Cassandra Cantu
UID: 305-100-205
CEE/MAE M20 Summer 2020
July 1st 2020

Homework 01

1 Calculating the exact and approximate surface area of an oblate spheroid

1.1 Introduction

The goal in this problem is to develop a script that calculates the exact and approximate surface of an oblate spheroid given the user-specified equatorial and polar radii.
After, the results up to 10 digits will be printed to the screen and any differences between the two values will be discussed.

1.2 Model and Theory

The applicable equations for this particular problem are:

$$\gamma = \arccos\left(\frac{r_2}{r_1}\right)$$

$$exact = 2\pi\left(r_1^2 + \frac{r_2^2}{\sin(\gamma)}\ln\left(\frac{\cos(\gamma)}{1-\sin(\gamma)}\right)\right)$$

$$approx = 4\pi\left(\frac{r_1+r_2}{2}\right)^2$$

Where

- γ = gamma
- exact = exact surface area
- approx = approximate surface area
- r_1 = equatorial radius
- r_2 = polar radius

1.3 Methods and Pseudocode

The flow of calculate should be as follows:

1. Prompt user to input equatorial (r1) and polar (r2) radii
    - Error check user input for non-zero and non-negative numbers
    - Error check user input for condition that r2 must be less than r1
2. Define functions
    - Let `g = acos(r2/r1)` (γ function)
    - Let `exact = 2*pi*(r1^2+((r2^2/sin(g))*log(cos(g)/(1-sin(g)))))` (exact function)
    - Let `approx = 4*pi*((r1+r2)/2)^2` (approx function)
3. Calculate `g` function using g via the above equations
4. Calculate the exact surface area using `exact` via the above equations
5. Calculate the approximate surface area using `approx` via the above equations
6. Prints results to the command window

- Displays 10 digits by using `%10.4e` as format specifier

1.4 Calculations and Results

With the given data for Earth where the equatorial radius is 6378.137 and the polar radius is 6356.752, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
1
Please enter the equatorial radius of the oblate spheroid: 6378.137
Please enter the polar radius of the oblate spheroid: 6356.752
The exact surface area is: 5.1007e+08
The surface area approximation is: 5.0950e+08
```

With an example equatorial radius of -1, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
1
Please enter the equatorial radius of the oblate spheroid: -1
The surface area cannot be solved given this value. Please enter an
appropriate value.
```

With an example equatorial radius of 5 and polar radius of 10, a case that violates the condition that r2 must be less than r1, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
1
Please enter the equatorial radius of the oblate spheroid: 5
Please enter the polar radius of the oblate spheroid: 10
The surface area cannot be solved given these values. Please enter
appropriate values.
```

1.5 Discussions and Conclusions

It can be shown that the exact and approximate surface area calculations for Earth's equatorial and polar radii have the same precision to one digit. The exact surface area is a slightly larger value than the approximate surface area.

2 Identifying the neighbors of a given cell in a rectangular array

2.1 Introduction

The goal in this problem is to develop a script that lists the diagonal and orthogonal numerical neighbors of an identified cell in a rectangular array. The user-input number for rows and columns make up the rectangular array. Depending if the identified cell is in the interior, is an edge but not corner cell, or is a corner cell, it can have 8, 5, or 3 neighbors respectively. The results will display the cell identity on one line and the neighbors on the next line.

2.2 Model and Theory

To find the possible orthogonal neighbors of the identified cell, adding/subtracting 1 from the identified cell will give the below/above neighbors while adding/subtracting M will give the right/left neighbors. The combinations of ± 1 and $\pm$M will give the diagonal neighbors. Examples of applicable equations for this particular problem are:

P_left = P-M;
P_up = P-1;
P_downright = P+M+1;

Where

- M = number of rows
- N = number of columns
- P = identified cell
- P_left = left orthogonal neighbor of P
- P_up = above orthogonal neighbor of P
- P_downright = lower right diagonal neighbor of P

2.3 Methods and Pseudocode

The flow of calculate should be as follows:

1. Prompt user to input number of rows (M), columns (N), and the identified cell (P)
    - Error check user input for positive integers
    - Error check user input for M, N greater than 2 and for P between 1 and N*M
2. Calculate all possible neighbors of P
    - Via above and similar equations
3. Identify location of P and its neighbors with if/else statements:
    - Corners (3 neighbors): uses conditions in an if statement to identify if P is the upper left, upper right, lower left, or lower right corner cell
        - Example when P is the upper left corner cell: `if (P==1)`
        - 3 neighbors: lists the two possible orthogonal and one possible diagonal neighbor for each corner cell depending on location under the if statement
            - Example when P is the upper left corner cell:
              `neighbors = [P_down, P_right, P_downright];`
    - Edges (not corner): uses two conditions to identify if P is on the edge and in between two corner cells
        - Example when P is a left edge cell: `elseif (1<P)&&(P<M)`
        - 5 neighbors: lists the three possible orthogonal and two possible diagonal neighbors for each edge depending on side under the if statement
            - Example when P is a left edge cell:
              `neighbors = [P_up, P_down, P_upright, P_right, P_downright];`
    - Interior: If P doesn't fall under the categories of corners or edges, it will be considered an interior cell via an else statement.
        - 8 neighbors: list all possible neighbors of P under the else statement
4. Print cell ID and neighbors in command window on separate lines
    - Spaces in quotation marks and after format specifier to fit format guidelines

2.4 Calculations and Results

With an example of 4 rows and 6 columns and an identity cell of 1 (corner), the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: 4
Please enter the number of columns in the array: 6
Please enter the number of the cell ID in the array: 1
Cell ID:   1
Neighbors: 2  5  6
```

With an example of 4 rows and 6 columns and an identity cell of 5 (edge that's not a corner), the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: 4
Please enter the number of columns in the array: 6
Please enter the number of the cell ID in the array: 5
Cell ID:   5
Neighbors: 1  2  6  9  10
```

With an example of 4 rows, 6 columns, and an identity cell of 18 (interior), the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: 4
Please enter the number of columns in the array: 6
Please enter the number of the cell ID in the array: 18
Cell ID:   18
Neighbors: 13  14  15  17  19  21  22  23
```

With an example of -1 rows, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: -1
Error using Cantu_305100205_HW_01_main (line 48)
Invalid value for number of rows
```

With an example of 3 rows and 4.1 columns, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: 3
Please enter the number of columns in the array: 4.1
Error using Cantu_305100205_HW_01_main (line 54)
Invalid value for number of columns
```

With an example of 4 rows, 6 columns, and an identity cell of 25, the resulting output from the script is:

```
Which problem do you want to run? Enter 1 for oblate spheroid calculations or
2 for neighbor identification.
2
Please enter the number of rows in the array: 4
Please enter the number of columns in the array: 6
Please enter the number of the cell ID in the array: 25
Error using Cantu_305100205_HW_01_main (line 60)
Invalid value for number of target cell
```

2.5 Discussions and Conclusions

To apply this to a 3D grid, I would treat each “sheet” of rows and columns as the above 2D rectangular array. I would still start the indexing at the upper left corner of the first sheet. Once we reach the lower right corner of the first sheet, the count will continue to the upper left corner of the next sheet. This indexing will continue until we reach the lower right corner of the last sheet. We can classify the possible locations and number of neighbors in the following way for a 3 x 3 x 3 3D matrix:

- Corner: 7 neighbors – 3 orthogonal and 4 diagonal
- Edge: 11 neighbors – 4 orthogonal and 7 diagonal
- Outside face (not edge or corner): 17 neighbors – 5 orthogonal and 12 diagonal
- Interior: 26 neighbors – 6 orthogonal and 20 diagonal