Michigan Tech

UN5390: Scientific Computing I

Fall 2016

Assignment #07 (Due 11:59 am on Sunday, 6th Nov 2016; 15%)

Problem 1

Each year, second Tuesday of October is celebrated as *Ada Lovelace Day* (ALD) in honor of her contributions and accomplishments. Write a four paragraph summary (each with 4-6 short but meaningful sentences), not exceeding one single-sided page, covering the following aspects: her personal life, her association with Charles Babbage, and her contributions to computing, technology and humanity. The last paragraph may be devoted to finding out why ALD is celebrated on second Tuesday of October when it's neither her birthday nor the day she died.

Ada's notes to the translation of Menabrea's memoirs [1, 2] terminated with a step-by-step description of how *the engine could compute the Numbers of Bernoulli, this being (in the form in which we shall deduce it) a rather complicated example of its powers.* This *algorithm* is often labeled the world's first computer program. Write an appropriately named and well-commented function (not a program) that generates Bernoulli Numbers in her honor.

Problem 2

VO_2Max is defined as the maximum oxygen consumption capacity $[mL/(kg \cdot min)$; milliliter of oxygen consumed per minute per kilogram of body weight] during an intense or a maximal exercise. While there are many techniques to estimate this entity, the Daniels and Gilbert formula [3, 4] is given by

$$VO_2\text{Max} = \frac{\text{Oxygen cost}}{\text{Drop dead}} = \frac{0.000104\,v^2 + 0.182258\,v - 4.60}{0.2989558\,e^{-0.1932605t} + 0.1894393\,e^{-0.012778t} + 0.80}$$

d is the distance in meters, t is the time in minutes, and v is the speed in meters per minute. Write a well-commented program, `vo2max.EXTN`, that predicts time (in human readable format, `h:mm:ss`) given the distance (in miles) and VO_2Max. Include necessary mathematical simplifications and/or thought process underlying your approach. Complete the table below.

d (miles)	t (`h:mm:ss`)	VO_2Max	t_c (`h:mm:ss`)	$\epsilon = \lvert t - t_c \rvert$
3.10	0:22:07			
6.20	0:47:47			
13.10	1:43:02			
26.20	4:06:21			

First, compute VO_2Max for each combination of d and t. Then, use the same d and computed VO_2Max to predict the time, t_c. Does the program reproduce the result in each case? What's the error?

Problem 3

Suppose that the temperature of a well stirred liquid does not depend on space. Further suppose that ΔT is the difference between room temperature and the temperature of the liquid at any given time. It could be observed that for small increments of time, Δt, the change in temperature is approximately equal to the product of some positive constant k (material properties of the liquid's container), Δt and ΔT. This observation forms the basis of *Newton's Law of Cooling*, and is expressed as the following differential equation.

$$\frac{dT}{dt} = k\ (T_{\text{room}} - T)$$

Suppose that T_n is the temperature of the liquid at time t_n and T_{room} is the room temperature. Then, in its discrete form, the law can be represented as

$$T_{n+1} - T_n = k\ (T_{\text{room}} - T_n)\ \Delta t$$

Re-arranging the terms, the above expression can be written as

$$T_{n+1} = \alpha\, T_n + \beta \quad \text{where} \quad \alpha = 1 - k\,\Delta t \quad \text{and} \quad \beta = k\,\Delta t\, T_{\text{room}}$$

Write an aptly named, well-commented and modularized program that simulates the cooling of this liquid over time. Test the program for $T_{\text{room}} = 70^{\circ}F$ and $\Delta t = 1$. What will be the eventual temperature of the liquid? Experimental data presented in Table 1 can be used to estimate k. In turn, it can be used to estimate α and β [5].

# of Δt (minutes)	0	5	10	15	20	25	30	45	60	75	90	120
T_{liquid} ($^\circ F$)	200	182	169	159	151	144	137	123	113	106	100	91

Table 1: Measured values of liquid's temperature at different intervals

Each temperature measurement in Table 1 is the arithmetic average of three independent trials performed using a domestic cooking thermometer (with an accuracy of 0.1 $^\circ F$) with water as the liquid filled to the brim in a 16 oz. Moscow Mule mug as the container [5–7]. The program must continue to run until the liquid's temperature does not change by more than 0.1 $^\circ F$. Plot T_n and T_{room} as a function of t_n, and analyze it. You may use aptly named Gnuplot (or other program but not Microsoft Excel) files if your programming language of choice does not provide implicit visualization.

Problem 4

Write a program, `matrix_multiply.EXTN`, to multiply a $N \times N$ matrix (A) and a $N \times 1$ matrix (B) to obtain a $N \times 1$ matrix (C). Test the program with following definitions of A and B for $N = 2 : 1 : 4$ (i.e., from two to four in steps of one). Commit the program to your GitHub repository with an appropriate message before proceeding ahead.

$$a_{ij} = \begin{cases} 1 & \text{If } i = j \\ 0 & \text{Otherwise} \end{cases}$$

$$b_{ij} = i + j + 2$$

Upon making sure that the program produces desirable results, comment the previous definition of A (still can be used for debugging purposes). Populate A and B (B's definition is the same as before) as follows:

$$a_{ij} = i \times (N - i - 1) \times j \times (N - j - 1)$$

Run the program for the following values of $N = 4$, 8, 16, 32, 64, 128, and note down the time required to the complete matrix multiplication, t_{N}. Can the performance for $N = 2$ be used to predict that of $N = 4$? Can the performance of $N = 2$ and $N = 4$ be used to predict that of $N = 8$? Can the performance of $N = 2$, $N = 4$ and $N = 8$ be used to predict that of $N = 16$, and so on?

Store the data, N and t_{N} in `matrix_multiply.dat`. Plot t_{N} as a function of N, and perform curve fitting. You may use aptly named Gnuplot (or other program but not Microsoft Excel) files if your programming language of choice does not provide implicit visualization. Write a note on the scalability of this program.

Problem 5

Isle Royale National Park [8] serves as a grand stage for the longest continuous research study of any predator-prey system in the world [9]. The institutional support for this study is provided by Michigan Technological University [10], National Park Service [11] and National Science Foundation [12].

The Lotka-Volterra equations [13–18], better known as the *predator-prey* equations, are a pair of first-order non-linear differential equations used to describe the dynamics of biological systems in which two species interact, one as a predator and the other as prey. These Lotka-Volterra system of equations is an example of a Kolmogorov model. The latter is a more generic framework which can be used to model the dynamics of ecological systems with predator-prey interactions, competition, disease, and mutualism.

The change in populations with time is given by

$$\frac{dm}{dt} = \alpha\, m - \beta\, m\, w \qquad \frac{dw}{dt} = \mu\, m\, w - \xi\, w \tag{5.1}$$

where α, β, μ, and ξ are parameters describing the interaction between the two species, and have the following meaning:

m	Moose (prey)
w	Wolves (predator)
α	Natural growth rate of prey
β	Death rate of prey
μ	Growth rate of predator
ξ	Natural death rate of predator

The values of $w_\circ$, $m_\circ$, α, β, μ, ξ, and h should be derived from a statistical analysis of the experimental (i.e., measured) data between 1959 and 2016. Save this measured data in `wolf_moose_expt.dat` (three columns: year, wolf population, moose population). Write a program, `wolf_moose.EXTN`, to model and solve this predator-prey system using RK4 method under the following assumptions.

1. Predator (wolf) and prey (moose) live in a closed system
2. Prey finds enough food at all times
3. Predator depends exclusively on prey population
4. Population growth is proportional to its size

5. Prey dies exclusively due to predation
6. Predator dies exclusively due to natural causes
7. Genetic adaptation is slow
8. Predator and prey generations overlap
9. Nature does not prefer one species over the other
10. Behavior of other species does not impact predator-prey interaction

The program should save the simulated values of wolf and moose population after each time step in `wolf_moose_sim.dat` (three columns: year, wolf population, moose population). Plot the simulated wolf and moose population as a function of time along with the corresponding measured values as shown in Fig. 1. You may use aptly named Gnuplot (or other program but not Microsoft Excel) files if your programming language of choice does not provide implicit visualization. Write a note on the scalability of this program.

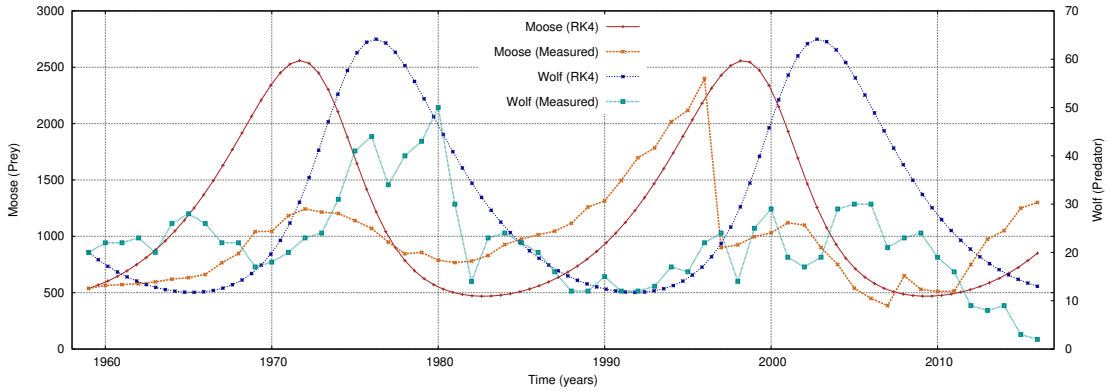


Figure 1: Measured and simulated wolf-moose population from 1959 – 2016

Explain the discrepancies between measured and simulated populations. And if time permits, discuss ways in which the model can be improved.

Problem 6

Heralded as *America's Best Idea* [19], the term *National Parks* is often used to refer to any of the 413 units managed by the National Park Service, an agency of the US Department of the Interior. They include 59 National Parks (e.g., Yellowstone, Yosemite, etc.), most National Monuments (e.g., Devils Tower, Statue of Liberty, etc.), National Preserves (e.g., Big Thicket, Big Cypress, etc.), National Seashores (e.g., Cape Cod, Point Reyes, etc.), and

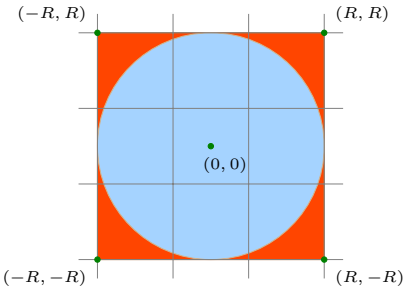
National Lakeshores (e.g., Pictured Rocks, Apostle Islands, etc.). Together, they cover an area of approximately 85 million acres (i.e., ∼340,000 km^2). 14 of the 59 National Parks are also designated as UNESCO World Heritage Sites.

The National Park Service [11] turned 100 years old on 25th August 2016. People of all ages, backgrounds, experiences and interest levels in nature are invited to a year-long celebration by participating in community engagement through recreation, conservation, and historic site preservation (noted by #FindYourPark or #NPS100 in social media). To participate in this centennial celebration, although only computationally, write an aptly named, well-commented and well-modularized program that will find the minimum total travel distance amongst 59 different possibilities (i.e., by treating each park as a starting point) while visiting every National Park exactly once using the *nearest neighbor* algorithm.

Incorporate `distance_slc(`θ_1`, `ϕ_1`, `θ_2`, `ϕ_2`)` or `distance_haversine(`θ_1`, `ϕ_1`, `θ_2`, `ϕ_2`)` from a previous assignment. Consider keeping the necessary data in a file external to the main program, say, `NationalParksList.dat` (five columns: 2-letter state code, name of the park, 4-letter park code, latitude, and longitude; choose an appropriate field delimiter). There should be 59 data files, one corresponding to each route, with 60 lines (61, if using a comment line to indicate the fields) in the following information/format. Name the files to indicate the starting park.

```
PARK_CODE    PARK_NAME    STATE_CODE    LATITUDE    LONGITUDE    DISTANCE
```

Problem 7



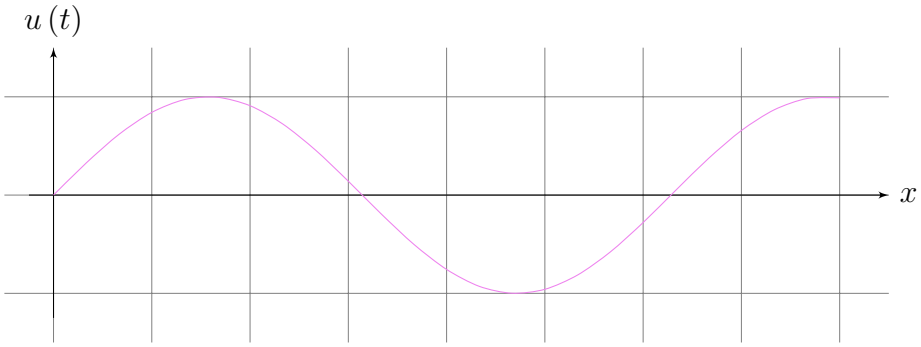
Write a program, `monte_carlo_pi.EXTN`, to estimate the value of of π using the pseudo-code discussed for Monte Carlo method. The program must be capable of repeating itself for the following values of $N = 10^m$ ($m = 0:1:9$), and storing the computed value, π_c, for each N along with corresponding error, ϵ, in `monte_carlo_pi.dat`. Plot separately π_c and ϵ as a function of N. You may use aptly named Gnuplot (or other program but not Microsoft Excel) files if your programming language of choice does not provide implicit visualization.

Write a note on the scalability of this program. The known value of π is 3.141592653589793.

Problem 8

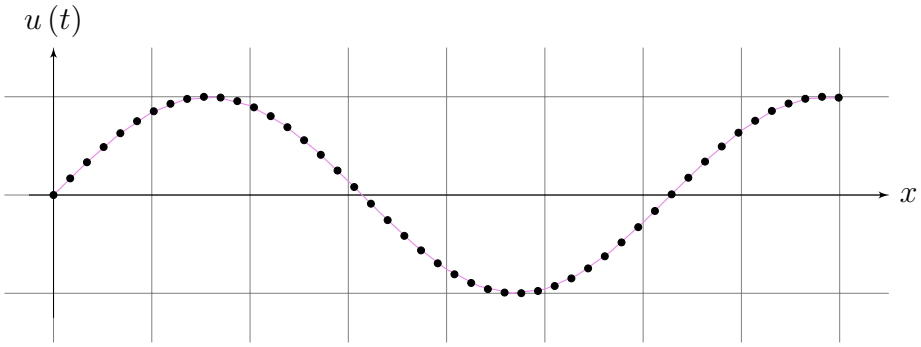
The equation describing the amplitude of a wave, $u(x,t)$, moving at a speed c is given by

$$\frac{\partial^2 u}{\partial t^2} \;=\; c^2\,\frac{\partial^2 u}{\partial x^2}$$



Upon discretizing the domain,

$$x_i \;=\; i\,\Delta\,x \qquad t_n \;=\; n\,\Delta\,t \qquad i,n = 0,1,...,N_{x|t}$$



Replacing the derivatives with finite (central) differences

$$\frac{\partial^2}{\partial t^2}\,u\,(x_i,t_n) \;\simeq\; \frac{u_i^{n+1} \;-\; 2\,u_i^n \;+\; u_i^{n-1}}{\Delta\,t^2} \qquad \frac{\partial^2}{\partial x^2}\,u\,(x_i,t_n) \;\simeq\; \frac{u_{i+1}^n \;-\; 2\,u_i^n \;+\; u_{i-1}^n}{\Delta\,x^2}$$

Substituting the finite differences in the wave equation

$$\frac{u_i^{n+1} \;-\; 2\,u_i^n \;+\; u_i^{n-1}}{\Delta\,t^2} \;=\; c^2\,\frac{u_{i+1}^n \;-\; 2\,u_i^n \;+\; u_{i-1}^n}{\Delta\,x^2}$$

Suppose that u_i^n and u_i^{n-1} are already computed for all values of i. Recursive relation for the unknown quantity, u_i^{n+1}, will be given by

$$u_i^{n+1} \;=\; 2\,u_i^n \;-\; u_i^{n-1} \;+\; C^2\,\left(u_{i-1}^n \;-\; 2\,u_i^n \;+\; u_{i+1}^n\right)$$

$C \;=\; c\,(\Delta\,t/\Delta\,x)$ is a dimensionless quantity known as the *Courant number*. In algorithmic/programmatic terms,

$$u\,(i,t+1) \;=\; 2\,u\,(i,t) \;-\; u\,(i,t-1) \;+\; C^2\,\left[u\,(i-1,t) \;-\; 2\,u\,(i,t) \;+\; u\,(i+1,t)\right]$$

Pseudo-code for a plucked string with ends held fixed is given as below:

```
Define spatial and temporal domains, x = 0 : L, t = 0 : T; c, Δt
Define uniform mesh in x direction with N + 2 points
x_0 and x_{N+1} are left and right boundary points
Choose Δx and Δt such that C = c (Δt/Δx) ⩽ 1.00
Define u_now[N + 2], u_old[N + 2], u_new[N + 2]
Populate u_now[i] = 0.50 − 0.50 ∗ cos(2 i π Δx/L); i = 1 : 1 : N
Set u_now[0] = u_now[N + 1] = 0, and u_old = u_now
For t = Δt : Δt : T (for loop #1)
  For i = 1 : 1 : N (for loop #2)
    u_new[i] = 2 u_now[i] − u_old[i] + C^2 (u_now[i − 1] − 2 u_now[i] + u_now[i + 1])
  End (for loop #2)
  Set u_new[0] = u_new[N + 1] = 0
  Set u_old = u_now and u_now = u_new
End (for loop #1)
```

Write a program, `wave_equation.EXTN`, to model a plucked string with its ends held fixed using the above pseudo-code. Select appropriate values for N, L, c, $\Delta\,t$, and T. Store the data, $u(i,t)$ after each time step in `wave_equation.dat`. Create a visualization (i.e., an animation) using this data and save it as `wave_equation.avi` or `wave_equation.mov`. Is the data (and in turn, the visualization) in compliance with your expectations?

Run the program for different values of N, note down the time in each case (t_{N}) and analyze the results. Is there a pattern?

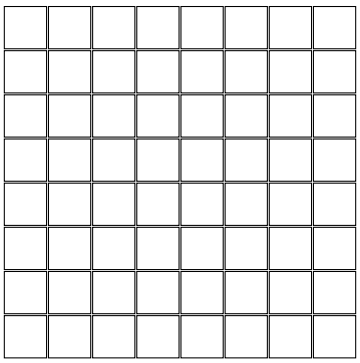
Problem 9

The equation describing the temperature of a plate with thermal diffusivity k at position (x, y) at time t, $u(x, y, t)$, is given by

$$\frac{\partial u}{\partial t} \;=\; k \left[\frac{\partial^2 u}{\partial x^2} \;+\; \frac{\partial^2 u}{\partial y^2} \right]$$

Upon discretizing the domain,

$$x_i \;=\; i\,\Delta\,x \qquad y_j \;=\; j\,\Delta\,y \qquad t_n \;=\; n\,\Delta\,t \qquad i, j, n = 0, 1, ..., N_{x|y|t}$$



Replacing the derivatives with finite (central) differences

$$\frac{\partial}{\partial t}\, u\left(x_i, y_j, t_n\right) \;\simeq\; \frac{u_{i,j}^{n+1} \;-\; u_{i,j}^{n}}{\Delta\, t}$$

$$\frac{\partial^2}{\partial x^2}\, u\left(x_i, y_j, t_n\right) \;\simeq\; \frac{u_{i+1,j}^{n} \;-\; 2\, u_{i,j}^{n} \;+\; u_{i-1,j}^{n}}{\Delta\, x^2}$$

$$\frac{\partial^2}{\partial y^2}\, u\left(x_i, y_j, t_n\right) \;\simeq\; \frac{u_{i,j+1}^{n} \;-\; 2\, u_{i,j}^{n} \;+\; u_{i,j-1}^{n}}{\Delta\, y^2}$$

Substituting the finite differences in the heat equation

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = k \left[\frac{u_{i+1,j}^{n} - 2\,u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2\,u_{i,j}^{n} + u_{i,j-1}^{n}}{\Delta y^2} \right]$$

Suppose that $u_{i,j}^{n}$ and $u_{i,j}^{n-1}$ are already computed for all values of i and j. Recursive relation for the unknown quantity, $u_{i,j}^{n+1}$, will be given by

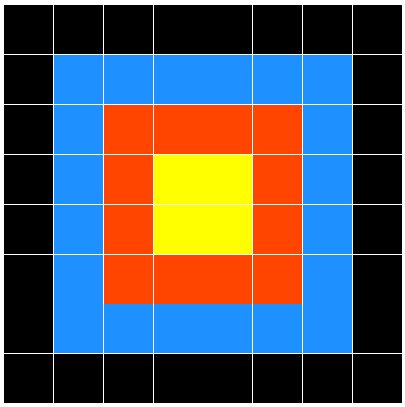
$$u_{i,j}^{n+1} = u_{i,j}^{n} + C_x \left(u_{i+1,j}^{n} - 2\,u_{i,j}^{n} + u_{i+1,j}^{n} \right) + C_y \left(u_{i,j+1}^{n} - 2\,u_{i,j}^{n} + u_{i,j-1}^{n} \right)$$

In algorithmic/programmatic terms,

$$\begin{aligned} u\,(i,j,t+1) = {} & u\,(i,j,t) + \\ & C_x \left[u\,(i+1,j,t) - 2\,u\,(i,j,t) + u\,(i-1,j,t) \right] + \\ & C_y \left[u\,(i,j+1,t) - 2\,u\,(i,j,t) + u\,(i,j-1,t) \right] \end{aligned}$$

$C_x = k\,(\Delta t/\Delta x^2)$ and $C_y = k\,(\Delta t/\Delta y^2)$ play an important role in determining the stability of the solution. For an explicit scheme to be stable,

$$\frac{k\,\Delta t}{\min\left(\Delta x^2,\ \Delta y^2\right)} \leqslant 0.50$$



One possible way to populate $u(x, y, t = 0)$ to reflect the above temperature spectrum (i.e., very hot in the middle with a gradual decrease towards the edge and the edges attached to a heat sink) is given by

$$u\,(i,j,t=0) = 100 \times [i \times (NX - i - 1)] \times [j \times (NY - j - 1)]$$

Pseudo-code for a square plate with edges attached to a heat sink is given below:

```
Define k, T, C_x, C_y, Δx, Δy, Δt, N (uniform mesh)
Choose Δx, Δy, Δt such that solution is stable
Define u_new[N][N], u_now[N][N]
Populate u_now[i][j] = 100 * i * (N - i - 1) * j * (N - j - 1)
For t = Δt : Δt : T (for loop #1)
  For i, j = 2 : 1 : N - 1 (for loop #2)
    u_new[i][j] = finite difference formula
  End (for loop #2)
  Set u_now = u_new
End (for loop #1)
```

Suppose that the idea is to find the time for complete cool down, i.e., find T such that $u(x, y, t = T) = 0$. How would the above pseudo-code change?

Write a program, `heat_equation.EXTN`, to model a square plate with its edges attached to a heat sink using the above pseudo-code but with necessary modifications to ensure the square plate is completely cooled. Select appropriate values for k, T, C_x, C_y, Δx, Δy, Δt, and N. Store the data, $u(i, j, t)$ after each time step in `heat_equation.dat`. Create a visualization (i.e., an animation) using this data and save it as `heat_equation.avi` or `heat_equation.mov`. Is the data (and in turn, the visualization) in compliance with your expectations?

Run the program for different values of N, note down the time to complete cool down in each case (t_{N}) as well as the number of time steps necessary to achieve complete cool down (T) and analyze the results. Is there a pattern?

Problem 10

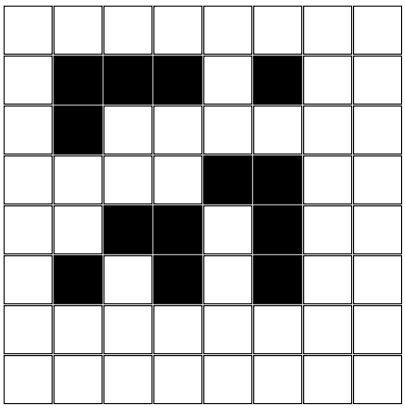
A cellular automaton has the following features:

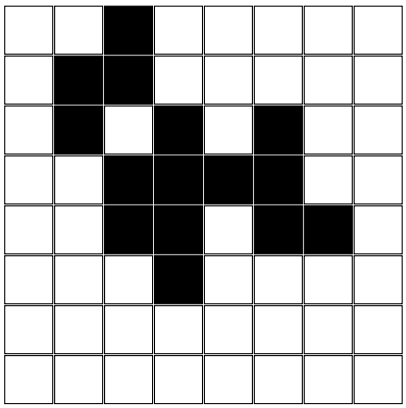
1. A regular grid of cells of any finite number of dimensions
2. Each cell has a finite number of states of existence
3. An initial state to each cell is assigned $t = 0$
4. A new generation is created at $t = 1$ via a rule
5. Rule for updating the state of cells is the same for all cells
6. Rule usually does not change over time
7. Rule is usually applied to the whole grid simultaneously

Game of Life is a 2D cellular automaton and can emulate the Universal Turing Machine. It consists of a grid of square cells having exactly two states: alive (black) or dead (white). The rules of the game (or interaction between cells) are as follows:

1. A dead cell with exactly 3 live neighbors becomes alive
2. A live cell with 2 or 3 live neighbors lives on
3. A live cell with less than 2 live neighbors dies
4. A live cell with more than 3 live neighbors dies

in the next time step. Suppose that a square grid of size N is chosen with the following initial configuration: the edge cells are dead, and a coin is flipped to decide if an inner cell is alive or dead. A sample transition from $t = t$ to $t = t + 1$ is shown below.





Write a program, `game_of_life.EXTN`, to mimic the game of life for T time steps. Select appropriate values for T and N. Store the data after each time step in `game_of_life.dat`. Create a visualization (i.e., an animation) using this data and save it as `game_of_life.avi` or `game_of_life.mov`. Is the data (and in turn, the visualization) in compliance with your expectations?

Run the program for different values of N, note down the time in each case (t_{N}) and analyze the results. Is there a pattern?

> Can problems #4, #6, #7, #8, #9, and #10 be split up into a series of sequential and independent parallel tasks? If yes, list them. Think of ways to implement such a strategy with M processors. How do you expect the problem will scale as a function of M for a given N in each case?

References

[1] L. F. Menabrea. Sketch Of The Analytical Engine Invented By Charles Babbage. *Bibliothéque Universelle de Genéve*, (82), 1842.

[2] A. Lovelace. Tranlslator Notes For Sketch Of The Analytical Engine Invented By Charles Babbage. *Scientific Memoirs*, 3, 1843.

[3] J. Daniels. *Running Formula*. Human Kinetics Publishers, 1998.

[4] Dr. Christopher Schwartz.
Lecturer, Kinesiology and Integrative Physiology, Michigan Tech
Discussions on and off campus on 13th October 2016.

[5] Dr. Allen Curran.
Chief Technology Officer, ThermoAnalytics, Inc.
Discussions in Keweenaw Brewing Company on 13th and 14th October 2016.

[6] Dr. Robert Handler.
Adj. Asst. Professor, Civil and Envinronmental Engineering, Michigan Tech
Operations Manager, Sustainable Futures Institute, Michigan Tech
Discussions on and off campus on 13th and 14th October 2016
Provided the Moscow Mule mug for measurements.

[7] Christine Handler.
Environmental Coordinator, US Forest Service
Discussions off campus on 15th October 2016
Provided a second Moscow Mule mug for second independent measurement.

[8] Isle Royale National Park.
`http://www.nps.gov/isro/`.

[9] J. A. Vucetich and R. O. Peterson. *The Population Biology Of Isle Royale Wolves And Moose: An Overview*.
`http://www.isleroyalewolf.org/`, 2012.

[10] Michigan Technological University.
`http://www.mtu.edu/`.

[11] National Park Service.
`http://www.nps.gov/`.

[12] National Science Foundation.
`http://www.nsf.gov/`.

[13] A. J. Lotka. Contribution To The Theory Of Periodic Reaction. *Journal of Physical Chemistry*, 14:271, 1910.

[14] A. J. Lotka. Analytical Note On Certain Rhythmic Relations In Organic Systems. *Proceedings of the National Academy of Sciences*, 6:410, 1920.

[15] V. Volterra. Variazioni e fluttuazioni del numero dindividui in specie animali conviventi. *Mem. Accademia dei Lincei Roma*, 2:31, 1926.

[16] V. Volterra. *Variations And Fluctuations Of The Number Of Individuals In Animal Species Living Together In Animal Ecology*. McGraw-Hill, 1931.

[17] H. I. Freedman. Deterministic Mathematical Models In Population Ecology. *Canadian Journal of Statistics*, 10:315, 1980.

[18] F. Brauer and C. Castillo-Chavez. Mathematical Models In Population Biology And Epidemiology. *The American Mathematical Monthly*, 110:254, 2003.

[19] K. Burns. *The National Parks, America's Best Idea*. PBS, 2009.

Guidelines

1. Every problem in this assignment has a value beyond its numerical score. It is either a *swan kick* moment for what you already know OR a *wax on, wax off* training for something you will learn soon. Searching the literature to learn more about the underlying concepts and treating every problem as a tiny research project (and the assignment as a proposal seeking external funding) is very highly encouraged

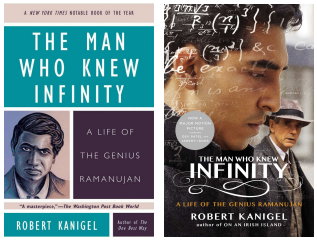
2. You may seek help from anyone as long as you understand and cite it appropriately. Lack of citation will lead to a case of academic dishonesty. Write the programs and scripts in compliance with programming etiquette discussion. Lack of clarity, readability and usage instructions (including information about the OS and software version you used for testing) will make the solution ineligible for grading

3. Review course material, additional references, and your notes, if necessary. Give yourself enough time to complete the assignment on or ahead of time. Check the assignment for spelling and grammatical correctness. Include suggested corrections from the instructor, if any, and show continuous improvement. Contact the instructor if you didn't understand the course material and/or the assignment

4. Typeset your solutions/answers in `${USER}_07.tex`, `${USER}_07.pdf`, and other files as indicated throughout this assignment. Include imagery to explain your answers as and when necessary. Do not include the source code or data in the `${USER}_07.tex` document. There needs to be at least one commit per problem to the GitHub repository to show timely work, demonstrate your work ethic, and a final tagged version of the assignment for submission to earn any credit

5. Problems #1 – #6 are necessary for graduate students, and is worth 2.50% each. Undergraduate and non-traditional students may solve any four of the first six problems, and each problem is worth 3.75%. Problems #7 – #10 represent an opportunity to *do a little more*. Each is worth 0.50%. Deadline for seeking help from the instructor: 7:59 am on Friday, 4th Nov 2016

   The first correct and complete submission earns the following combination



Assignment Submission Workflow

1. Replace `#1` with the appropriate problem number. Keep all necessary files and folders under

   `${UN5390}/CourseWork/Week_07/${USER}_07/`

2. Before starting to work on a problem (or part thereof)

   ```
   cd ${UN5390}
   git pull
   ```

   Suggestions to correct/improve your answers, if any when requested by you, can be found in `${UN5390}/CourseWork/Week_07/suggested_corrections_07.txt`

3. After each problem (or a part thereof), commit to the repository

   ```
   cd ${UN5390}/LaTeXTemplates/Course/
   git pull
   git add ${USER}.bib
   cd ../../CourseWork/Week_07/
   git add ${USER}_07
   git commit -m "Submitting problem #1 in assignment #07"
   git push origin master
   ```

4. When the assignment in its entirety is ready for final submission

   ```
   cd ${UN5390}/LaTeXTemplates/Course/
   git pull
   git add ${USER}.bib
   cd ../../CourseWork/Week_07/
   git add ${USER}_07
   git tag -a a07 -m "Submitting assignment #07"
   git push origin a07
   ```

5. If you made edits to your submission after tagging but before the final due date

   ```
   cd ${UN5390}/LaTeXTemplates/Course/
   git pull
   git add ${USER}.bib
   cd ../../CourseWork/Week_07/
   git add ${USER}_07
   git tag -a a07.1 -m "Submitting assignment #07"
   git push origin a07.1
   ```

   and so on