University of Cape Town • iYunivesithi yaseKapa • Universiteit van Kaapstad

SPES BONA

Flow diagram

Provide your peoplesoft ID as an input, the get_parameter function will then fetch the specific parameters assigned to that student ID. Then each value fetched is assigned to a1,b1,a2… which are all in a list called parameters_list

Input the desired number of nodes along and width. The coordinates of each node and their corresponding list of elements will be produced by a function called in the code

The user must then input the order of integration desired. The inputed variables will be used for the rest of the system

The function find_ecm will have the function of locating the k, conductivity matrix through the process of Gauss quadrature.

The assemble_gcm function provided will be used to find the gcm matrix and the input used will come from the find_ecm function.

The gcm value produced will be the unconstrained value. This unconstrained value alongside parameters will be inputs for the make_constrained_system function

The result of the make_constrained_system function produces the constrained K value and the constrained heat vector

The Cholesky function then calculates the L matrix and thereafter U can be calculated as it is the transpose of L

After U is found, forward and back substitution will take place to calculate the temperatures. Each condition (set) is considered and then the temperature is used to produce a scatter plot in correlation with the coordinates.

Flow chart bullet point as flow diagram is more general.

1. start
2. Enter peoplesoft number
3. get_parameters
4. get_paramaters outputs a and b specific values
5. n and m size to be entered MIN=10
6. get_node_element _arrays
7. get_node_element _arrays ouputs node list and element list
8. enter order and element number
9. find_ecm which outpouts the find ecm matrix
10. Assemble_gcm with outputs gcm matrix
11. Make_constrained_system which outputs contrained K and H
12. Solve for T at the nodes
13. Cholesky to get L and then U
14. Plot
15. end

Get Parameters

```
Run Cell | Run Above | Debug Cell
#%%
def getparameters():
    EMPL_ID = input("Enter student code:\n")

    data = []

    with open("allocation.csv") as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            if row[0].startswith(EMPL_ID):
                data.append(row)
    return data
Run Cell | Run Above | Debug Cell
#%%
parameters = getparameters()

#  getting list of parameters, accessing list made in get paramters but now i am creating  new a list
# setting a1.b1.a3.b2.b3 to the correct values so that they can be accessed later on

a1 = float(parameters[0][1])
b1 = float(parameters[0][2])
a2 = float(parameters[0][3])
b2 = float(parameters[0][4])
a3 = float(parameters[0][5])
b3 = float(parameters[0][6])


parameters_list = [a1,b1,a2,b2,a3,b3]
```

The function get_parameters is defined. It prompts the user to enter a PeopleSoft code, “EMPL_ID” and then reads from a CSV file which is the name “allocations.csv” and then returns a list of rows that match the entered EMPL_ID if it was a valid ID.

The function first initializes an empty list. Then the open() function I used to open the “allocation csv file”, and creates a csv.reader object to read the file. Then the function iterates through each of the rows in the file through the used of a for loop and in the process it checks if the first column matches, if this is the case the append() method is used to add the data. If none of the rows match the function then prints "invalid peoplesoft ID"

The function then returns the data list which now contains all the rows from the CSV file. After this a1,b1,a2 ect are assigned to each correct value from the data list which can then be accessed later on in the code. (get_paramters assumes that the allocation csv file is in the same folder which is allowed)

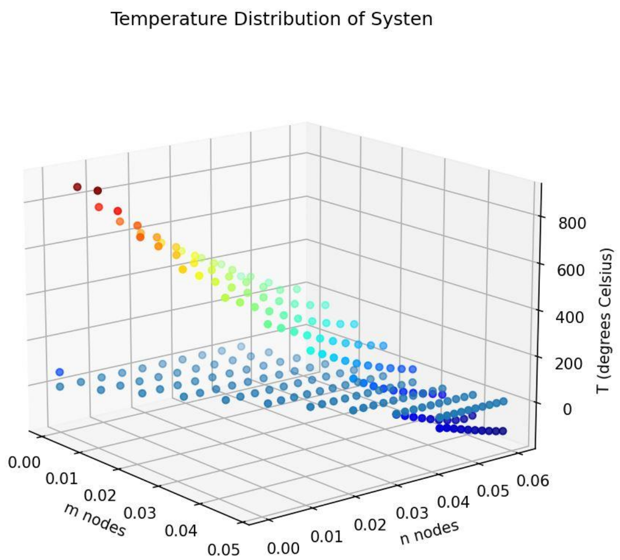
Minimum Number of Mesh Points

The accuracy of the temperature simulation distribution across the plate is dependent on the mesh size. Increasing the number of mesh points will improve the accuracy of the simulation however this will also increase the computational cost of running the simulation. In my system the user can decide on the number of mesh points, too few will result in an inaccurate simulation. If 3 points are used only one node will be in the center of the plate, which will lead to significant errors in the simulation as the temperature distribution is not accounted for fully. In my system the minimum is 10 for m and n to achieve accurate results.

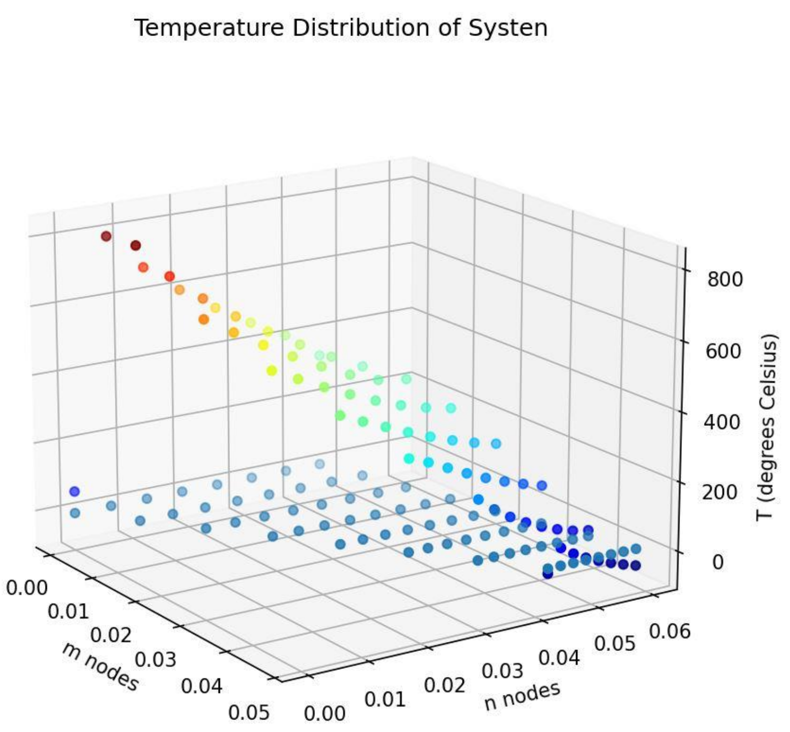
Gauss Reduction vs LU factorization

The two methods are used to solve systems of linear equations. They are used when more than one calculation is required as they are expensive. Gauss reduction must be performed separately for each calculation which results in it being very computationally expensive where as LU factorization can be performed for multiple calculations while only the right-hand side of the equation will change and this will help reduce the computational cost when doing large calculations. Therefore when it is a one-off calculation gauss reduction could be used however when doing large calculations repeatedly it makes more sense to make use of LU factorization or other iterative techniques such as the Jacobian.

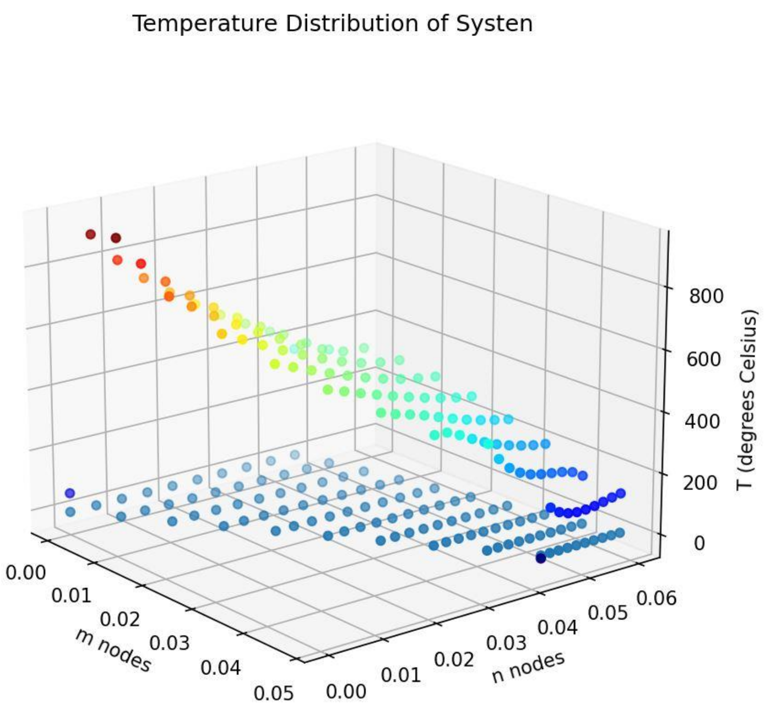
Temperature Distribution Plot



A1,B1


Temperature Distribution of Systen
T (degrees Celsius)
m nodes
n nodes
0
200
400
600
800
0.00
0.01
0.02
0.03
0.04
0.05
0.06

A2,B2


Temperature Distribution of Systen
T (degrees Celsius)
m nodes
n nodes
0
200
400
600
800
0.00
0.01
0.02
0.03
0.04
0.05
0.06

A3,B3