UN5390: Scientific Computing I

Michigan Technological University
Fall 2016

Assignment # 07

Sandeep Lanka (`slanka@mtu.edu`) · Micheal Roggemann (`mroggema@mtu.edu`)

Problem 1

Augusta Ada King-Noel, Countess of Lovelace (born Byron; 10 December,1815 − 27 November,1852) was an English mathematician and writer, chiefly known for her work on Charles Babbage's early mechanical general-purpose computer, the "Analytical Engine".

She was the daughter of poet George, Lord Byron. Her educational and social exploits brought her into contact with scientists such as Andrew Crosse, Sir David Brewster, Charles Wheatstone, Michael Faraday and the author Charles Dickens, which she used to further her education. Ada described her approach as "poetical science" and herself as an "Analyst (and metaphysician)".

Lovelace became close friends with her tutor Mary Somerville, who introduced her to Charles Babbage in 1833. During a nine-month period in 1842−43, Lovelace translated the Italian mathematician Luigi Menabrea's article on Babbage's newest proposed machine, the analytical engine. With the article, she appended a set of notes explaining the analytical engine's function was a difficult task, as even many other scientists did not really grasp the concept and the British establishment was uninterested in it, Lovelace's notes even had to explain how the Analytical Engine differed from the original "Difference Engine". Her work was well received at the time; the scientist Michael Faraday described himself as a supporter of her writing.

Second tuesday of October is celebrated as Ada Lovelace Day (ALD) in honor of her contributions. The date is arbitrary, chosen in an attempt to make the day maximally convenient for the most number of people avoiding major public holidays, school holidays, exam season, and times of the year when people might be hibernating. Ada was born on 10 December and, in the UK where Ada Lovelace day is based, December is swamped by Christmas parties, making venue hire tricky. Given her tragically early death at the age of 36, it would feel inappropriate to celebrate her deathday on 27 November.[1]

Ada Lovelace is known for the algorithm that could compute Bernoulli numbers which was considered by many as the first computer program in the history of mankind, making her the first computer programer.

The function is given as,

$$0 = -\frac{1}{2}.\frac{2n-1}{2n+1} + B_1\left(\frac{2n}{2}\right) + B_3\left(\frac{2n.(2n-1).(2n-1)}{2.3.4}\right) + B_5\left(\frac{2n.(2n-1).(2n-4)}{2.3.4.5.6}\right) +.. + B_{2n-1} \tag{1.1}$$

where B_1, B_3, B_5 are the Bernouli's numbers. The above equation can be written in general form as ,

$$0 = A_0 + A_1B_1 + A_2B_2 + A_3B_3 + A_5B_5 + ... + B_{2n-1} \tag{1.2}$$

So the strategy could be used to calculate Bernoulli's numbers recursively. The first few numbers are,

$$B_2 = \frac{1}{6}, B_4 = \frac{-1}{30}, B_6 = \frac{-1}{42}$$

The machine that used the Algorithm (Analytical Engine) could compute B_{2n} as long as it could store all the previous numbers of Bernoulli.[2] The function for the algorithm is in the file `Bernoullinos.m`[3]

Problem 2

VO2max is the maximum oxygen consumption $[mL/(kg.min)]$. By Daniels and Gilbert formula VO2max can be calculated and subsequently given a distance(S) and VO2max, we can predict the time by numerical methods as given in the program `vo2max.m`. Velocity(v) required in the formula is calcuated as S/t where t is the time. The table is completed as follows:

Table 1: Time given VO2Max and Distance

$d(miles)$	$t(h:mm:ss)$	VO_2Max	$t_c(h:mm:ss)$	$e = \|t - t_c\|$
3.10	00:22:07	43.829318	00:22:07	0
6.20	0:47:47	41.776160	00:47:47	0
13.10	1:43:02	43.231491	1:43:02	0
26.20	4:06:21	36.411280	4:06:21	0

The program that generated the outputs in the table is named `vo2max.m`

Problem 3

Newton's law of cooling states that the rate of heat loss of a body is proportional to the difference in temperatures between the body and its surrounding environment which can be written as,

$$\frac{dT}{dt} = k(T_{room} - T) \tag{3.1}$$

where t is the time at that instance, T is the temperature of the liquid at that instance k is a constant.

With assumptions that the temperature of the room remains constant and also assuming discrete time $(mins)$ steps $\Delta t = 1$.

$$\frac{1}{T_{room} - T} = kdt \tag{3.2}$$

Integrating on both sides, we get,

$$log|T_{room} - T| = kt + c \tag{3.3}$$

$$e^{log|T_{room} - T|} = e^{kt+c} \tag{3.4}$$

$$|T_{room} - T| = C.e^{kt} \tag{3.5}$$

where $C = e^c$.

$$T = T_{room} - C.e^{kt} \tag{3.6}$$

C can be easily found by using the initial conditions i.e at $t = 0$ thus, $C = T_{room} - T$ which is $70 - 200 = -130$ in this problem.

Substituting the above in (3.6), we have the Equation,

$$T = T_{room} - 130.e^{kt} \tag{3.7}$$

For discrete time, this equation can be written as

$$T_{n+1} = T_n + k(T_{room} - T_n)\Delta t \tag{3.8}$$

which can further be written as

$$T_{n+1} = \alpha T_n + \beta \Delta t \tag{3.9}$$

where $\alpha = 1 - k\Delta t$ and $\beta = k\Delta t T_{room}$

Using the equation (3.9), the temperature and the time required for the liquid to reach the room temperature within in a error margin of $0.1^o F$ can be predicted. The program is in the file named NewtonsCooling.m

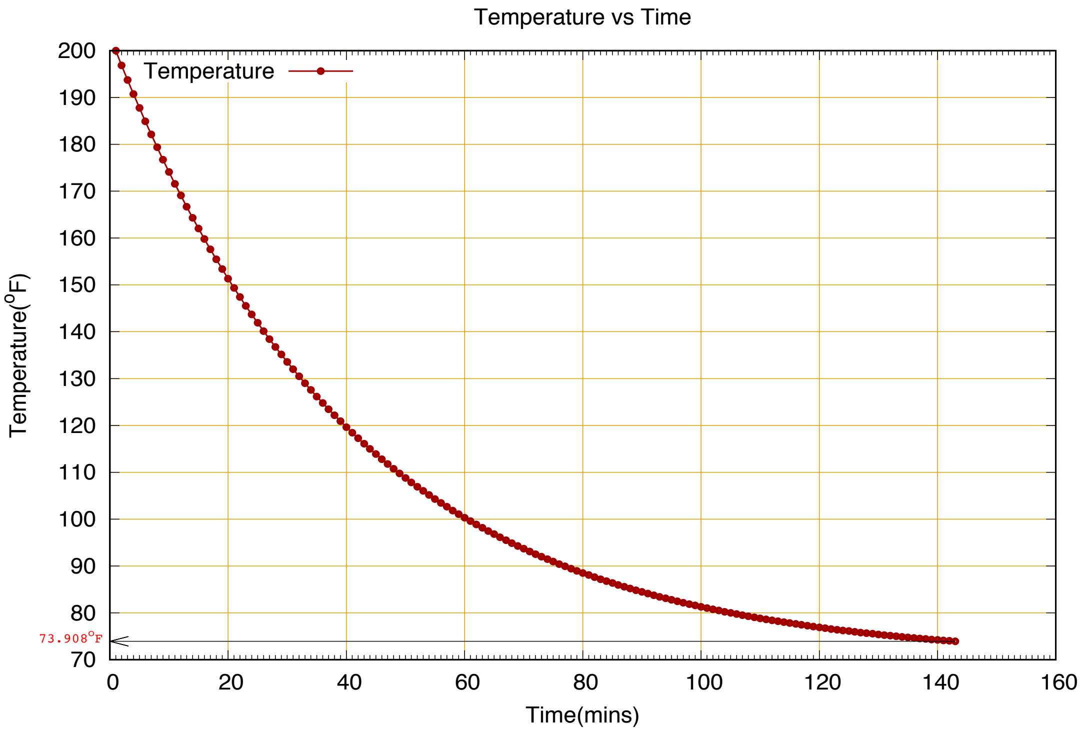

Figure 1: Rate of change of temperature of the liquid w.r.t time

Problem 4

As the N value increases, the time of executions increase. The time of executions t_n are plotted against N as,

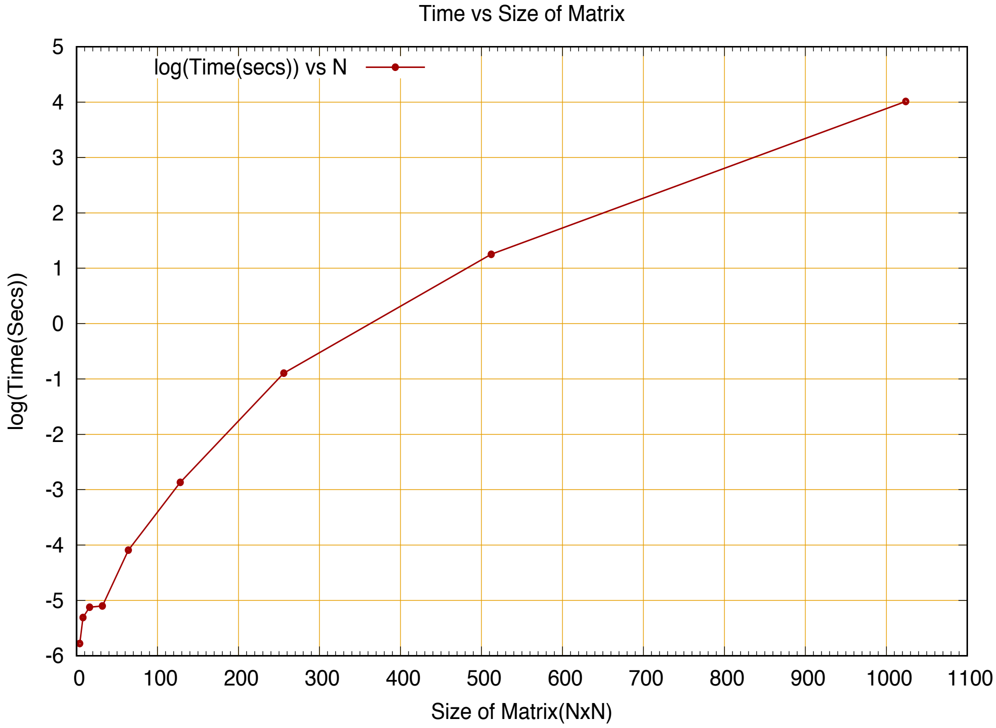

Figure 2: time of executions vs different N's

The data points are fitted to a curve using MATLAB's curve fitting tool. The curve fits well for a Polynomial of 3rd order. The fitted curve can be used to predict the future values of the time. The curve if fitted for a linear model polynomial of 3rd order:[4]

$$f(x) = p1\, x^3 + p2\, x^2 + p3\, x + p4 \tag{4.1}$$

with the coefficients within a 95% confidence are given with the confidence bounds as,

$$p1 = 1.834\, e^{-8} \text{ with 95\% confidence bounds (8.74e-09, 2.794e-08)} \tag{4.2}$$

$$p2 = \text{-3.701 e-5 with 95\% confidence bounds (-5.09e-05, -2.312e-05)} \tag{4.3}$$

$$p3 = \text{0.02833 with 95\% confidence bounds (0.02377, 0.03288)} \tag{4.4}$$

$$p4 = \text{-5.947 with 95\% confidence bounds(-6.179, -5.715)} \tag{4.5}$$

This plot is shown in Figure 3.

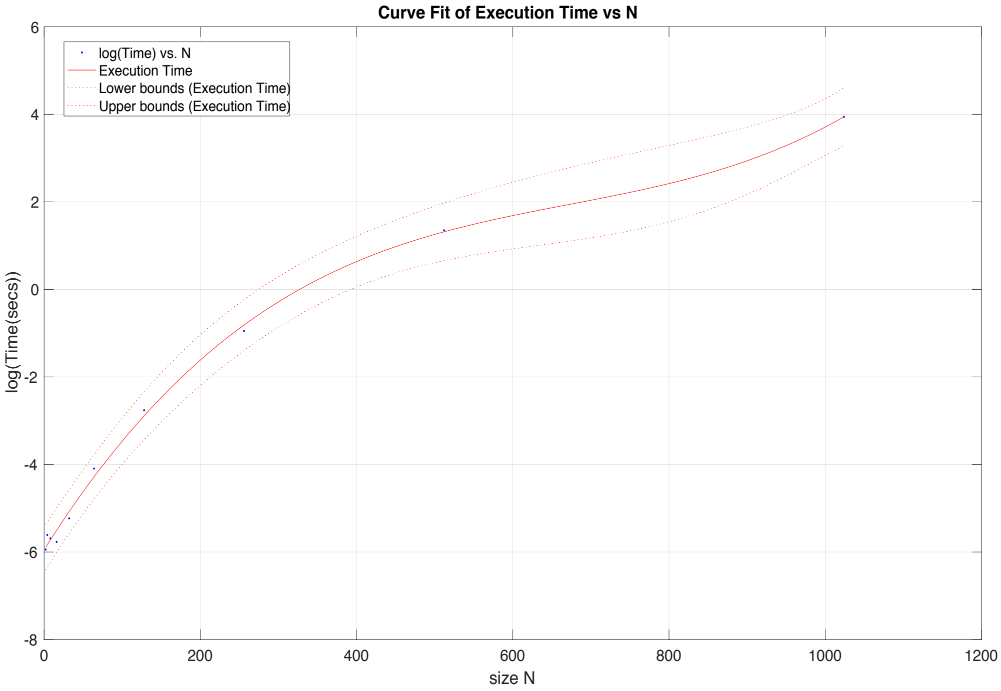

Figure 3: Time of executions vs Different N's

In order to predict the time for a size x_i we simply substitute it in (4.1) with the above coefficients for p and predict it's time within 95% confidence bounds. As the number of elements increase in the matrix it consumes more RAM as it executes. So multiplication of big matrices or finding big matrices' inverse is computionally more difficult. This can be overcome by making the matrice as sparse as we can or using a GPU or any other hardware accelerator to process data parallely to increase the performance in terms of speed of computation.

The program for the matrix multiplication is in the `matrix_multiply.m`

Problem 7

The value of *pi* can be approximated by using the Monte Carlo method. We consider a circle with radius $r = 0.5$ inscribed within a square of area 1. The ratio of area the circle to the area of the square should be equal to *pi*. This can be achieved by randomly picking points in the area and counting the number of points inside the circle. This ratio approximates the value of *pi*. The more the number of random points, the more accurate is the approximation.

The approximation is used ot calcuate the value of *pi* and the error associated with it. Figures below show the accuracy of the approximation with respect to the number of points and also the value of *pi* with respect to the number of points.

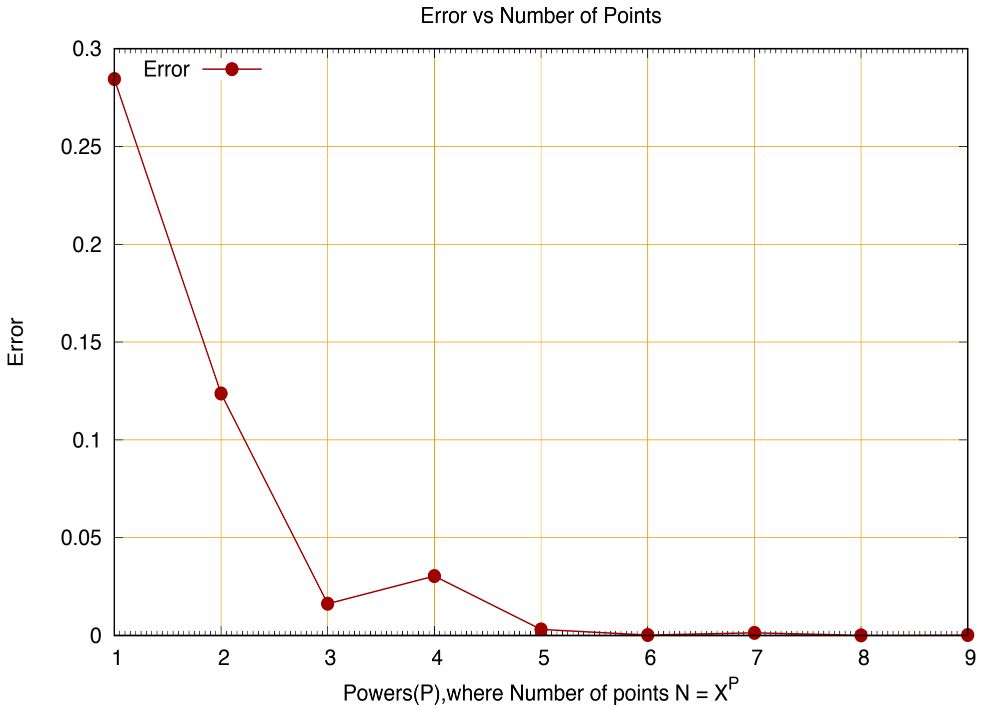

Figure 4: Error vs Number of points

The figures show that as the number of points that approximate the value of *pi*, the more accurate is the value of *pi*. The problem comes only when the approximated value is either too small or too large and there is a need for a large number of points which is computationaly complex and might require large memory. e.g. In cases where there is a large matrix multiplication involved or when the computed value is smaller than the machine epsilon etc.

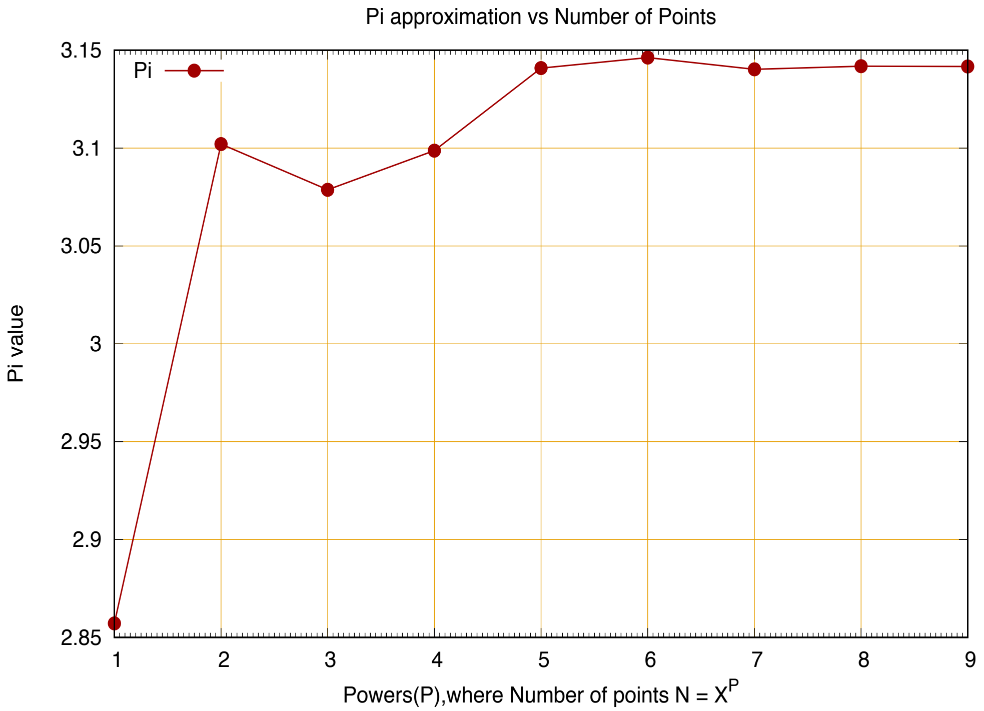

Figure 5: *pi* value vs Number of points

The program for approximating the value of *pi* using the monte carlo method is in the file `MonteCarloPi.m`

References

[1] https://en.wikipedia.org/wiki/Ada_Lovelace.

[2] http://people.maths.ox.ac.uk/kar/AdaLovelace.html.

[3] Prithvi Kambhampati.
On campus Discussions
Discussions on different algorithms of Bernoulli numbers on Nov 6,7 2016.

[4] https://www.mathworks.com/products/curvefitting/.