FVM Solver

Project Documentation

Developed and tested by Aditya Natu

Introduction (or, what can this code do for you?)

This code solves an equation of the following form over a rectangular domain for an unknown field T

$$\vec{\nabla}.(\vec{V}T) + \vec{\nabla}.\left(D_y \frac{\partial T}{\partial y}\hat{j} + D_z \frac{\partial T}{\partial z}\hat{k}\right) + P_{rp}T = R \tag{1}$$

Where,

$\vec{V} \equiv C_y\hat{j} + C_z\hat{k}$

$\vec{\nabla} \equiv \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k}$

The meanings of the terms are as follows:

$\hat{j},\ \hat{k}$ are the unit vectors along y and z directions respectively

$\vec{V}$ is the velocity vector

$D_y,\ D_z$ are the diffusion coefficients for the y and z components of gradient of T

The boundary conditions are

$$\alpha T + \beta \frac{\partial T}{\partial n} = G \tag{2}$$

Where,

n is the direction along the *outward normal* to the wall on which the boundary condition is applied.

The boundary conditions are applied on the following walls

$y = y_1.\ y = y_2,\ z = z_i,\ z = z_2$

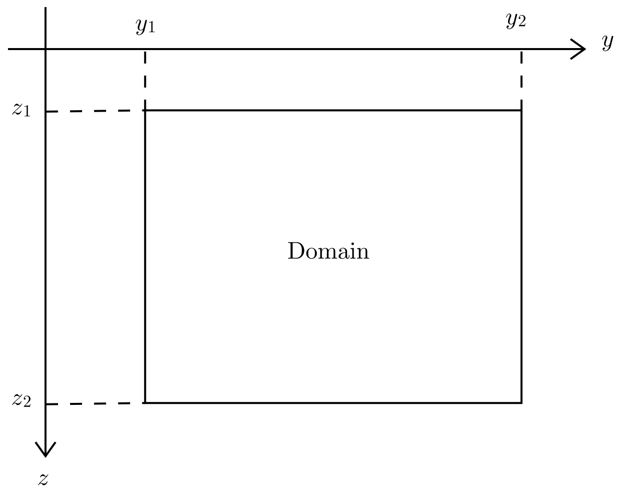


Fig. 1. The definition rectangular domain.

Types of grid

You have two options:

Option 1: Use the bundled function ci_walls.m to define a compound interest grid which has the largest cells at

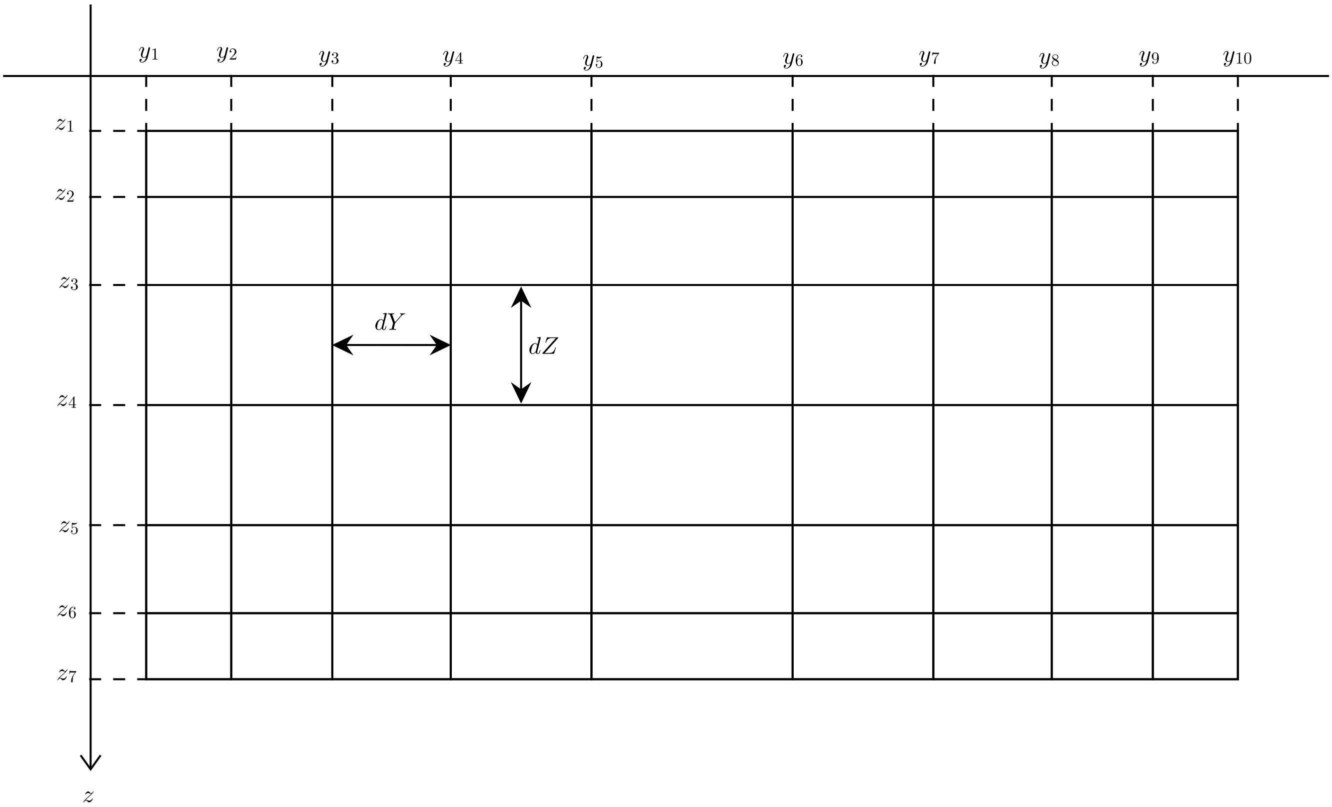


Fig. 2. The compound interest grid. The cells become larger as one moves towards the center. Each cell is surrounded by exactly 4 cells (except the cells at the boundary)

Option 2: Input a user-defined grid in which: (1) The domain is rectangular, (2) Each cell is rectangular with only vertical and horizontal walls, and (3) Each cell (except the boundary cells) is surrounded by exactly 4 cells.

Defining constants

The quantities C_y, C_z, D_y, D_z, P_{rp}, R can, in general, take a different value for *every cell* since the user inputs these quantities as matrices.

The quantities α, β, G can, in general, take a different value for *every cell* on the boundary since the user inputs these quantities as matrices.

Discretization

Discretization is handled automatically by the code using the *Finite Volume Method*. The equations are solved *implicitly. Central differencing* is used for discretization.

Instructions for use

Step 1: Setting up the grid

1.1. Deciding the number of cells you want
Let m be the number of cells along y, i.e. the number of *columns* of cells. For example in Fig. 2, $m = 9$.
Let n be the number of cells along z, i.e. the number of *rows* of cells. For example in Fig. 2, $n = 6$.

The total number of cells, i.e. $n \times m$ directly affects how computationally expensive the calculation gets. However, a large number of cells are unavoidable if you want to accurately capture complicated phenomena. One way to enjoy the best of both the worlds is to use smaller cells in regions where the phenomena of interest are prominent, and larger cells elsewhere.

1.2. Specifying the cell wall positions
Look at Fig. 2 again. You will see that the vertical walls are common for all cells in a particular column. Their $y-$ co-ordinates

are $y_1,\ y_2,\ y_3,\ \ldots$. The horizontal columns are common for all the cells in a particular row. Their $z-$co-ordinates are $z_1,\ z_2,\ z_3,\ \ldots$. You will need 2 matrices, of size $1\times(m+1)$ and $1\times(n+1)$. Let's call them y and z respectively. Then,

$$y = \begin{bmatrix} y_1 & y_2 & y_3 & \ldots & y_{m+1} \end{bmatrix}$$
$$z = \begin{bmatrix} z_1 & z_2 & z_3 & \ldots & z_{n+1} \end{bmatrix}$$

You have the option to generate these matrices using the code ci_walls.m. Simply call

```
y = ci_walls(y_first,y_last,m+1,r1);
```

Where,
y_first $= y_1$
y_last $= y_{m+1}$
r1 is the ratio of two adjacent cell widths (dY), the greater width divided by the lesser width. This is equivalent to $\mathrm{r1} = 1+c$ where c is the compound interest rate (as a fraction, *not percentage*). For example, if adjacent cells are such that the wider cell is 1.05 times the smaller one, r1=1.05

Similarly, one could do along z. The compound interest rate need not be common.

1.3. Creating the grid matrices

Now that you have the cell wall positions, you are now ready to create the grid. Simply call

```
[Y, Z, dY, dZ, Y_ext, Z_ext] = mk_grid(y,z);
```

The outputs are explained below:

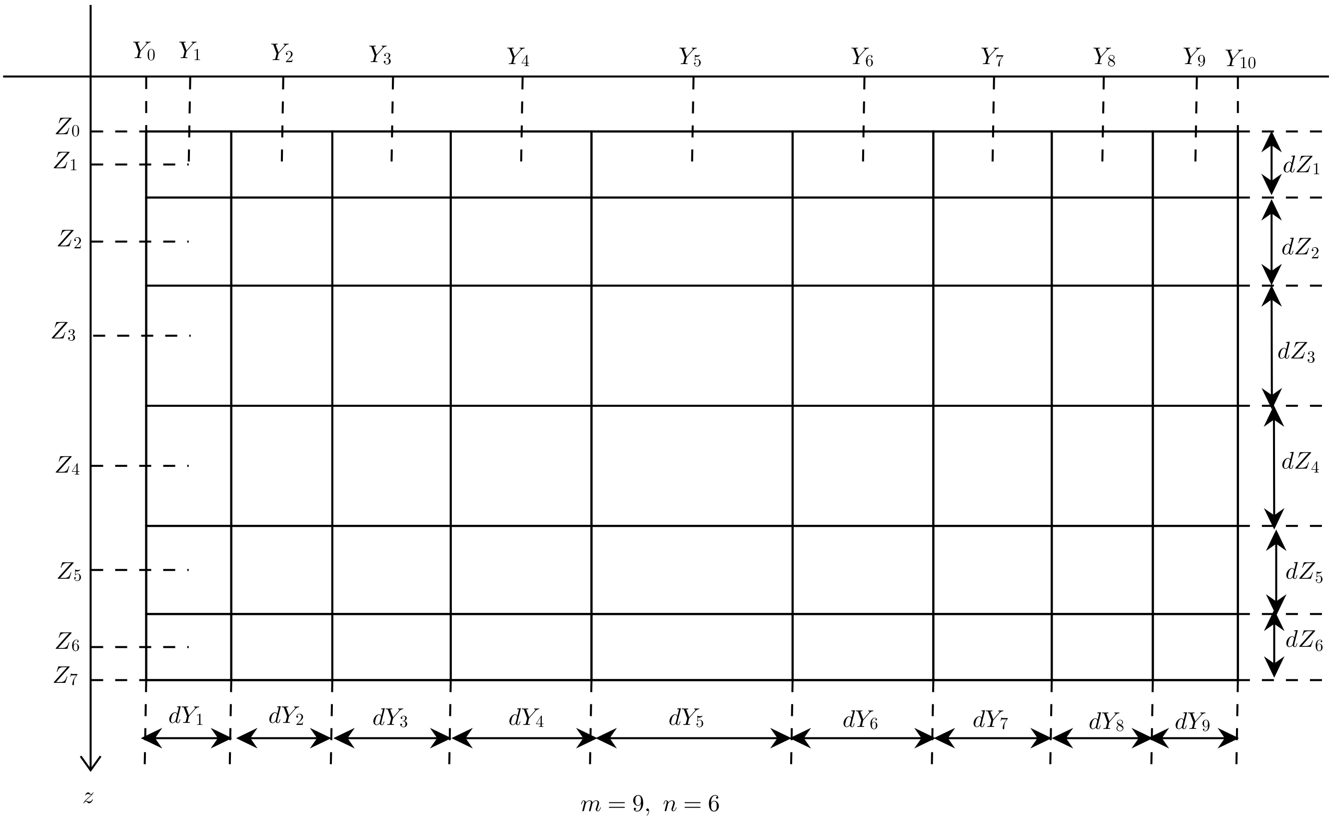


Fig. 3. Explanation of output of the function mk_grid.m. Refer to Table 1.

Table 1. List of outputs of mk_grid.m and their description

Output	Size	Description
Y	n × m	The $y-$co-ordinates of all cell centers. In the example of Fig. 3, the matrix contains n rows each containing m cells with values $Y_1,\ Y_2,\ Y_3,\ \ldots,\ Y_m$. All n rows are identical.
Z	n × m	The $z-$co-ordinates of all cell centers. In the example of Fig. 3, the matrix contains m columns each containing n cells with values $Z_1,\ Z_2,\ Z_3,\ \ldots,\ Z_n$. All m columns are identical.

dY	n × m	The width (dY) of all cells. The matrix contains n rows, each with m columns containing the values $dY_1,\ dY_2,\ dY_3,\ ...,\ dY_m$. All n rows are identical.
dZ	n × m	The height (dZ) of all cells. The matrix contains m columns, each with n rows containing the values $dZ_1,\ dZ_2,\ dZ_3,\ ...,\ dZ_m$. All m columns are identical.
Y_ext	(n+2) × (m+2)	The $y-$ co-ordinates of all cell centers and boundary walls. In the example of Fig. 3, the matrix contains n+2 rows each containing m+2 cells with values $Y_0,\ Y_1,\ Y_2,\ Y_3,\ ...,\ Y_{m+1}$. All n+2 rows are identical.
Z_ext	(n+2) × (m+2)	The $z-$ co-ordinates of all cell centers and boundary walls. In the example of Fig. 3, the matrix contains m+2 columns each containing n+2 cells with values $Z_0,\ Z_1,\ Z_2,\ Z_3,\ ...,\ Z_n,\ Z_{n+1}$. All m+2 columns are identical.

Step 2: Setting up the boundary conditions

Recall that there are two vertical and horizontal walls. The horizontal walls have m cells each and the vertical walls have n cells each.

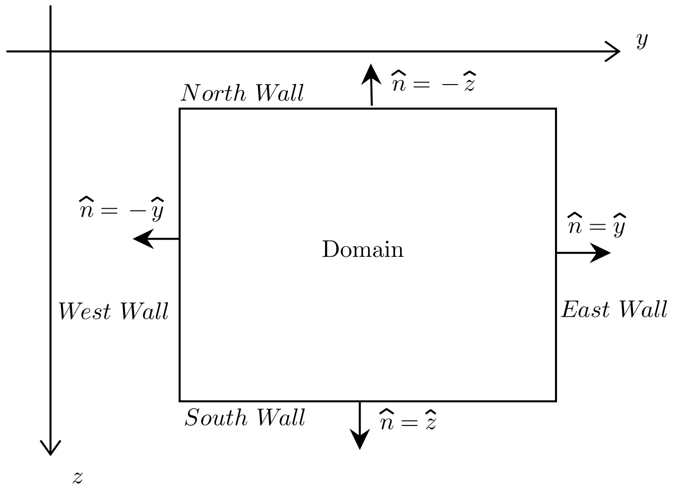


Fig. 4. Illustration of boundary walls.

Create 4 matrices 1 each for each matrix. Refer to table 2.

Table 2. Matrices for setting up the boundary conditions.

Matrix name (only for this example)	**Size**	**Description**	**Outward normal direction**	**Boundary condition equation form**
NB	3 × m	North wall boundary condition	$\widehat{n} = -\widehat{z}$	$\alpha T - \beta\dfrac{\partial T}{\partial z} = G$
SB	3 × m	South wall boundary condition	$\widehat{n} = \widehat{z}$	$\alpha T + \beta\dfrac{\partial T}{\partial z} = G$
WB	3 × n	West wall boundary condition	$\widehat{n} = -\widehat{y}$	$\alpha T - \beta\dfrac{\partial T}{\partial y} = G$
EB	3 × n	East wall boundary condition	$\widehat{n} = -\widehat{z}$	$\alpha T + \beta\dfrac{\partial T}{\partial y} = G$

Each matrix has 3 rows. They contain:

Table 3. Contents of each row in the boundary condition matrices

Row no.	Description
1	The value of α for each boundary cell. Set $\alpha = 0$ for Neumann boundary conditions.
2	The value of β for each boundary cell. Set $\beta = 0$ for Dirichlet boundary conditions.
3	The value of G for each boundary cell.

Refer to equation (2) for the meaning of each term.

Step 3: Setting up the matrices of known value.

Refer to **Table 4.** for the details on setting up the matrices seen in equation (1)

Table 4. Known quantities to be entered, and their meanings

Matrix name	Size	Description
C_y	(n+2) × (m+2)	The value of y-component of the advection velocity at each cell center, and the center of boundary walls of boundary cells.
C_z	(n+2) × (m+2)	The value of z-component of the advection velocity at each cell center, and the center of boundary walls of boundary cells.
D_y	(n+2) × (m+2)	The value of the diffusion coefficient for y-gradient at each cell center, and the center of boundary walls of boundary cells.
D_z	(n+2) × (m+2)	The value of the diffusion coefficient for z-gradient at each cell center, and the center of boundary walls of boundary cells.
P_{rp}	(n+2) × (m+2)	The value of the coefficient of the proportional term at each cell center, and the center of boundary walls of boundary cells.
R	(n+2) × (m+2)	The value of the source term at each cell center, and the center of boundary walls of boundary cells.

Step 4: Solving the equation:

To solve the said equation, simply call

```
[T_ext] = solve_eqn(Cy,Cz,Dy,Dz,Prp,RHS,dY,dZ,NB,EB,WB,SB);
```

T_ext is a (n+2) × (m+2) matrix that contains the values of T at each cell center, and the center of boundary walls of boundary cells.

Step 3: Calculating gradients

Calculating $dT\,/\,dy$

Simply call:

```
dT_dy = d_dy(T_ext,dY);
```

dT_dy is a (n+2) × (m+2) matrix that contains the values of $dT\,/\,dy$ at each cell center, and the center of boundary walls of boundary cells.

Calculating $dT\,/\,dz$

Simply call:

```
dT_dz = d_dz(T_ext,dZ);
```

dT_dz is a (n+2) × (m+2) matrix that contains the values of $dT\,/\,dz$ at each cell center, and the center of boundary walls of boundary cells.

Calculating $d^2T\,/\,dy^2$

Simply call:

```
d2T_dy2 = d2_dy2(T_ext,dY);
```

d2T_dy2 is a (n+2) × (m+2) matrix that contains the values of $d^2T\,/\,dy^2$ at each cell center, and the center of boundary walls of boundary cells.

Calculating $d^2T\,/\,dz^2$

Simply call:

```
d2T_dz2 = d2_dz2(T_ext,dZ);
```

d2T_dz2 is a (n+2) × (m+2) matrix that contains the values of $d^2T\,/\,dz^2$ at each cell center, and the center of boundary walls of boundary cells.

Example

Problem: Let us solve for T in an example where

$$\vec{\nabla}.\,(\vec{V}T) + \vec{\nabla}.\left(D_y\frac{\partial T}{\partial y}\hat{j} + D_z\frac{\partial T}{\partial z}\hat{k}\right) + P_{rp}T = R,$$

$$D_y = 0.1(y)(1-y)$$
$$D_z = 0.1(z)(2-z)$$
$$P_{rp} \;=\; -0.01\left(y^2+z^2\right)$$
$$R = 0.01\sqrt{y^2+z^2}$$

$$\vec{V} \equiv C_y\hat{j} + C_z\hat{k}$$
$$C_y = \sin(\pi y)\cos\left(\frac{\pi}{2} - \pi z\right)$$
$$C_z = \sin(2\pi y)\cos\left(\frac{\pi}{2} - 2\pi z\right)$$

$$North\ wall: z=0,\ yT-(1-y)\frac{\partial T}{\partial z} = 1$$
$$South\ wall: z=2,\ yT+(1-y)\frac{\partial T}{\partial z} = 0$$
$$East\ wall: y=1,\ zT+(2-z)\frac{\partial T}{\partial y} = 0$$
$$West\ wall: y=0,\ zT-(2-z)\frac{\partial T}{\partial y} = 1$$

on a rectangular domain divided into a compound interest grid, with a compound interest of 2% along z direction, and 1% along y direction. There should be 200 elements along z-direction and 100 along y-direction. Plot T, $\frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}, \frac{\partial^2 T}{\partial y^2}, \frac{\partial^2 T}{\partial z^2}$

Solution:

Step 1: We begin by defining cell wall positions as follows:

```
y = ci_walls(0,1,101,1.01); % m = 100
z = ci_walls(0,2,201,1.02); % n = 200
```

Step 2: Create the grid matrices by calling

```
[Y, Z, dY, dZ, Y_ext, Z_ext] = mk_grid(y,z);
```

Step 3: Define the boundary condition matrices by calling:

```
NB = [Y(1,:) ;1-Y(1,:) ; ones(1,100)];
SB = [Y(1,:) ;1-Y(1,:) ;zeros(1,100)];
EB = [Z(:,1)';2-Z(:,1)';zeros(1,200)];
WB = [Z(:,1)';2-Z(:,1)'; ones(1,200)];
```

(Refer to table 2 and 3)

Step 4: Define the matrices for known quantities as follows:

```
Cy = sin(pi*Y_ext).*cos(pi/2 - pi*Z_ext);
Cz = sin(2*pi*Y_ext).*cos(pi/2 - 2*pi*Z_ext);
Dy = 0.1*Y_ext.*(1-Y_ext);
Dz = 0.1*Z_ext.*(2-Z_ext);
Prp = -0.01*(Y_ext.^2 + Z_ext.^2);
RHS = 0.01*sqrt(Y_ext.^2 + Z_ext.^2);
```

Step 5: Solve the set of equations by calling

```
T_ext = solve_eqn(Cy,Cz,Dy,Dz,Prp,R,dY,dZ,NB,EB,WB,SB);
```

Step 6: Calculate the gradients by calling

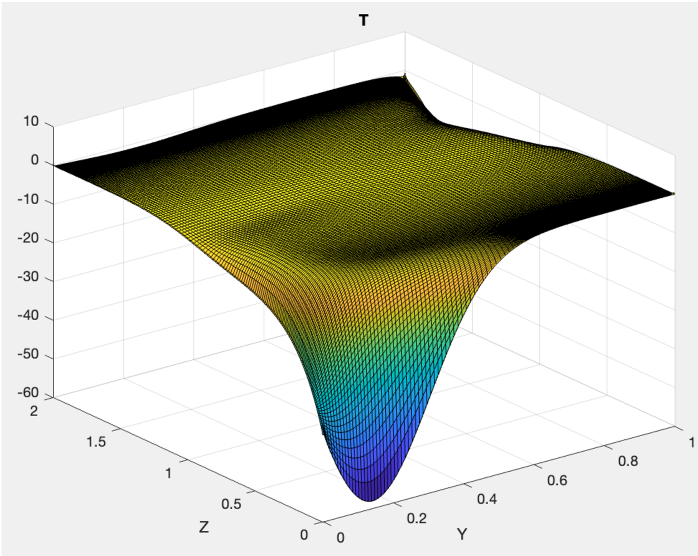
```
dT_dy = d_dy(T_ext,dY);
dT_dz = d_dz(T_ext,dZ);
d2T_dy2 = d2_dy2(T_ext,dY);
d2T_dz2 = d2_dz2(T_ext,dZ);
```

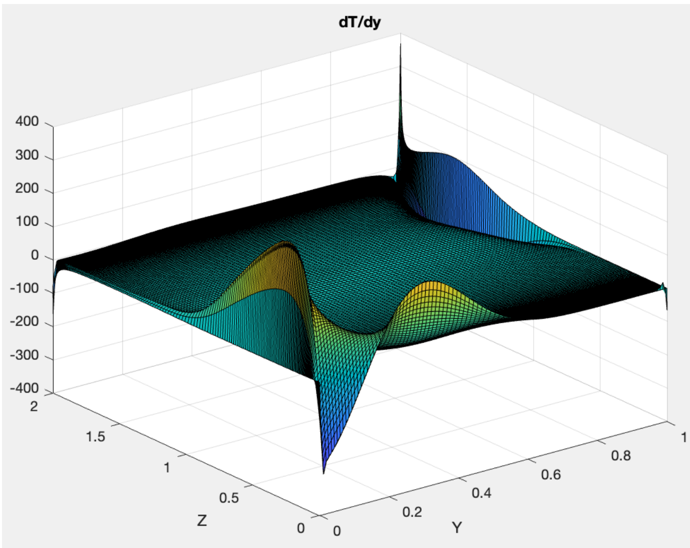
Step 7: Plot the results>> surf(Y_ext,Z_ext,T_ext)

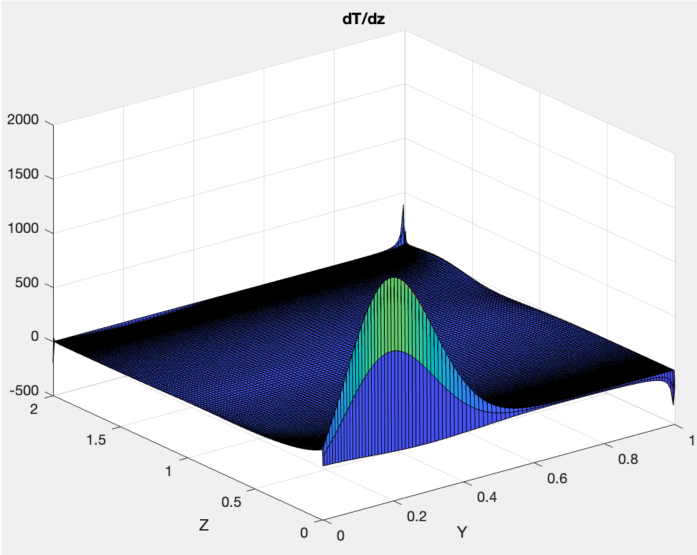
```
figure
surf(Y_ext,Z_ext,T_ext)
xlabel('Y')
ylabel('Z')
title('T')

figure
surf(Y_ext,Z_ext,dT_dy)
xlabel('Y')
ylabel('Z')
title('dT/dy')

figure
surf(Y_ext,Z_ext,dT_dz)
xlabel('Y')
ylabel('Z')
title('dT/dz')

figure
surf(Y_ext,Z_ext,d2T_dy2)
xlabel('Y')
```

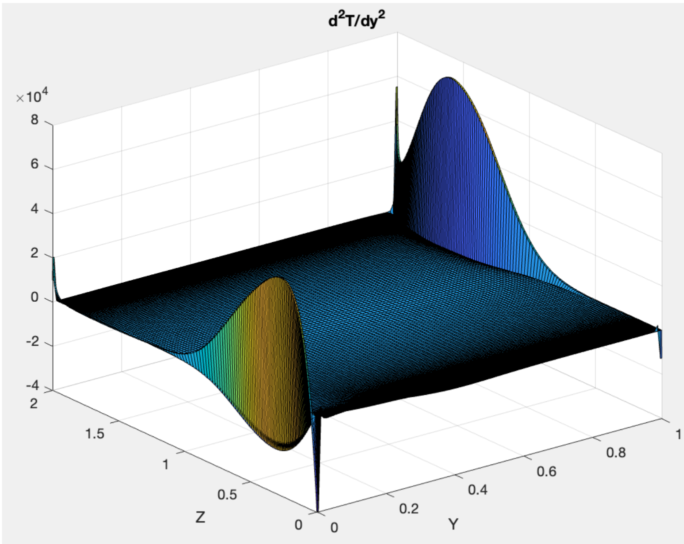
```
ylabel('Z')
title('d^2T/dy^2')

figure
surf(Y_ext,Z_ext,d2T_dz2)
xlabel('Y')
ylabel('Z')
title('d^2T/dz^2')
```

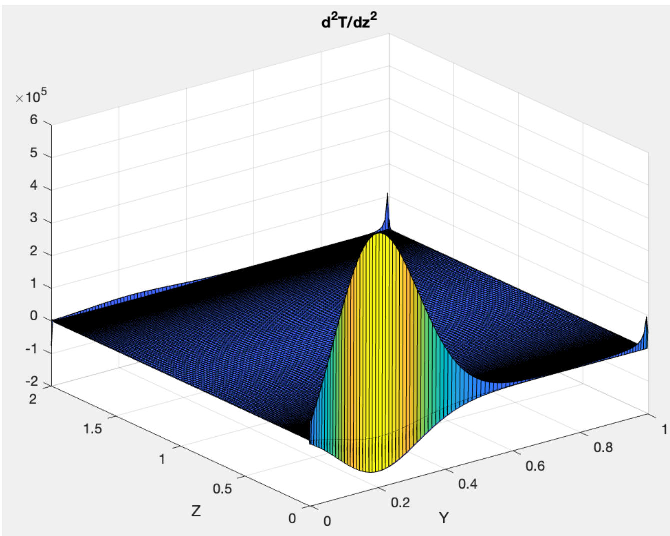
Find the plots for this example below:






dT/dz
2000
1500
1000
500
0
-500
2
1.5
1
0.5
0
Z
0
0.2
0.4
0.6
0.8
1
Y


d2T/dy2
×10^4
8
6
4
2
0
-2
-4
2
1.5
1
0.5
0
Z
0
0.2
0.4
0.6
0.8
1
Y


d2T/dz2
×10^5
6
5
4
3
2
1
0
-1
-2
2
1.5
1
0.5
0
Z
0
0.2
0.4
0.6
0.8
1
Y