = 10 and $\kappa = 0.001$

Here, by increasing the number of elements in our matrix and by diminishing κ, we can see a surface that is much smoother and has more of the ripple effect than with the original parameters. Examining the matrix fill pattern it is possible to observe that it is very similar to the one from the original problem.

Since we are dealing with a real-world problem, a heat equation, let's try and change the source function to see what would happen if the only source was at the center of the f matrix. Here we set N = 32, $\kappa = 0.001$ and we set $f(i, j) = 0$ everywhere but $f(16, 16) = 1$.

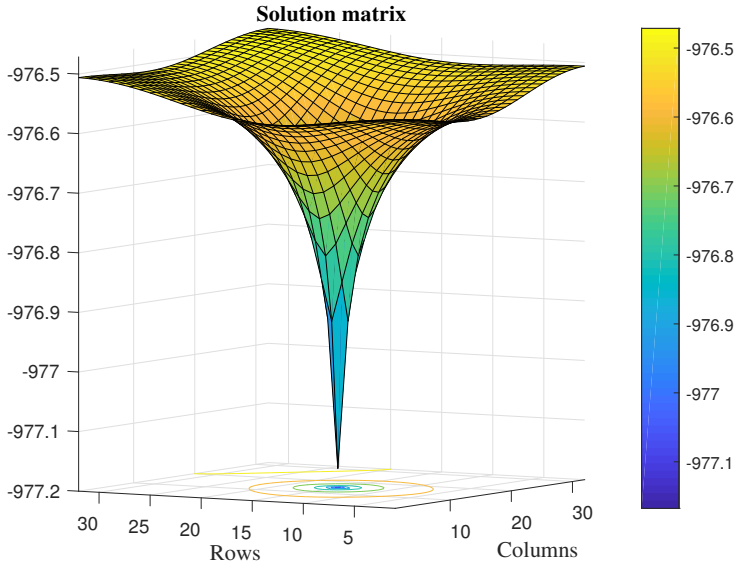


Figure 3: Solution matrix plotted as a surface

Here we can see that the heat leaves from the source in the center and gets spread out in a non-uniform way throughout the matrix.

In conclusion, the model seems to have similar results (the matrix structure does not change) and behavior for various parameter values, which is an indication that our numerical method is verifiable (agrees with the mathematical model).

Lecture 11 - Slide 46

We now consider a 1-dimensional example where we have the PDE $\frac{\partial^2 u}{\partial x^2} = 0$. Our domain gos from $x_a = 0$ to $x_b = 1$. We sample any $n > 3$ points in between, with $\Delta x = \frac{1}{n+1}$. We use a central difference approximation to

compute update equations for the domain nodes.

Write the finite difference approximation equation for the i^{th} node

We can derive an update equation for the i^{th} node of our mesh by using a central difference approximation of $\frac{\partial^2 u}{\partial x^2}$.

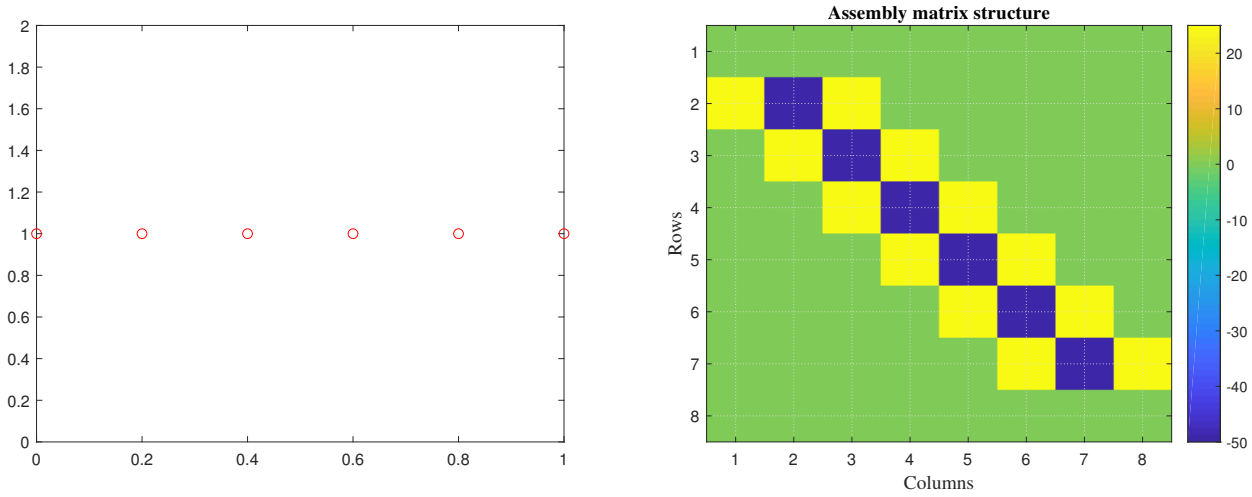
$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

Since we know that $\frac{\partial^2 u}{\partial x^2} = 0$, we can apply that to our approximation to obtain the update formula for u_i

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = 0 \implies u_i = \frac{u_{i+1} + u_{i-1}}{2}$$

Draw computational mesh and assemble matrix system

The parameter used for this simulation was $n = 4$. With that being defined, we can plot the computational mesh, which is the interval separation starting at 0 and ending at 1 with evenly spaced points sampled on that interval, and the assembly matrix structure, which was built using the coefficients found in the update formula derived above.



Plots of fill pattern and solution surface for 2D toy problem with parameters N = 10 and $\kappa = 0.001$

The assembly matrix structure is similar to the one from the 2-dimensional problem. We get a diagonal structure were the dark purple pixel represents the domain node u_i and the yellow nodes are its adjacent neighbors u_{i-1} and u_{i+1}. With the choice of $\frac{\partial u}{\partial x} = 0$ for the boundary condition, our matrix has two zero rows that correspond to the ghost nodes in our mesh. It is also possible to compute the eigenvalues and see that two of them are zero.

This means that our assembly matrix is singular. To be able to find a solution to our system, the matrix would have to be non-singular. To do that we can use the same geometric considerations for the ghost nodes and add the coefficient values to the zero rows of the matrix.