This assignment is based on the material covered in these two classes:

[Tues Mar 30, 2021](https://web.microsoftstream.com/video/5fddd682-2033-4045-98bc-2216abad5f6d)

Went over aspects related to the the pthread matrix vector homework.  Specifically about reading the data in one big block, not in a doubly nested for loop (conversation with Gavin).  One large read, versus lots of small reads.  When over concepts to consider when debugging your code (verifying the data in the files, hexdump), looking not just at file sizes but also file contents. Thinking about anytime you're multiplying integers, if they are large, then overflow can occur, and need to consider this in mallocs(), large reads, writes, and even in the loop indices.  Be methodical, and  pay attention to the details.  The we went over the 9-point stencil concepts.  How my example data was generated, and how to visualize it using the Jupiter notebook I shared.  Then went over the parallelization of this code.  The need to exchange border information / ghost cells. [Ghost Cell Pattern -- article](http://fredrikbk.com/publications/ghost_cell_pattern.pdf).  Using MPI\_Sendrecv() to exchange information with neighbor cells.   [Updated slides from today](https://moodle.coastal.edu/pluginfile.php/1036785/mod_page/content/40/StencilCodes%20%281%29.pptx) [Accessibility score: Low Click to improve](https://moodle.coastal.edu/mod/page/view.php?id=744776)  .

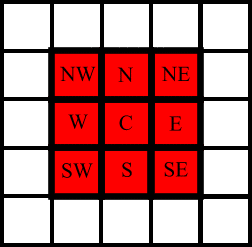
Friday Mar 26, 2021 -- asynchronous

I went over stencil project, showing example execution. [Here is an echo360 video of that](https://echo360.org/media/eec52d67-8016-4323-a26b-7738c2fb3f29/public). [Here is the example 9-point input data and some outputs](https://moodle.coastal.edu/pluginfile.php/1036785/mod_page/content/40/9-point.tar.gz) you can use to verify if your program is working.

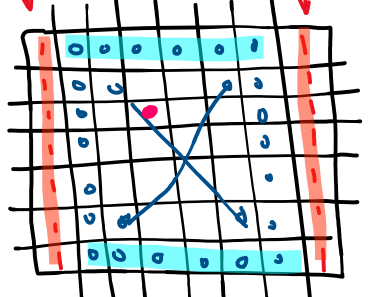
[Thurs Mar 25, 2021](https://web.microsoftstream.com/video/683eaa31-3756-49bc-847e-a571eeb6ac8f)

Started with stencil codes.   [Slides from today](https://moodle.coastal.edu/pluginfile.php/1036785/mod_page/content/40/StencilCodes.pptx) [Accessibility score: Low Click to improve](https://moodle.coastal.edu/mod/page/view.php?id=744776)  .  [Wikipedia](https://en.wikipedia.org/wiki/Iterative_Stencil_Loops).  Discussed where we were going with this, how the BLOCK\_macros could be used to partition matrix by rows, using pthreads in shared memory, so no explicit message passing.  Getting boundary conditions setup (the 0's and 1's), general loop structure (be careful of the "boundaries"), the outer time loop.  Also about ghost cells (that we will need them).  MPI\_Gather and \_Scatter will be used in the MPI version.  Alluded to hybrid programing (pthreads + MPI, or OpenMP + MPI, etc) and why such a thing is done.  Making copies of data to be shared around.

This is a 9-point stencil code, where the given C is updated to be the unweighted average of its 9 surrounding neighbors.



You will have an initial matrix generation program that will setup the initial conditions as described in class and a print utility as shown here:



*(initial conditions in the pate, 1.0 on the left and right walls, and 0.0 on the top and bottom walls, and 0.0 internally. Note, the boundary conditions do not change over the time iterations)*

(wjones-venv) % ./make-2d

usage: ./make-2d -r <rows> -c <cols> -o <output\_file>

(wjones-venv) %

(wjones-venv) % ./make-2d -r 4 -c 5 -o initial.dat

(note, the data file has the same format and metadata as used in the matrix vector project, i.e. two ints, followed by the double data).

(wjones-venv) % ls -l initial.dat

-rw-r--r-- 1 wjones 1334764962 168 Mar 31 11:22 initial.dat

(wjones-venv) %

(wjones-venv) % ./print-2d

usage: ./print-2d -i <input data file>

(wjones-venv) %

(wjones-venv) % ./print-2d -i initial.dat

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

(wjones-venv) %

Then there is the program itself:

(wjones-venv) % ./stencil-2d

usage: ./stencil-2d -n <num iter.> -i <infile> -o <outfile>

(wjones-venv) %

(wjones-venv) % ./stencil-2d -n 3 -i ./initial.dat -o final.dat

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.33 0.00 0.33 1.00

1.00 0.33 0.00 0.33 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.41 0.15 0.41 1.00

1.00 0.41 0.15 0.41 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.00 0.00 0.00 1.00

1.00 0.46 0.21 0.46 1.00

1.00 0.46 0.21 0.46 1.00

1.00 0.00 0.00 0.00 1.00

(wjones-venv) %

The first block, is the intial state, and then each block after that is the state of the plate after each of the 3 iterations.

(wjones-venv) % ls -l final.dat

-rw-r--r-- 1 wjones 1334764962 168 Mar 31 11:28 final.dat

(wjones-venv) %

As you can see, the final state is indeed in the final.dat file:

(wjones-venv) % ./print-2d -i final.dat

1.00 0.00 0.00 0.00 1.00

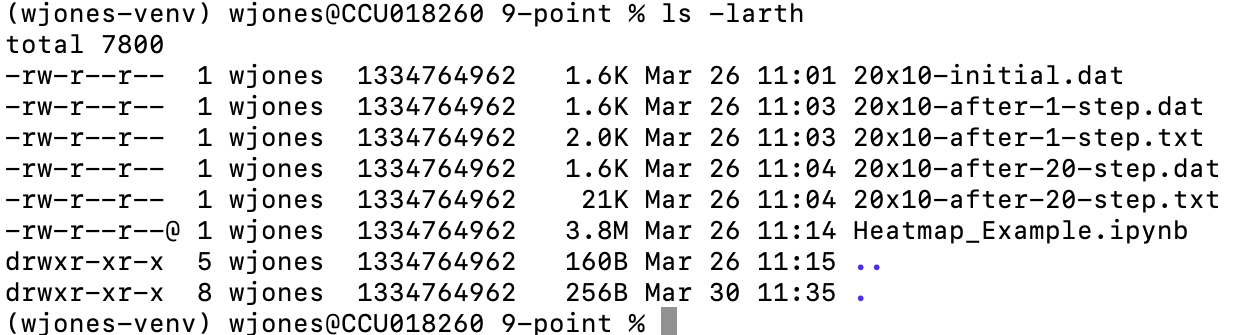
1.00 0.46 0.21 0.46 1.00

1.00 0.46 0.21 0.46 1.00

1.00 0.00 0.00 0.00 1.00

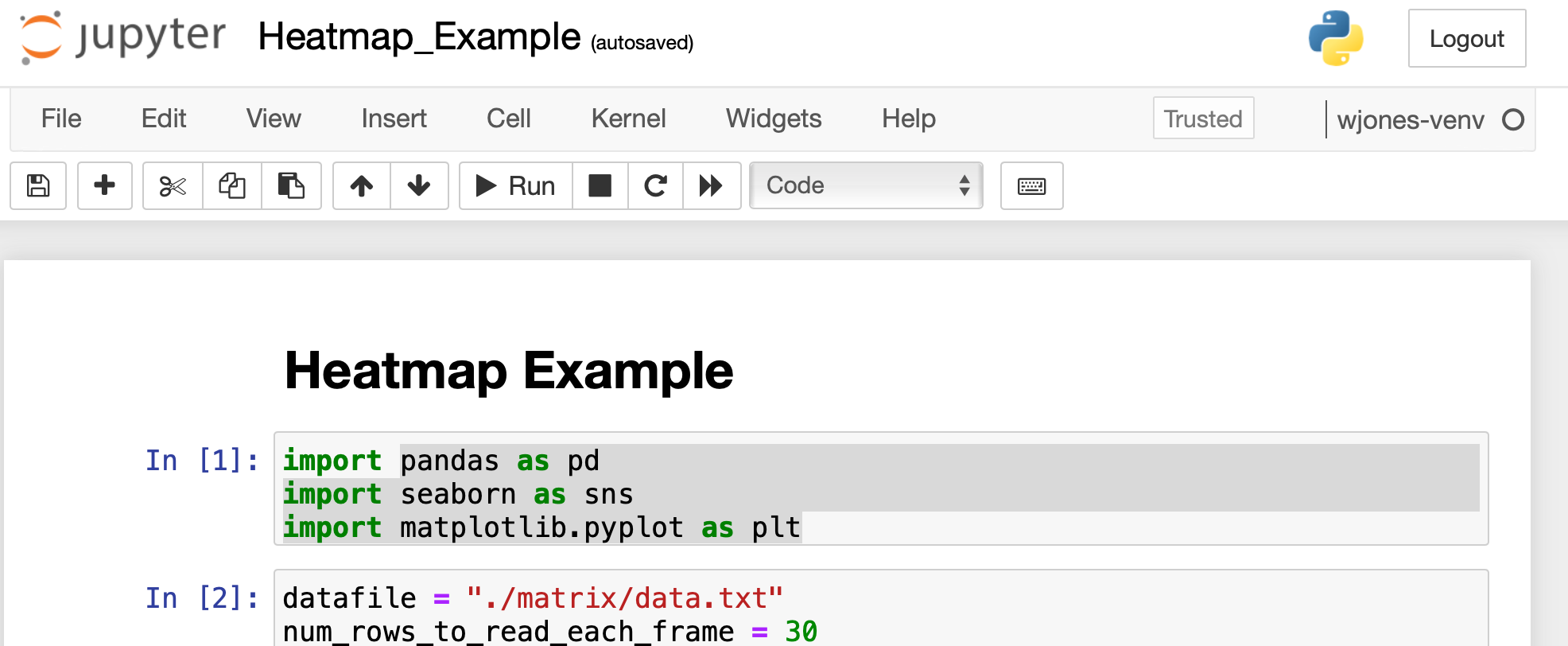
(wjones-venv) %

Note, in the archive I posted in the assignment, there are these files:

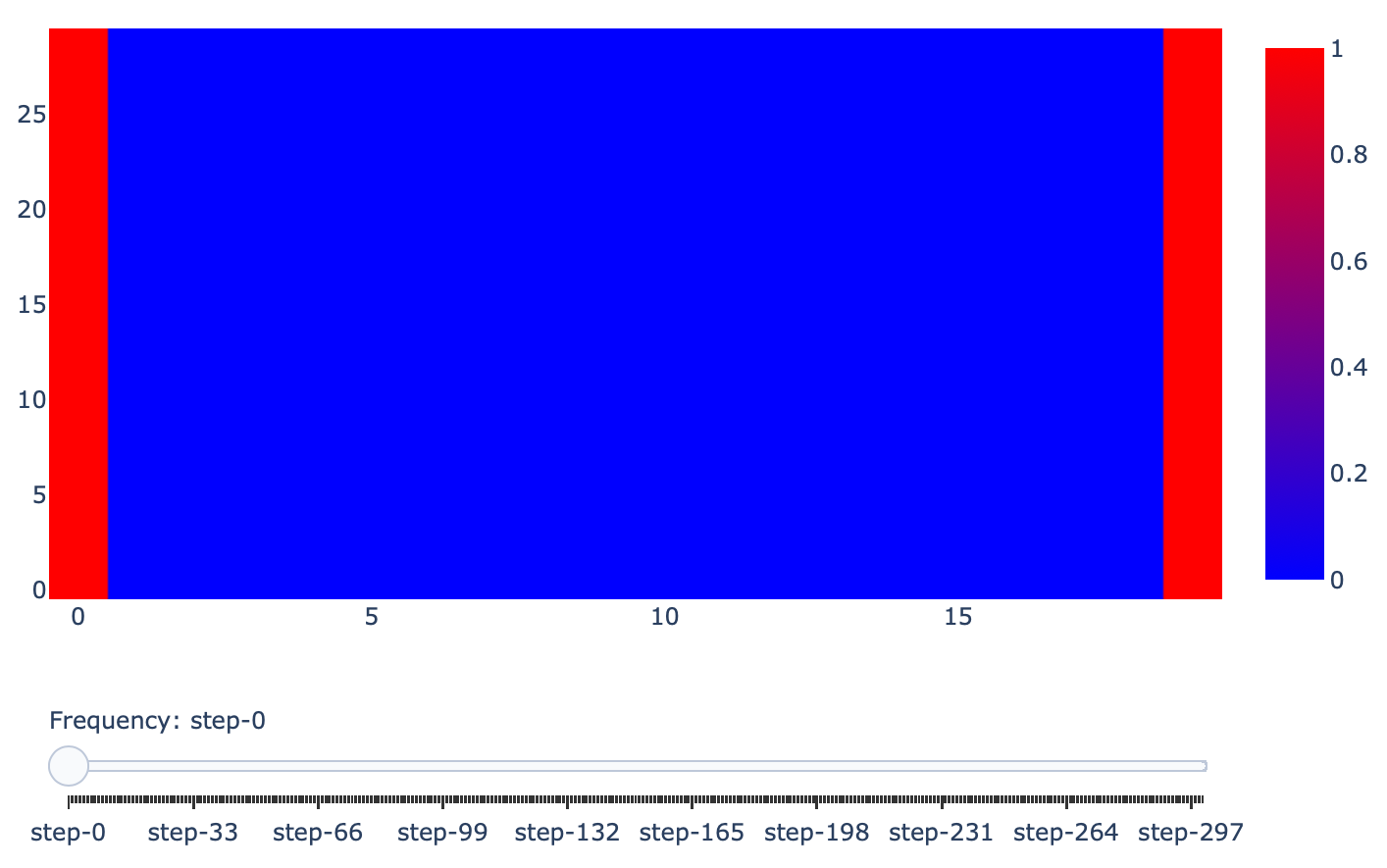


This will give you some initial data you can start with, to make sure that your own programs are working correctly. In other words, if you generate a 20x10 initial matrix, it should be identical to the one above. Also, you use that as input to your serial stencil program, you should be able to compare your program’s output after 1 iteration to the one above (1-step) and likewise after 20 steps, to make sure that they are the same.

Finally, there is a Jupyter notebook there that can take as input the output from running your code (the text output from the screen, which I obtained from redirecting. (look at the .txt file above). That can be feed into the notebook to visualize the data.

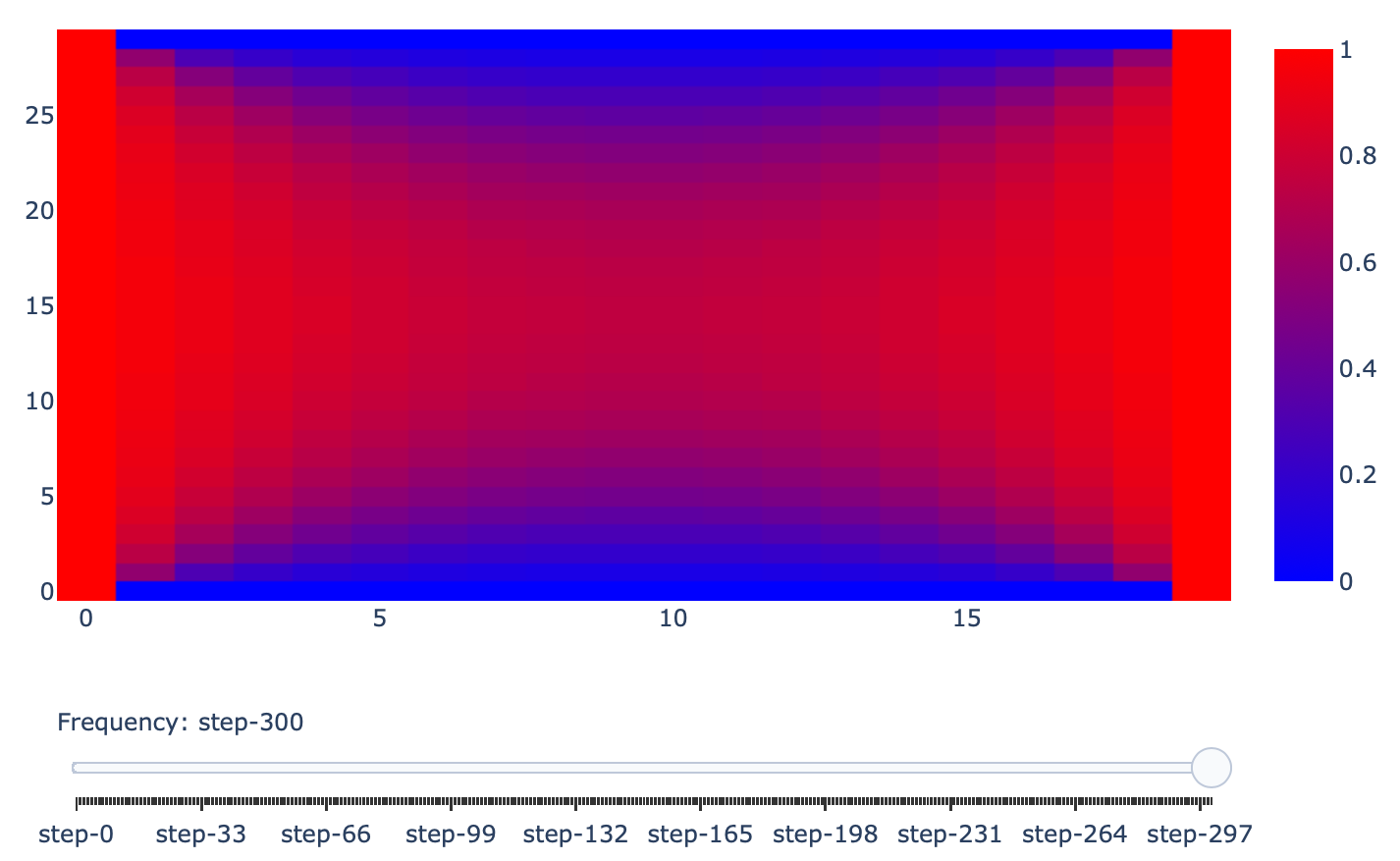


Keeep in mind that the last line there in the screenshot has to be set to the number of rows for your matrix (is hard coded).



*initial conditions*

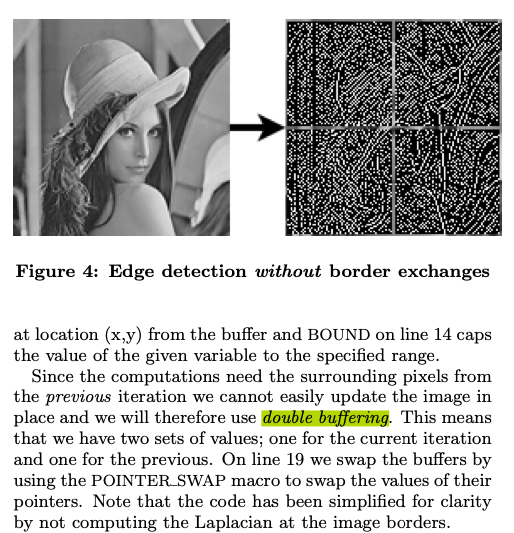
and then, after 300 time steps:



Remember, the stencil update for a give cell **IS NOT DONE IN PLACE**. You actually have **TWO** matrices in memory, and you alternate between them. That is referred to double buffering.

You can see it discussed as a concept here:

<http://fredrikbk.com/publications/ghost_cell_pattern.pdf>



*(author mentions double buffering)*

main concept:

A ß 2D matrix in Memory

B ß 2D matrix in Memory

Matrix A – initialized to initial boundary and internal conditions

Matrix B – also initialized to initial boundary and internal conditions

for(all time iterations)

for(the correct rows)

for(the correct cols)

place into B, stencil computation based on A

swap pointers for A and B

**Submission:**

Your project WILL contain:

Makefile

make-2d.c

print-2d.c

stencil-2d.c

and may also have for any functions you use across programs

function.c

functions.h

You will also include any documentation, observations, and screenshots of your code working, here below this, and submit the WORD document into Moodle (NOT A PDF).

Tar GZ and submit into moodle.

REPORT BELOW HERE

In completing this assignment, I modified the gen-matrix program from creating a matrix with random values between given bounds to create a matrix with the first and last columns set to 1.0 and the rest of the values set to 0. For the print-2 program I reused the print-matrix program we have been using. For the stencil-2d program I first made sure I was able to read in the matrix file and print the initial values. After that I got the program to create a second copy of the initial matrix and worked on swapping the pointers, which I did by creating a temporary double\*\* variable to store the pointer for one of the matrices while swapping. Finally, I completed the program by implementing the math to do the 9-point average for each element of the matrix and then swapping the pointers after all of the new elements had been calculated, in order to perform the next iteration.

