

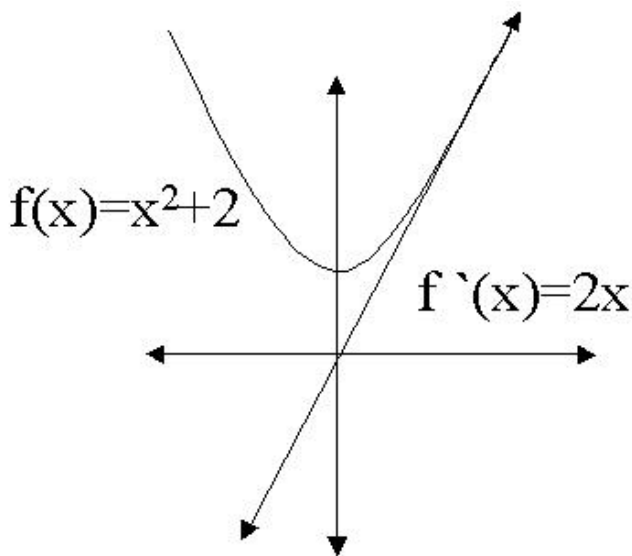
ASSIGNMENT #9

Heat Diffusion Using the Explicit Method

DUE - 11/26/17, 11:59pm

You will practice writing a **Python program** and gain an understanding of the 1-D and 2-D heat diffusion model, the explicit method for solving finite difference approximations, redirecting output to a file, and visualization of data. 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.

Background Information:

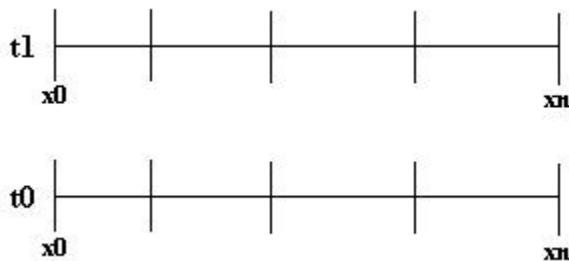


I. How do we do this? Finite Difference Approximation to Solve Derivatives

II. Function of Space and Time - $u(x, t)$

- Boundary conditions - $u(x_0, t) = \text{boundary1}$, $u(x_n, t) = \text{boundary2}$
- Initial conditions - $u(x_i, 0) = \text{initial value}$, $i = 1..n-1$

III. What exactly do we mean by 1-D heat diffusion on a wire?



- leave time continuous
- divide on space/finite intervals

IV. Finite Difference Approximation, FDA, for Derivatives

- $f'(x) \approx (f(x+h) - f(x)) / h$
- $f''(x) \approx (f(x+h) - 2 * f(x) + f(x-h)) / h * h$
- So what is this h thing?
- the change in the variable you are taking the derivative w/ respect to

VI. What is the heat diffusion equation?

- Conservation of mass, energy, momentum, etc.
- Rate of flow in - Rate of flow out = Rate of heat storage

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

Don't freak out!!! We don't have to do derivatives to write a program for it ☺

Let's use the explicit method for finite difference approximation (FDA) to solve the equation...

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

What are we interested in? $u_x^{t+\Delta t}$

Writing Your Program:

1. What are the inputs/outputs to the heat diffusion problem?

- Material Parameters: thermal conductivity (k), density (ρ), specific heat (c)
- Initial and Boundary conditions
- Material Length and how to divide length
- Number of Timer Intervals and change in time
- Time instance and values of all elements of 1-D or 2-D object at that time instance
- $|(k * \text{change in time}) / (\text{change in } x^2 * c * \rho)| < .5$ for stability

2. Pseudocode for Explicit heat diffusion

For all time instances

For all points on material (u)

predict new value at new time instance (equation)

End For

End For

3. Program Input/Output

- Read whether the object is 1-D or 2-D from command-line arguments, catch bad input
- Read the information from a file, make sure to handle anything that isn't good (including an unstable condition for calculation $< .5$)
- Print the time interval and values of the wire at each time instance to the screen
- Print the binary information to a file for grads visualization (instructions below)

4. Example Input w/ Thermal Conductivity, Density, and Specific Heat for Nickel (you can look up other metals here: http://www.engineersedge.com/properties_of_metals.htm).

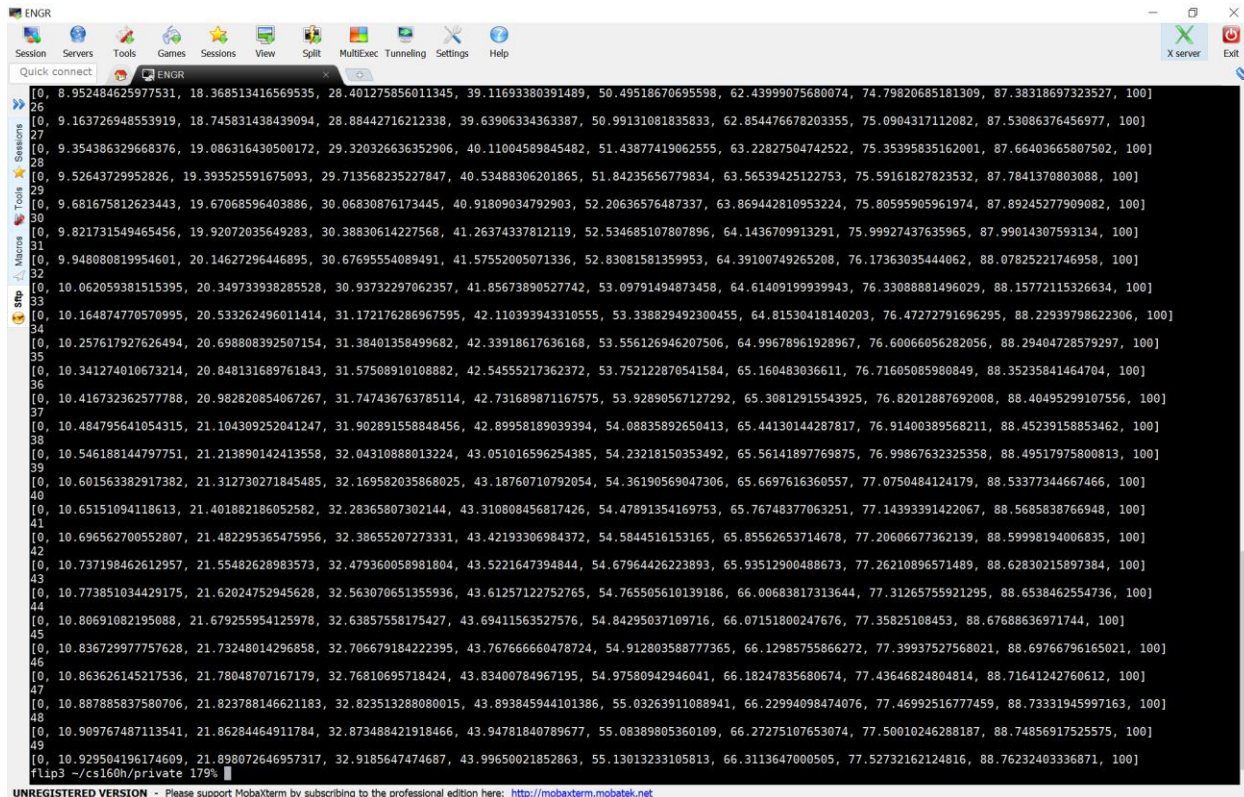
52.4 0.321 0.12 // Thermal conductivity (k), density (ρ), specific heat (c)

0.0 0.0 100.0 //Initial temp, and boundary conditions (left and right)

10 10 //Length of wire and sections

50 .000335 //Time intervals and change in time

Output:



```
[0, 8.952484625977531, 18.368513416569535, 28.401275856011345, 39.11693380391489, 50.49518670695598, 62.43999075680074, 74.79820685181309, 87.38318697323527, 100]
26
[0, 9.163726948553919, 18.745831438439094, 28.88442716212338, 39.63906334363387, 50.99131081835833, 62.854476678203355, 75.0904317112082, 87.53086376456977, 100]
27
[0, 9.354386329668376, 19.086316430500172, 29.320326636352906, 40.11004589845482, 51.43877419062555, 63.22827504742522, 75.35395835162001, 87.66403665807502, 100]
28
[0, 9.52643729952826, 19.393525591675093, 29.713568235227847, 40.53488306201865, 51.84235656779834, 63.56539425122753, 75.59161827823532, 87.7841370803088, 100]
29
[0, 9.681675812623443, 19.67068596403886, 30.06830876173445, 40.91809034792903, 52.20636576487337, 63.869442810953224, 75.80595905961974, 87.89245277909082, 100]
30
[0, 9.821731549465456, 19.92072035649283, 30.38830614227568, 41.26374337812119, 52.534685107807896, 64.1436709913291, 75.99927437635965, 87.99014307593134, 100]
31
[0, 9.948080819954601, 20.14627296446895, 30.67695554089491, 41.57552005071336, 52.83081581359953, 64.39100749265208, 76.17363035444062, 88.07825221746958, 100]
32
[0, 0.062059381515395, 20.349733938285528, 30.93732297062357, 41.85673890527242, 53.09791494873458, 64.61409199939943, 76.33088881496029, 88.15772115326634, 100]
33
[0, 0.164874770570995, 20.533262496011414, 31.172176286967595, 42.110393943310555, 53.338829492300455, 64.81530418140203, 76.47272791696295, 88.22939798622306, 100]
34
[0, 0.257617927626494, 20.698808392507154, 31.38401358499682, 42.33918617636168, 53.556126946207506, 64.99678961928967, 76.60066056282056, 88.29404728579297, 100]
35
[0, 0.341274010673214, 20.848131689761843, 31.57508910108882, 42.5455217362372, 53.752122870541584, 65.160483036611, 76.71605085980849, 88.35235841464704, 100]
36
[0, 0.416732362577788, 20.982820854067267, 31.747436763785114, 42.731689871167575, 53.92890567127292, 65.30812915543925, 76.82012887692008, 88.40495299107556, 100]
37
[0, 0.484795641054315, 21.104309252041247, 31.902891558848456, 42.89958189039394, 54.08835892650413, 65.44130144287817, 76.91400389568211, 88.45239158853462, 100]
38
[0, 0.546188144797751, 21.213890142413558, 32.04310888013224, 43.051016596254385, 54.23218150353492, 65.56141897769875, 76.99867632325358, 88.49517975800813, 100]
39
[0, 0.601563382917382, 21.312730271845485, 32.169582035868025, 43.18760710792054, 54.36190569047306, 65.6697616360557, 77.0750484124179, 88.53377344667466, 100]
40
[0, 0.65151094118613, 21.401882186052582, 32.28365807302144, 43.310808456817426, 54.47891354169753, 65.76748377063251, 77.14393391422067, 88.5685838766948, 100]
41
[0, 0.696562700552807, 21.482295365475956, 32.38655207273331, 43.42193306984372, 54.5844516153165, 65.85562653714678, 77.20606677362139, 88.59998194006835, 100]
42
[0, 0.737198462612957, 21.55482628983573, 32.479360058981804, 43.5221647394844, 54.67964426223893, 65.93512900488673, 77.26210896571489, 88.62830215897384, 100]
43
[0, 0.773851034429175, 21.62024752945628, 32.563070651355936, 43.61257122752765, 54.765505610139186, 66.00683817313644, 77.31265755921295, 88.6538462554736, 100]
44
[0, 0.80691082195088, 21.679255954125978, 32.63857558175427, 43.69411563527576, 54.84295037109716, 66.07151800247676, 77.35825108453, 88.67688636971744, 100]
45
[0, 0.83672997757628, 21.73248014296858, 32.706679184222395, 43.76766660478724, 54.91280358877365, 66.12985755866272, 77.39937527568021, 88.69766796165021, 100]
46
[0, 0.863626145217536, 21.78048707167179, 32.76810695718424, 43.83400784967195, 54.97580942946041, 66.18247835680674, 77.43646824804814, 88.71641242760612, 100]
47
[0, 0.887885837580706, 21.823788146621183, 32.823513288080015, 43.893845944101386, 55.03263911088941, 66.22994098474076, 77.46992516777459, 88.73331945997163, 100]
48
[0, 0.909767487113541, 21.86284464911784, 32.873488421918466, 43.94781840789677, 55.08389805360109, 66.27275107653074, 77.50010246288187, 88.74856917525575, 100]
49
[0, 0.929504196174609, 21.898072646957317, 32.9185647474687, 43.99650021852863, 55.13013233105813, 66.3113647000505, 77.52732162124816, 88.76232403336871, 100]
fltp3 ~/cs160h/private 179%
```

Since GrADS only visualizes binary data, go back to your python program and add then information need to print to a binary file:

Import the struct library for packing binary data

```
import struct
```

Open the file for writing in binary

```
f=open("heat.dat", 'wb')
```

Write the calculated u for the new time instance

```
f.write(struct.pack('f', u_new[i]))
```

Once you have your file, you can use GrADS to visualize your data. Read more about GrADS Visualization package, <http://cola.gmu.edu/grads/gadoc/gadoc.php>, and GrADS Gridded Data Sets, <http://cola.gmu.edu/grads/gadoc/aboutgriddeddata.html>.

I have this setup in my directory on the ENGR server. Therefore, in order to see things displayed on the ENGR server, you have to use MobaXterm (Windows) or start Xquartz and ssh with -Y (MacOS).

Back to the ENGR server...

Download the grads control file, [heatdiff.ctl](#), and the grads script, [1d_grads_script.gs](#), that will open the binary data, heat.dat, and display the heat diffusion in grads. You can read the GrADS descriptor/control files for the data:

<http://cola.gmu.edu/grads/gadoc/aboutgriddeddata.html#descriptor>.

Set the GADDIR environment variable:

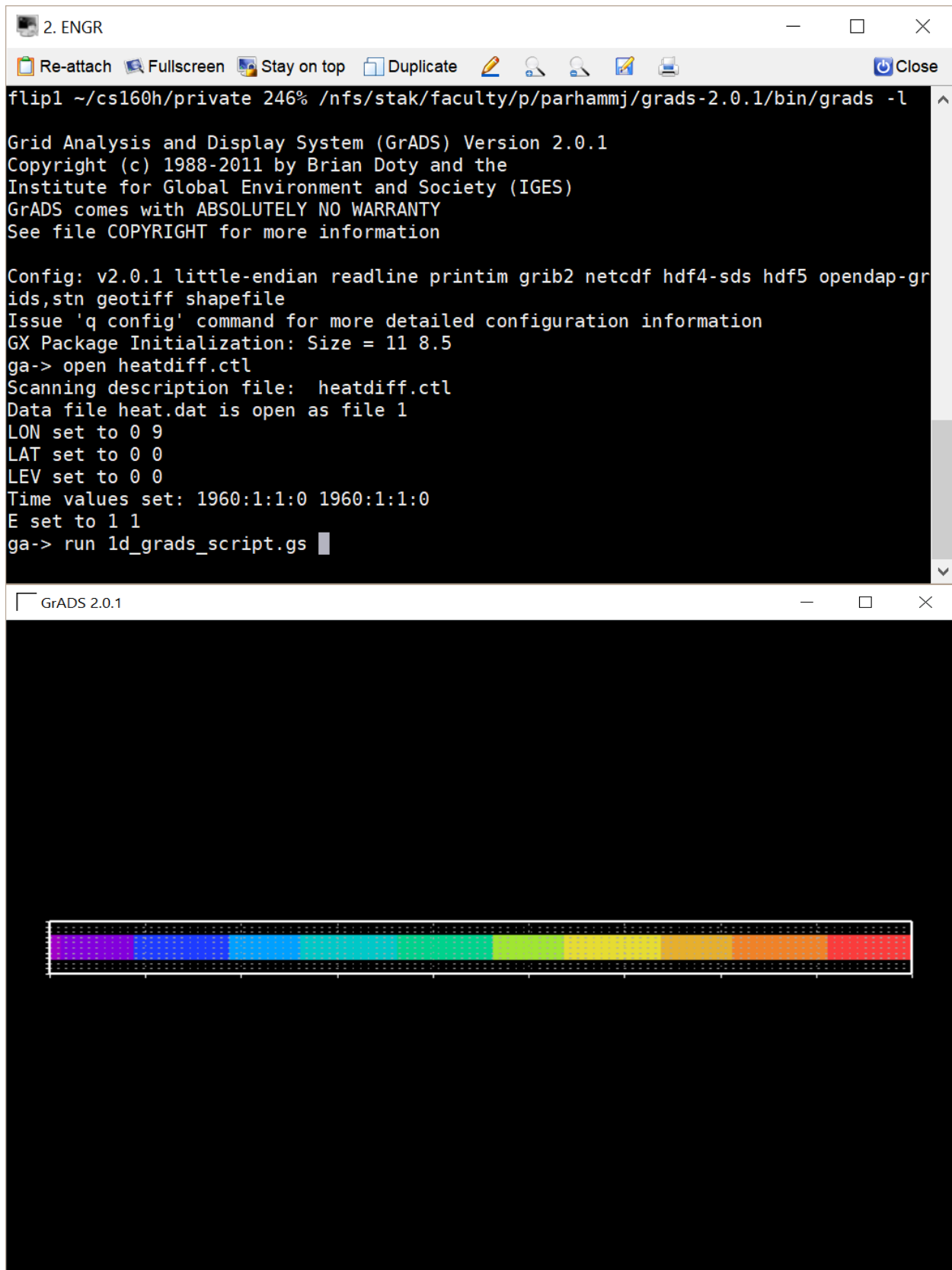
```
setenv GADDIR /nfs/stak/faculty/p/parhammj/grads-2.0.1/lib
```

Now, you can actually run the grads visualization tool

```
/nfs/stak/faculty/p/parhammj/grads-2.0.1/bin/grads -l
```

You should now get a prompt that looks like, ga->, and this is where you can **open** the **heatdiff.ctl** file and then **run** the **1d_grads_script.gs** files to make sure you did everything right. There is a picture below that shows these commands and the pictures you should get showing the heat diffusion. You should see heat diffuse down your wire, if your output is correct... **quit** will close grads!!!!

Below is the picture you should get:



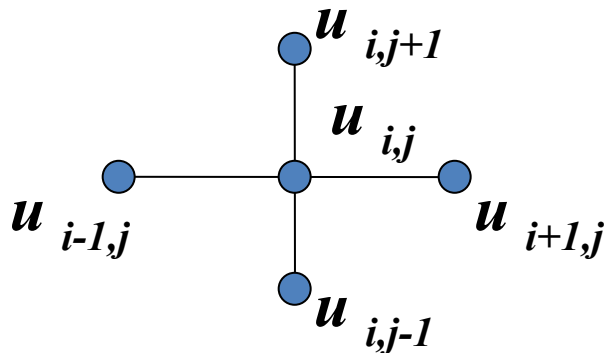
Now, Extend this to 2-D

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

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

$$\Downarrow$$

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



For our 2-D array, we will just make it square to make it easy, i.e. same number of sections in x and y direction!

Example Input w/ Thermal Conductivity, Density, and Specific Heat for Nickel (you can look up other metals here: http://www.engineersedge.com/properties_of_metals.htm).

```
52.4 0.321 0.12 // Thermal conductivity (k), density (ρ), specific heat (c)
0.0 100.0 100.0 //Initial temp, and boundary conditions (left and right)
100.0 100.0 //Boundary conditions (top and bottom)
10 10 //Length of wire and sections
50 .0000335 //Time intervals and change in time
```

Output:


```
4. ENGR
Re-attach Fullscreen Stay on top Duplicate Close
flip3 ~/cs160h/private 189% ~/grads-2.0.1/bin/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 openda
p-grids, stn geotiff shapefile
Issue 'q config' command for more detailed configuration information
GX Package Initialization: Size = 11 8.5
ga-> open heatdiff2.ct1
Scanning description file: heatdiff2.ct1
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
```

