

Assignment 5 Code Peer Reviews Due: Thursday, 3/07/19, 11:59pm

Assignment 5 Back Evaluations: Sunday, 3/10/19, 11:59pm

Go to Peerceptiv and complete your 3 peer reviews by Thursday at midnight. After Thursday at midnight, you will evaluate how helpful your peer reviews are by Sunday at midnight.

Assignment #6: 1-D and 2-D Arrays (150 pts)

Design Due: Sunday, 3/10/19, 11:59pm

Design Peer Reviews Due: Thursday, 3/14/19, 11:59pm

Code and Design Back Evaluations: Sunday, 3/17/19, 11:59pm

Make sure you demo Assignment #5 within two weeks of the due date to receive full credit. If you go outside the two-week limit without permission, you will lose 50 points. If you fail to show up for your demo without informing anyone, then you will automatically lose 10 points.

(10 pts) Assignment 5 Reflective Post-Peer Review Survey:

http://oregonstate.qualtrics.com/jfe/form/SV_0cuJBjYqkCe8SJD

Problem Statement

You will apply your knowledge of arrays to gain an understanding of a heat transfer model over one- and two-dimensional objects. To do this, you will implement the explicit method for solving finite difference approximations. Your program will simulate the diffusion of heat through a 1-D or 2-D object, such as a wire or plate, using the explicit method to solve for new time instances. Since diffusion is the rate of change, then derivatives are needed to calculate this rate of change in heat over time.

This following equation is how you calculate the diffusion of heat in a 1-D object over time:

$$k \frac{\partial^2 u}{\partial x^2} = c\rho \frac{\partial u}{\partial t}$$

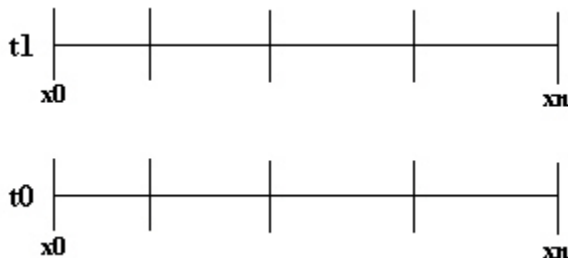
To solve this, it would require techniques from calculus, but we can't do calculus/take a derivative on the computer like we would do by hand. Instead, we need to approximate the derivative, and one approximation method is the explicit method for finite difference approximations to solve this equation.

$$k \frac{u_{x+1}^t - 2u_x^t + u_{x-1}^t}{\Delta x^2} = c\rho \frac{u_x^{t+\Delta t} - u_x^t}{\Delta t}$$

Where the inputs are as follows:

k	Thermal conductivity
ρ (rho)	Material density
c	Specific heat capacity of material
u	1-D object
x	Location on u
t	Time

Note, the symbol Δ represents the change in a variable. For example, Δx is the change in x , and Δt is the change in time. The change in x , Δx , is calculated by knowing the length of the 1-D object and how many sections you want to calculate the diffusion of the heat for. For example, a wire that is 5 inches divided into 5 sections have 1-inch sections, i.e. Δx is 1. Similarly, the change in time, Δt , is calculated by knowing how long to run the simulation and for how many time instances to calculate the diffusion. For example, we want to simulate the heat diffusion for 10 minutes at 5 time instances is Δt of 2 minutes. Below shows a picture of the first two time instances for a 1-D object broken into 5 sections.



$$u_x^{t+\Delta t}$$

Now, we want to solve for:

Lastly, we want to make sure that our simulation is stable. Use this equation:

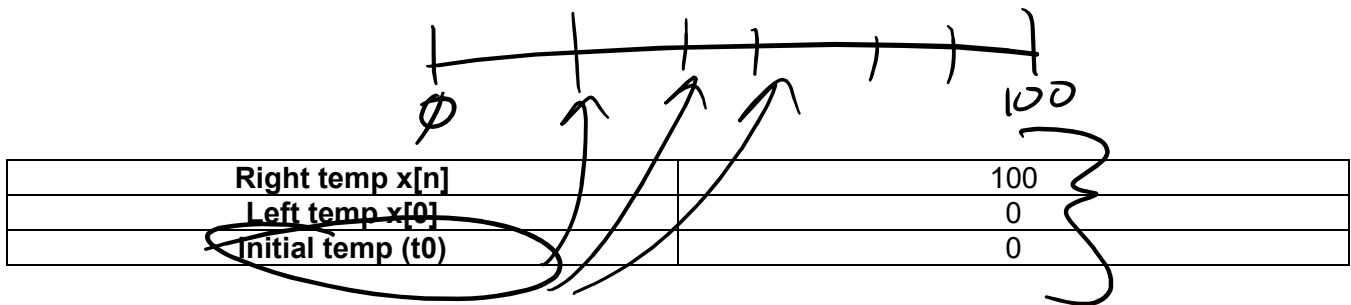
$$\left| \frac{k * \Delta t}{\Delta x^2 * c * \rho} \right|$$

Any value below **0.5** is **stable**, otherwise it is **unstable**.

Start by using the material properties of Nickel:

k	52.4
ρ (rho)	0.321
c	0.12
Wire length	10
Sections	10
Time intervals	50
Time of Simulation	0.01675

Handwritten notes:
 Δx (pointing to Wire length / Sections)
 Δt (pointing to Time intervals)
 - size array
 $x=1, x < \text{sections}-1$



Example Output:

```

2. ENGR
Re-attach Fullscreen Stay on top Duplicate
0 3.58786 7.97197 13.9748 22.1366 33.0024 46.5734 62.747 80.8305 100
0 3.95072 8.70962 14.9587 23.3688 34.2352 47.7594 63.6173 81.3254 100
0 4.31902 9.3887 15.9435 24.4882 35.4464 48.8229 64.4605 81.7659 100
0 4.66111 10.0655 16.8503 25.5881 36.5484 49.8533 65.2205 82.1891 100
0 4.99983 10.6946 17.7403 26.6009 37.6168 50.7931 65.9503 82.573 100
0 5.31651 11.3102 18.5674 27.5831 38.6013 51.6958 66.6181 82.9395 100
0 5.62512 11.886 19.3688 28.4957 39.5475 52.5288 67.2557 83.2763 100
0 5.91483 12.4428 20.118 29.3729 40.4268 53.3243 67.8453 83.5967 100
0 6.19426 12.9656 20.8379 30.1927 41.2669 54.0641 68.406 83.8938 100
0 6.45725 13.4673 21.5135 30.9763 42.0521 54.7681 68.9282 84.1756 100
0 6.70918 13.9395 22.1591 31.7114 42.7995 55.4262 69.4237 84.4385 100
0 6.94666 14.3903 22.7664 32.4113 43.5006 56.0509 69.8873 84.6876 100
0 7.17314 14.8152 23.3446 33.0695 44.1664 56.637 70.3266 84.921 100
0 7.38685 15.2196 23.8894 33.6947 44.7924 57.1925 70.7389 85.1418 100
0 7.59004 15.6011 24.4069 34.2837 45.3859 57.7149 71.1292 85.3493 100
0 7.78189 15.9633 24.8949 34.8421 45.945 58.2095 71.4964 85.5455 100
0 7.96393 16.3052 25.3577 35.3688 46.4744 58.6754 71.8438 85.7303 100
0 8.13588 16.6293 25.7945 35.8676 46.9736 59.1162 72.171 85.9049 100
0 8.29883 16.9355 26.2082 36.3383 47.446 59.5319 72.4805 86.0695 100
0 8.45277 17.2254 26.5989 36.7838 47.8918 59.925 72.7723 86.2251 100
0 8.59851 17.4992 26.9687 37.2045 48.3134 60.296 73.0483 86.3719 100
0 8.73623 17.7584 27.3179 37.6024 48.7116 60.6468 73.3087 86.5107 100
0 8.86653 18.0033 27.6483 37.9782 49.088 60.9779 73.5548 86.6416 100
0 8.98967 18.2349 27.9604 38.3336 49.4435 61.291 73.7872 86.7654 100
0 9.10613 18.4538 28.2556 38.6693 49.7796 61.5866 74.0068 86.8822 100
0 9.21619 18.6607 28.5344 38.9867 50.0971 61.8661 74.2143 86.9927 100
0 9.32025 18.8563 28.7981 39.2866 50.3972 62.13 74.4103 87.097 100
0 9.4186 19.0412 29.0473 39.5701 50.6808 62.3795 74.5955 87.1956 100
0 9.51157 19.2159 29.2828 39.838 50.9488 62.6152 74.7705 87.2887 100
0 9.59943 19.3811 29.5053 40.0912 51.202 62.838 74.9359 87.3767 100
0 9.68249 19.5372 29.7157 40.3304 51.4413 63.0485 75.0922 87.4599 100
0 9.76098 19.6848 29.9145 40.5565 51.6675 63.2474 75.2398 87.5385 100
flip1 ~/cs161/private/new 212%

```

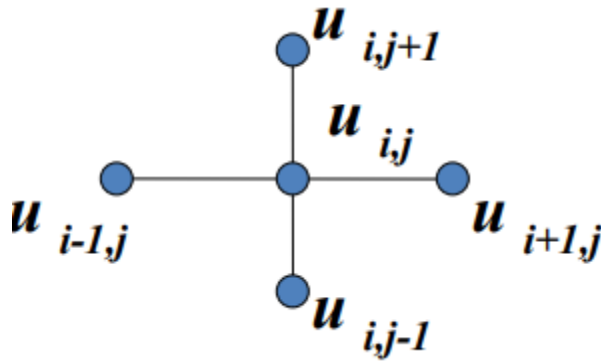
Next, you can implement the two-dimensional model:

$$k \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = c \rho \frac{\partial u}{\partial t}$$

$$\Downarrow$$

$$k \left(\frac{u^t_{i+1,j} - 2u^t_{i,j} + u^t_{i-1,j}}{(\Delta x)^2} + \frac{u^t_{i,j+1} - 2u^t_{i,j} + u^t_{i,j-1}}{(\Delta y)^2} \right) = c \rho \frac{u^{t+\Delta t}_{i,j} - u^t_{i,j}}{\Delta t}$$

Here, each point in your material is influenced by the points above and beside it.



Lastly, we want to make sure that our simulation is stable. Use this equation:

$$\Delta t \leq \frac{\Delta x * \Delta y * \rho * C}{4 * K}$$

Use the properties of nickel again:

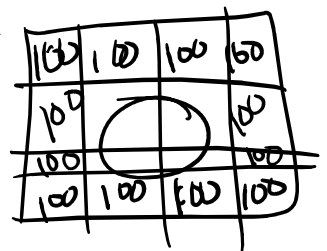
k	52.4
ρ (rho)	0.321
c	0.12
Plate height and width	10
Sections across height/width	10
Time intervals	50
Time of simulation	0.001675
Left/Right column temp	100
Bottom/Top row temp	100
Initial temp	0

Example Output:

```

2. ENGR
Re-attach Fullscreen Stay on top Duplicate
100 100 100 100 100 100 100 100 100 100
100 86.5792 75.9629 69.3441 66.3699 66.3699 69.3441 75.9629 86.5792 100
100 75.9629 56.957 45.1169 39.8006 39.8006 45.1169 56.957 75.9629 100
100 69.3441 45.1169 30.0369 23.2718 23.2718 30.0369 45.1169 69.3441 100
100 66.3699 39.8006 23.2718 15.861 15.861 23.2718 39.8006 66.3699 100
100 66.3699 39.8006 23.2718 15.861 15.861 23.2718 39.8006 66.3699 100
100 69.3441 45.1169 30.0369 23.2718 23.2718 30.0369 45.1169 69.3441 100
100 75.9629 56.957 45.1169 39.8006 39.8006 45.1169 56.957 75.9629 100
100 86.5792 75.9629 69.3441 66.3699 66.3699 69.3441 75.9629 86.5792 100
100 100 100 100 100 100 100 100 100 100
100 100 100 100 100 100 100 100 100 100
100 86.8348 76.3743 69.8032 66.8272 66.8272 69.8032 76.3743 86.8348 100
100 76.3743 57.6101 45.8311 40.5005 40.5005 45.8311 57.6101 76.3743 100
100 69.8032 45.8311 30.7947 23.9956 23.9956 30.7947 45.8311 69.8032 100
100 66.8272 40.5005 23.9956 16.5364 16.5364 23.9956 40.5005 66.8272 100
100 66.8272 40.5005 23.9956 16.5364 16.5364 23.9956 40.5005 66.8272 100
100 69.8032 45.8311 30.7947 23.9956 23.9956 30.7947 45.8311 69.8032 100
100 76.3743 57.6101 45.8311 40.5005 40.5005 45.8311 57.6101 76.3743 100
100 86.8348 76.3743 69.8032 66.8272 66.8272 69.8032 76.3743 86.8348 100
100 100 100 100 100 100 100 100 100 100
flip3 ~/csl61/private/new 173%

```



You can look up other metals and their properties to run more simulations.
https://www.engineersedge.com/properties_of_metals.htm

Design Document – Due Sunday 3/10/19, 11:59pm
Refer to the Example Polya Document - [Polya template.pdf](#)

Step 1: Understanding the Problem/Problem Analysis. (15 pts)

Do you understand everything in the problem? List anything you do not fully understand, and make sure to ask a TA or instructor about anything you do not understand.

- What are the user inputs/requirements, program outputs, etc.? (5 pts)
- What assumptions are you are making? (5 pts)
- What are all the tasks and subtasks in this problem? (5 pts)

Step 2: Program Design. (25 pts)

- What does the overall big picture of this program look like? (flowchart or pseudocode) (15 pts)
 - What data do you need to create, when do you read input from the user, what is the dealer strategy going to be?
 - What are the decisions that need to be made in this program?
 - What tasks are repeated?
- What kind of bad input are you going to handle? (5 pts)
- Provide a drawing/diagram of what the initial plate and wire will look like (5 pts).

Based on your answers above, list the **specific steps or provide a flowchart** of what is needed to create this program.

Step 4: Program Testing. (10 pts)

Create a test plan with the test cases (bad, good, and edge cases). What do you hope to be the expected results?

- What are the good, bad, and edge cases for ALL input in the program? Make sure to provide enough of each and for all different inputs you get from the user.

Program Code – Due Sunday 3/17/19, 11:59pm

(90 pts) Write the program to simulate heat diffusion.

Program Input/Output

- Prompt the user to simulate heat diffusion of over a plate or wire.
- Allow the user to modify ρ , k , and c .
- Ask the user for the wire/plate length and segments/create 1-d or 2-d dynamic array of correct dimensions
- Ask the user for the time length and intervals
- Ask the user for the initial temp, and right and left (in addition to top and bottom if plate) constant temperature
- Check if the parameters are stable, if it is not stable, you must alert the user.
- Print the calculated data at each time interval to the screen
- Handle all bad input from the user, even data of the wrong type
- Re-prompt the user to see if they want to go again.

- Make sure you follow the style guidelines, have a program and function headers with appropriate comments, and be consistent.
- Automatic Deductions: function over 15 lines, use of globals, use of gotos, and memory leaks.

(10 pts) Extra Credit

Allow the user to enter command-line arguments for wire or plate, rho, k, and c using the following options to designate the information, **a.out -u wire -r 0.321 -k 52.4 -c .12**. These option value pairs should come in any order, and you should handle any errors with these.

Visualizing the data

Type the following commands in the directory where your code is.

```
$ curl -S http://classes.engr.oregonstate.edu/eecs/winter2019/cs161-001/heat_diffusion/setup.sh > setup.sh
$ chmod a+x setup.sh
$ ./setup.sh
$ export GADDIR=/nfs/stak/faculty/p/parhammj/grads-2.0.1/lib
OR
$ setenv GADDIR /nfs/stak/faculty/p/parhammj/grads-2.0.1/lib
```

Note: You will need to run the final **export** or **setenv** command every time you start a new session.

After running these commands, you should see the following files in the directory:

```
grads
heatdiff.ctl
heatdiff2.ctl
1d_grads_script.gs
2d_grads_script.gs
helper.hpp
```

There is one final step before you can visualize your simulation. GrADS requires binary data from a file. To store your data in a file, you will use the helper.hpp file.

In your program, add the following line at the top of your program with the other libraries:

```
#include "helper.hpp"
```

Add this line of code to the beginning of the function where you are writing your 1-d and 2-d arrays to the screen:

```
fileWriter fw("heat.dat");
```

Finally, add this line of text where you are printing your 1-D data to the screen.

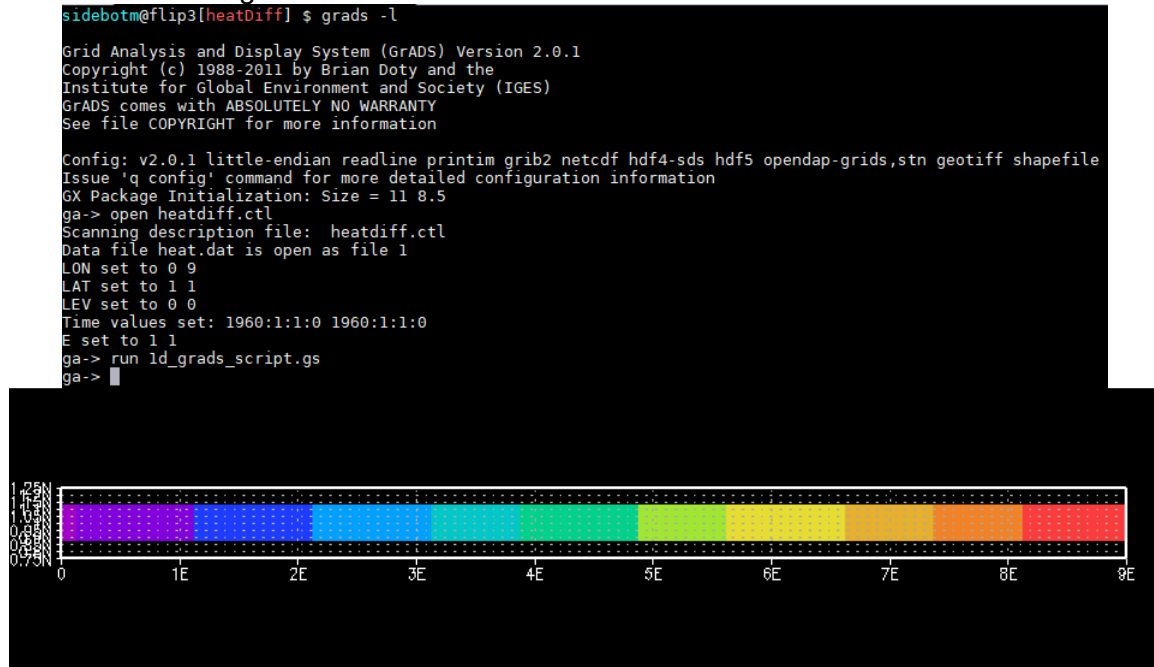
```
fw << data[i];
```

Once you run your program, you should see a file named **heat.dat** in your local directory.

Now you are ready to use GrADS to visualize your data. To do so, follow these commands.

```
$ grads -l
ga-> open heatdiff.ctl
ga-> run 1d_grads_script.gs
```

You should see something like this:



To exit GrADS use:

```
ga-> quit
```

Use the same method to output to the binary file, instead use **heat2.dat**.

```
fileWriter fw("heat2.dat");
...
fw << data[i][j];
```

Now we can visualize our data with GrADS for the two-dimensional simulations.

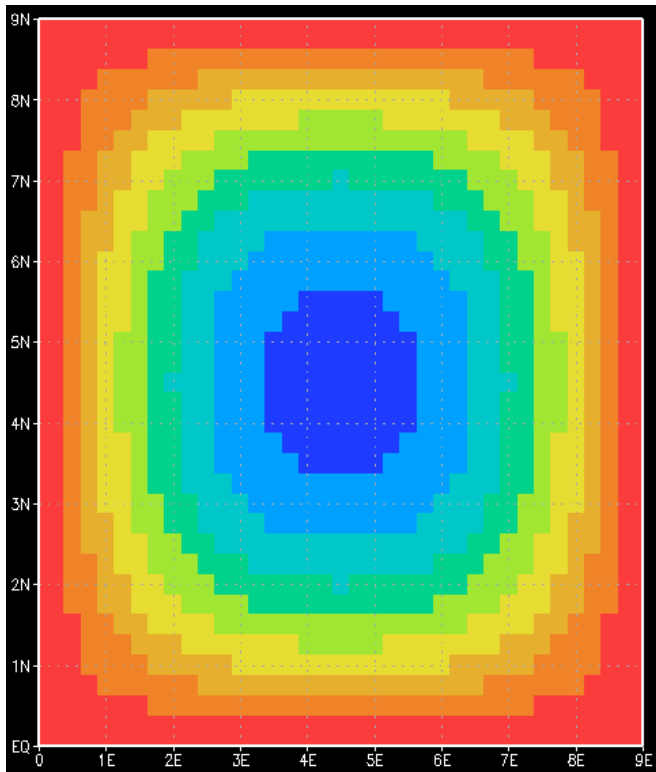
```
$ grads -l
ga-> open heatdiff2.ctl
ga-> run 2d_grads_script.gs
```

You should see something like this.

```
sidebotm@flip3[heatDiff] $ grads -l

Grid Analysis and Display System (GrADS) Version 2.0.1
Copyright (c) 1988-2011 by Brian Doty and the
Institute for Global Environment and Society (IGES)
GrADS comes with ABSOLUTELY NO WARRANTY
See file COPYRIGHT for more information

Config: v2.0.1 little-endian readline printim grib2 netcdf hdf4-sds hdf5.opendap-grids, stn geotiff shapefile
Issue 'q config' command for more detailed configuration information
GX Package Initialization: Size = 11 8.5
ga-> open heatdiff2.ctl
Scanning description file: heatdiff2.ctl
Data file heat2.dat is open as file 1
LON set to 0 9
LAT set to 0 9
LEV set to 0 0
Time values set: 1960:1:1:0 1960:1:1:0
E set to 1 1
ga-> run 2d_grads_script.gs
ga->
```



Electronically submit your Design Document by the design due date and your C++ program (.cpp file, not your executable!!!) by the code due date using Peerceptiv.