High Performance computing : Assigment 5

Banele Blessing Zondo ID 149141

June 28, 2020

Section 1 :Steady heat equation

In our problem we have a thin plate of dimensions $a \times b$ that is insulated along its long sides while the left side is embedded in ice-water and right right side is exposed to air.A heating element passes through the centre of the plate, we will consider this heating element to be a wire with 1500 watts heat output.We take initial temperature of the plate to be the room temperature $25°C$, the temperature of the ice to be $0°C$ C and the temperature of the air in the room to be $27°$. We assume that the plate is small so that it does not change the temperature of the air and ice, such that the air and ice act as a sink for heat, that is, it only absorbs the heat from the plate. Since by defination a square is a rectangle with equal sides then I have $a = b$ we make the problem simpler and we also took the plate to be copper with thermal diffusivity 1.1234×10^{-4}.

Section 2: Equations

This problem is a nonhomogenous steady 2 dimesional heat equation with Dirichlet boundary conditions but the steady part of it makes the problem very simple, it becomes a poisson equation, a nonhomogeneous one becauses of the applied heat. For a 2 dimensional heat equation we have

$$\frac{\partial T}{\partial t} = -k\left(\frac{\partial^2 T}{\partial^2 x} + \frac{\partial^2 T}{\partial^2 y}\right) + F(x, y) \tag{1}$$

Since this is a steady heat equation it means the temperature doesn't change with time, that is $\frac{\partial T}{\partial t} = 0$. So we have

$$0 = -k\left(\frac{\partial^2 T}{\partial^2 x} + \frac{\partial^2 T}{\partial^2 y}\right) + F(x, y) \tag{2}$$

Now dicretizating the equation, we used standard Poisson finite difference method

$$-k\left(\frac{T^k_{i-1,j} - 2T^k_{i,j} + T^k_{i+1,j}}{\Delta x^2} + \frac{T^k_{i,j-1} - 2T^k_{i,j} + T^k_{i,j+1}}{\Delta y^2}\right) + F(x, y) = 0 \tag{3}$$

let $\Delta x^2 = \Delta x^2 = h^2$

$$\left(\frac{T^k_{i-1,j} - 4T^k_{i,j} + T^k_{i+1,j} + T^k_{i,j+1} + T^k_{i,j-1}}{h^2}\right) = \frac{1}{k}F(x,y) \quad (4)$$

$$T^k_{i-1,j} - 4T^k_{i,j} + T^k_{i+1,j} + T^k_{i,j-1} + T^k_{i,j+1} = \frac{h^2}{k}F(x,y) \quad (5)$$

$$T^k_{i,j} = \frac{1}{4}\left(T^k_{i-1,j} + T^k_{i+1,j} + T^k_{i,j-1} + T^k_{i,j+1} + \frac{h^2}{k}F(x,y)\right) \quad (6)$$

$$(7)$$

we write this as

$$T_{i,j} = \frac{1}{4}\left(T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1} + \frac{h^2}{k}F(x,y)\right)$$

The boundary equations will be give by

$$T_{0,j} = 0^\circ C \quad (8)$$

$$T_{M,j} = 27^\circ C \quad (9)$$

$$T_{i,0} = 2T_{i,1} + T_{i-1,0} + T_{i+1,0} \quad (10)$$

$$T_{i,N} = 2T_{i,N} + T_{i-1,N} + T_{i+1,N} \quad (11)$$

$$(12)$$

We used iteration scheme with jacobi method to solve the problem. The convergence of the Jacobi method is given by $1 - O(h^2)$. The convergence test or stopping test was

$$\frac{||T_{n+1} - T_n||}{||T_n||} < \epsilon$$

I used three values of ϵ, those were $10^{-4}, 10^{-6}, 10^{-10}$.

Section 3: Algorithm

The sequential code

I first created a matrix for the unknowns with given length and filled in the temperatures according to the intial conditions.Then I create a sparse matrix but since according to the scheme the next value is indepedent of the previous

but is depedent of the surrounding so I made the sparse matrix as

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & . & . & . & 0 \\ 1 & 0 & 1 & 0 & 0 & . & . & . & 0 \\ 0 & 1 & 0 & 1 & 0 & . & . & . & 0 \\ . & . & 1 & 0 & 1 & . & . & . & . \\ . & . & . & 1 & 0 & 1 & . & . & . \\ . & . & . & . & 1 & 0 & 1 & . & . \\ 0 & 0 & 0 & . & . & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & . & . & . & 1 & 0 & 1 \\ 0 & 0 & 0 & . & . & . & . & 1 & 0 \end{pmatrix}$$

If i let this matrix to be $T1$ then a product of this matrix and A(Matrix of unknown) $T1A$ adds up the $T_{i,j+1}$ and $T_{i,j-1}$ assuming the j axis moves up and down, then the product of T matrix and A(Matrix of unknown) $AT1$ adds up the $T_{i-1,j}$ and $T_{i+1,j}$. Now what is let is the insulated boundary conditions let assume the insulated region is on the left and right, the sum of the two products $T1A + AT1$ doen't include some term.The condition is given as

$$\begin{aligned} T_{i,0} &= 2T_{i,1} + T_{i-1,0} + T_{i+1,0} && (13) \\ T_{i,N} &= 2T_{i,N} + T_{i-1,N} + T_{i+1,N} && (14) \\ & && (15) \end{aligned}$$

The product doesn't includes the $2T_{i,1}$ and $2T_{i,N}$ corrently what it does is sum them up as

$$\begin{aligned} T_{i,0} &= T_{i,1} + T_{i-1,0} + T_{i+1,0} && (16) \\ T_{i,N} &= T_{i,N} + T_{i-1,N} + T_{i+1,N} && (17) \\ & && (18) \end{aligned}$$

To correct this and add the second $T_{i,1}$ and $T_{i,N}$ terms such that it is $2T_{i,1}$ and $2T_{i,N}$ as needed i created a second matrix called $T3$ given by

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ 1 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ . & . & 0 & 0 & 0 & . & . & . & . \\ . & . & . & 0 & 0 & 0 & . & . & . \\ . & . & . & . & 0 & 0 & 0 & . & . \\ 0 & 0 & 0 & . & . & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & . & . & . & 0 & 0 & 1 \\ 0 & 0 & 0 & . & . & . & . & 0 & 0 \end{pmatrix}$$

The product $AT3$ corrects insulated boundary condition issue(i will admit after some reading i see they way to make a sparse matrix that include the insulated bountry conditions), so each iteration was given by

$$T_{i,j} = \frac{1}{4}\left(T1A + AT1 + AT3 + F[i,j]\right) \tag{19}$$

Assuming A is the matrix of unknowns where $F[i,j]$ is of the heated wire, then to test for convergence on the first iteration convergence is set to be 1 and on other iterations convergence is started using given stopping condition

$$\frac{||T_{n+1} - T_n||}{||T_n||} < \epsilon$$

using the previous values and current. The previous values are saved before new iteration start, all this was done using a while statement.

parallel code

Main body

On rank 0 I first created a matrix for the unknowns with given length and filled in the temperatures according to the intial conditions. Then broadcasted to other ranks. In all of the ranks i created the sparse matrix $T1$ and $T3$ as i stated in the sequential code part. I created a function that uses all the processors to matrix by matrix returning one matrix in rank 0 (I defined the mechanics of the function below). For each new matrix, we heard then to test for convergence but on the first iteration convergence is set to be 1 and on other iterations convergence is started using given stopping condition

$$\frac{||T_{n+1} - T_n||}{||T_n||} < \epsilon$$

using the previous values and current. The previous values are saved before new iteration start, all this was done using a while statement. Let the function be P

$$T_{i,j} = \frac{1}{4}\left(T1A + AT1 + AT3 + F[i,j]\right) \tag{20}$$

$$\begin{aligned} C1 &= P(T1, A) && (21)\\ C2 &= P(A, T1) && (22)\\ C3 &= P(A, T3) && (23)\\ T_{i,j} &= \frac{1}{4}\left(C1 + C2 + C3 + F\right) && (24) \end{aligned}$$

where $F[i,j]$ is of the heated wire, the new value of T is broadcast to all other processors because a "gather" command in the function gathers the matrix in rank 0.

The Matrix matrix multiplication function

This function has three inputs. At first i wanted each processors to have one row of the first matrix and one column of the second matrix(i think this is called element multiplication) but the number of processors were limited to 15, also thought of giving one row to each processor multiple that by the second matrix(i think this is called block multiplication) but this wouldn't work for matrix with more than 15 rows. So i divided the number of rows by number processors such that each processor has $\frac{N_r}{p}$ rows of the first matrix(where N_r is the number of rows and p is the number of processors).My parallel code doesn't work if the number of rows is not dividable by the number of processors. Consider the product AB , the function takes the first matrix of order $l \times l$ then reshapes it into a matrix or vector of order $1 \times l^2$. It then uses the "scatter" command to split the function equally into a matrix of order $\frac{N_r}{p} \times l$ (where N_r is the number of rows and p is the number of processors) a matrix of this order is created before hand in each processor and it is the second input of the function the last input function is the second matrix in the product(that is given the product AB , A is the first input and B is the third input). A product of the matrix of order $\frac{N_r}{p} \times l$ and matrix B is done in each processors and the result is a matrix of order $\frac{N_r}{p} \times l$ in each processors. Using the "gather" command all the $\frac{N_r}{p} \times l$ matrixes are gathered back into one matrix of order $l \times l$ and that matrix is returned

Section 4 : Results and analysis

Speedup and Efficiency

number of processors	T_p	Speedup S_p	efficiency
2	2.033826512098312	1.7391304347826089	0.8695652173913044
3	1.5327388207117718	2.3076923076923075	0.7692307692307692
4	1.2821949750185013	2.758620689655172	0.689655172413793
5	1.131868667602539	3.1250000000000004	0.6250000000000001
6	1.0316511293252308	3.428571428571429	0.5714285714285715
10	0.8312160527706146	4.25531914893617	0.425531914893617
15	0.7309985144933064	4.25531914893617	0.28368794326241137

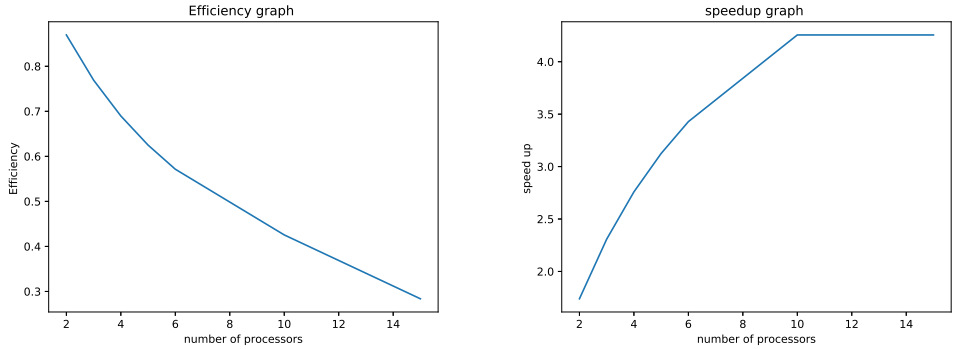
where the T_p , speedup S_p and Effieciency E_p are given by

$$T_p = T1\left(F_s + \frac{F_p}{P}\right) \tag{25}$$

$$S_p = \frac{T_p}{p} \tag{26}$$

$$E_p = \frac{S_p}{p} \tag{27}$$

$$\tag{28}$$



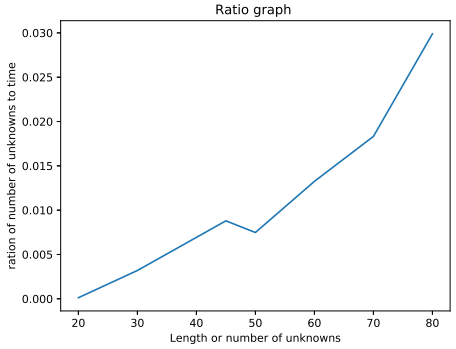
(a) Efficiency (b) Speed up

Figure 1: Speed up and Efficiency

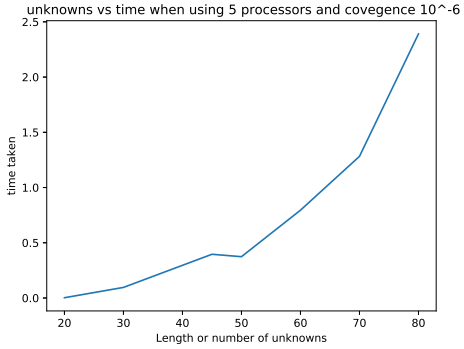
According to Amdahl's Law because most codes are not fully parallizable then speedup on that code is limited to a factor of 20, a code that is 90 percent parallizable would have the highest speedup at factor around 20 and at 75 percent parallizable have speed up that is around factor 4. My code that was around 80 percent to 85 percent parallizable or 80 percent to 85 percent of it was in parallel has a speed up that is limited by a factor of number close to 4, and this is in agreement with amdahl"s Law. The Efficiency appear to decrease with increase of processors appromimating $\sim \frac{1}{p}$ this too is in agreement with Amdahl's Law.

Input Output ration

number of unknowns	Time	Ratio
20	0.0024089813232421875	0.00012044906616210937
30	0.09577298164367676	0.003192432721455892
45	0.3958549499511719	0.008796776665581598
50	0.37390947341918945	0.7948787212371826
60	0.7948787212371826	0.013247978687286378
70	1.2822117805480957	0.01831731115068708
80	2.3909013271331787	0.029886266589164732



(a) Covergence 10^{-6}

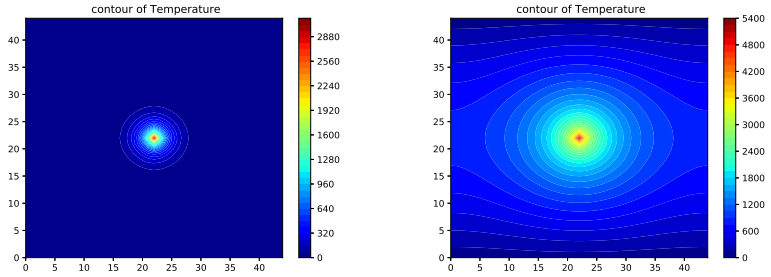


(b) Covergence 10^{-6}

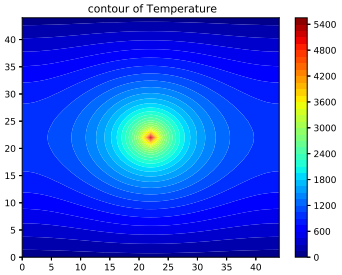
Figure 2: Covergence

Looking at the graph and table of ratio I see that the ratio of number of unknowns(input) to time taken(output) is almost linear, it steadlly increase with increase in unknowns.

The plate at 3 difference times



(a) Covergence 10^{-4} (b) Covergence 10^{-6}

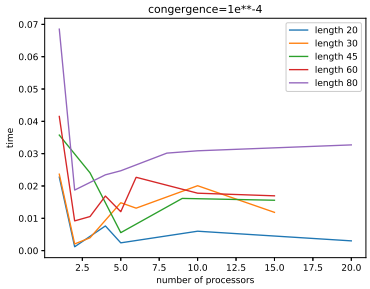


(c) Covergence 10^{-10}

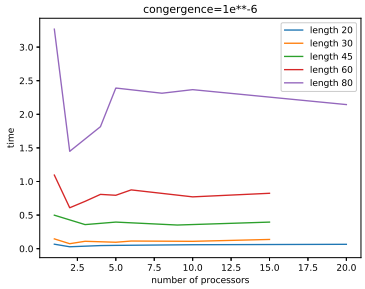
Figure 3: Covergence:Temperature of the plate at different time

The length of the plate is 45, as time increase(time here is represented by the convergence number the smaller the number the longer the time) the temperature of the plate increase and move out from the centre it's distribution is elliptically, and this makes sense if you consider that the temperature would move faster towards the insulted sides than the region exposed to ice and air.

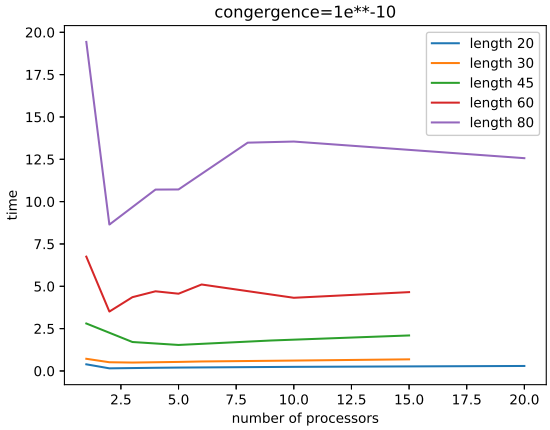
Time for different number of processors



(a) Covergence 10^{-4}



(b) Covergence 10^{-6}



(c) Covergence 10^{-10}

Figure 4: Covergence

More analysis on time shows that Amdahl's law is correct become the increase of processors also increase the communication time causing the time for n processors to approch $T1$. From these graphs we see that for the fewer number of processors the time is very low but as the number of processors increase so does the time, this becomes more clear if the convergence is smaller(that is given more time and iterations).

Iterations

length	$cor10^{-}4$	$cor10^{-}6$	$cor10^{-}10$
20	7	360	1997
30	12	772	4342
45	23	1676	9540
60	37	2862	16697
80	58	4941	29341

From the graph and the table we can see that number of iterations is directionlly propotional to the number of unknowns since the efficiency and speed up is limit and increasing the number of processors doesn't help it means for large unknowns even if you were to use a quantum computer the time would be very long .

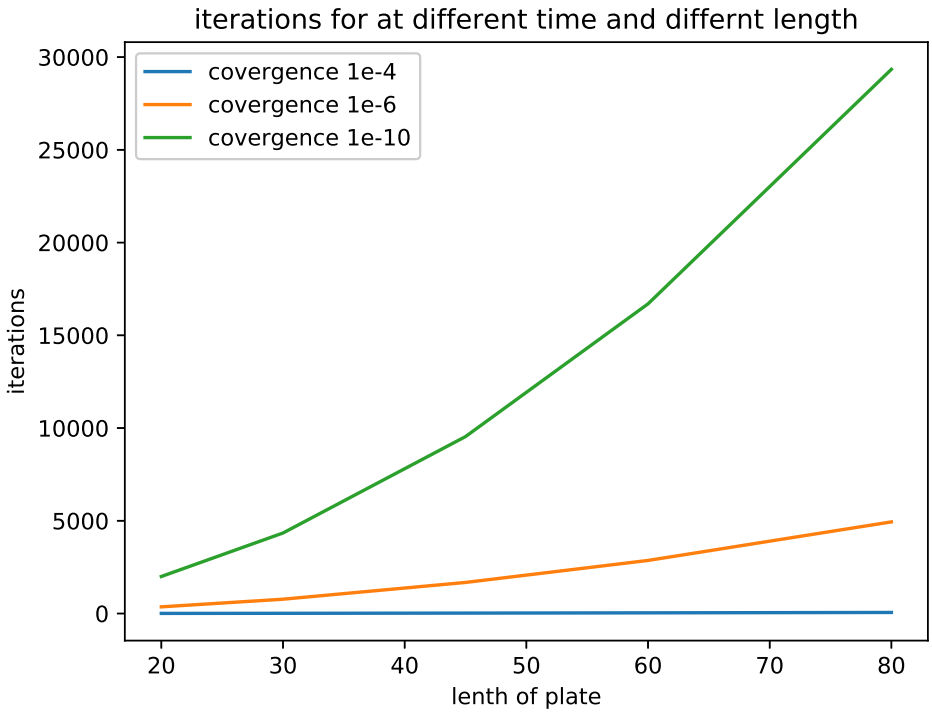
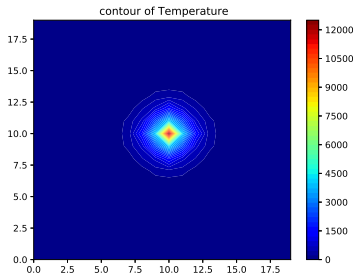


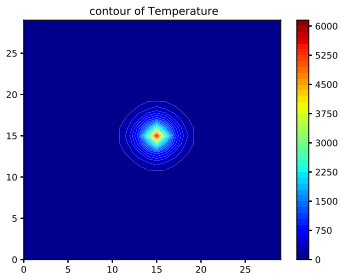
Figure 5: Iterations graph

How the temperture on different lengths given fix time

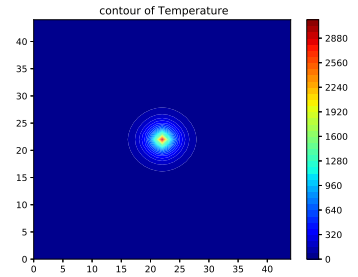
Convergence = $10^{-}4$



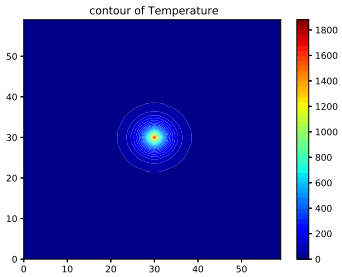
(a) length = 20



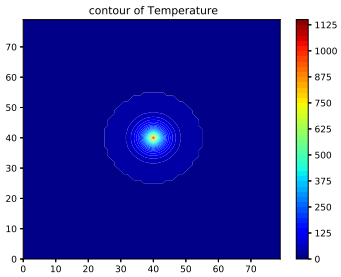
(b) length = 30



(c) length = 45



(d) length = 60

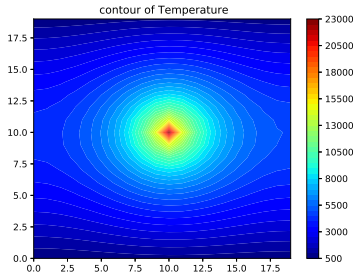


(e) length = 80

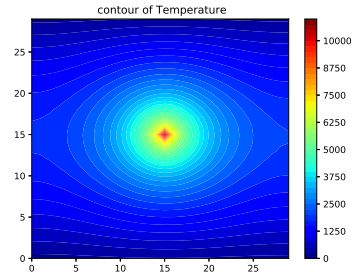
Figure 6: Covergence $10^{-}4$

Early on the size doesn't affect the distribution of temperature, the flow appear circular in all plate sizes.

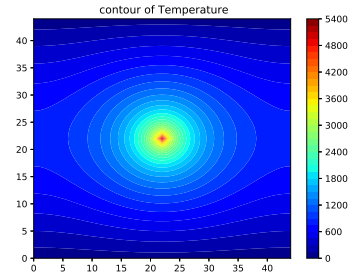
Convergence = $10^{-}6$



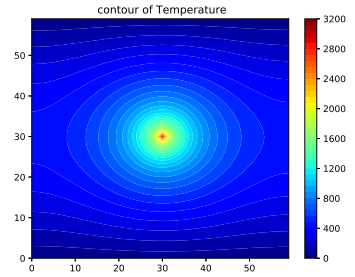
(a) length = 20



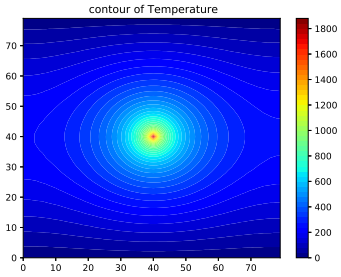
(b) length = 30



(c) length = 45



(d) length = 60

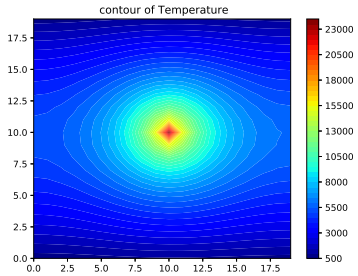


(e) length = 80

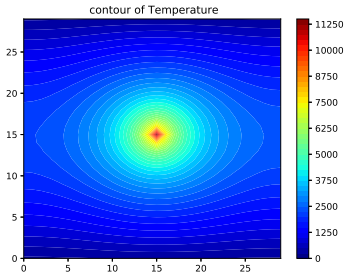
Figure 7: Covergence$10^{-}6$

A difference is starting to show especially between the smallest plate and the largest plate since the temperature difference is high and the high temperature has reach the boundaries.

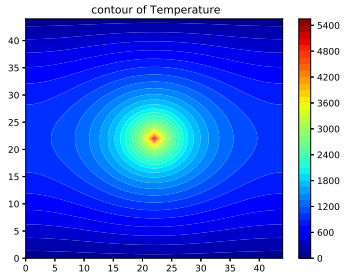
Convergence = $10^{-}10$



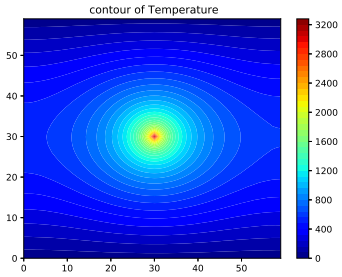
(a) length = 20



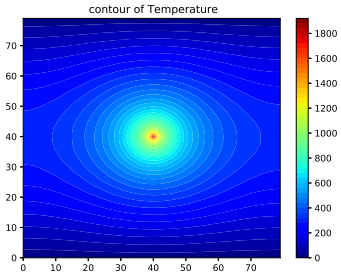
(b) length = 30



(c) length = 45



(d) length = 60



(e) length = 80

Figure 8: Covergence $10^{-}10$

Excluding the difference in temperature between big and small plate, the flow of the temperature through the plate seem to be constant with time which should be obvious because this is a steady state problem. The size in general does affect how high the temperature will be after time T_f but it doesn't affect the distributon of different temperature regions in the plate(how different temperature move from the center).

Section 5: Conclusion

After studying the problem and looking at analysis we have conclude that plitting the problem over more and more processors does not make help, at a certain point there is just not enough work for each processor to operate efficiently.The problem has a weak scalabiltyas problem size and number of processors grow in such a way that the amount of data per processor stays constant, the speed in operations per second of each processor also stays constant.