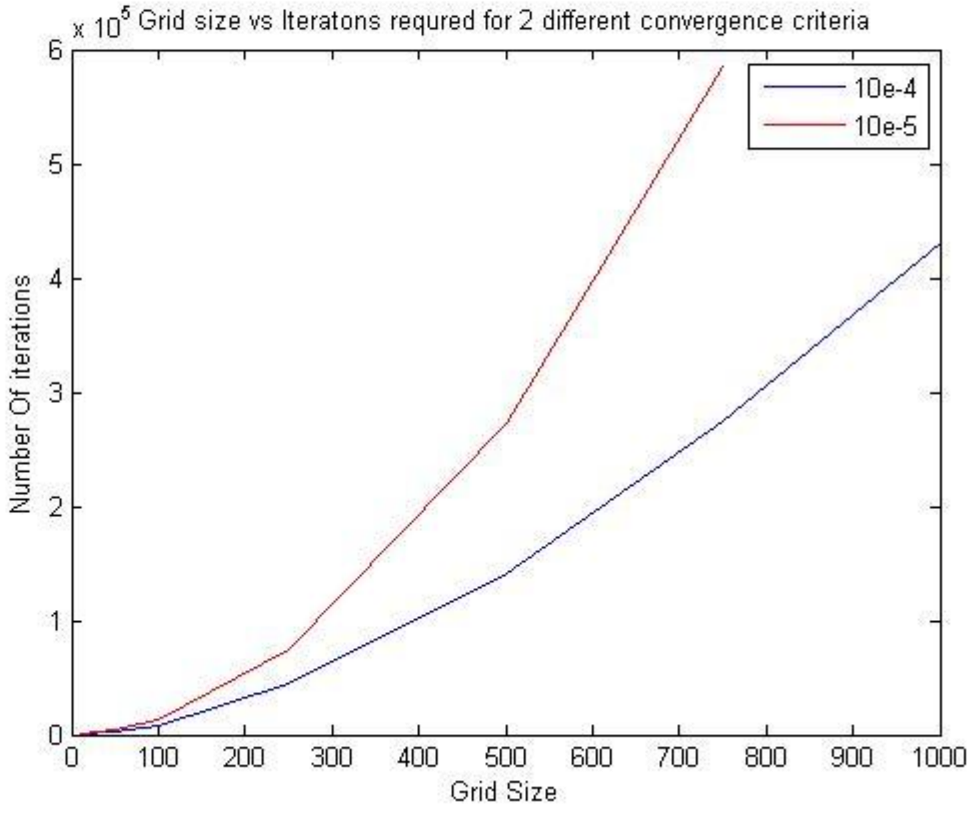
## Question 1

a. Code is attached in the appendix

b. The graph for number of iterations required wrt grid size is as follows



Clearly in the sequential code if the number of grid size is increased the nuber of iterations required for L2 norm condition for convergence increases. Two different convergence criteria were also tested: 10e-4 and 10e-5; which also gives sensible results that lower the convergence criteria, more iterations are required for the same grid size

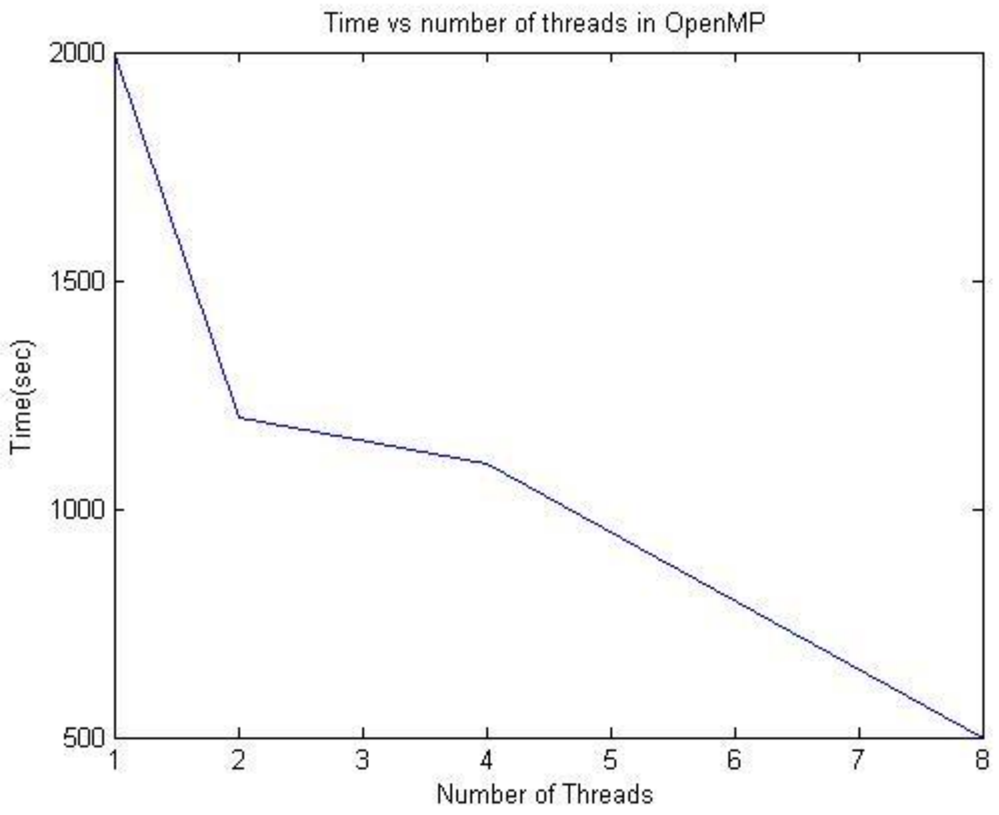
These calculations were carried out on a general 8 core processor on CCR

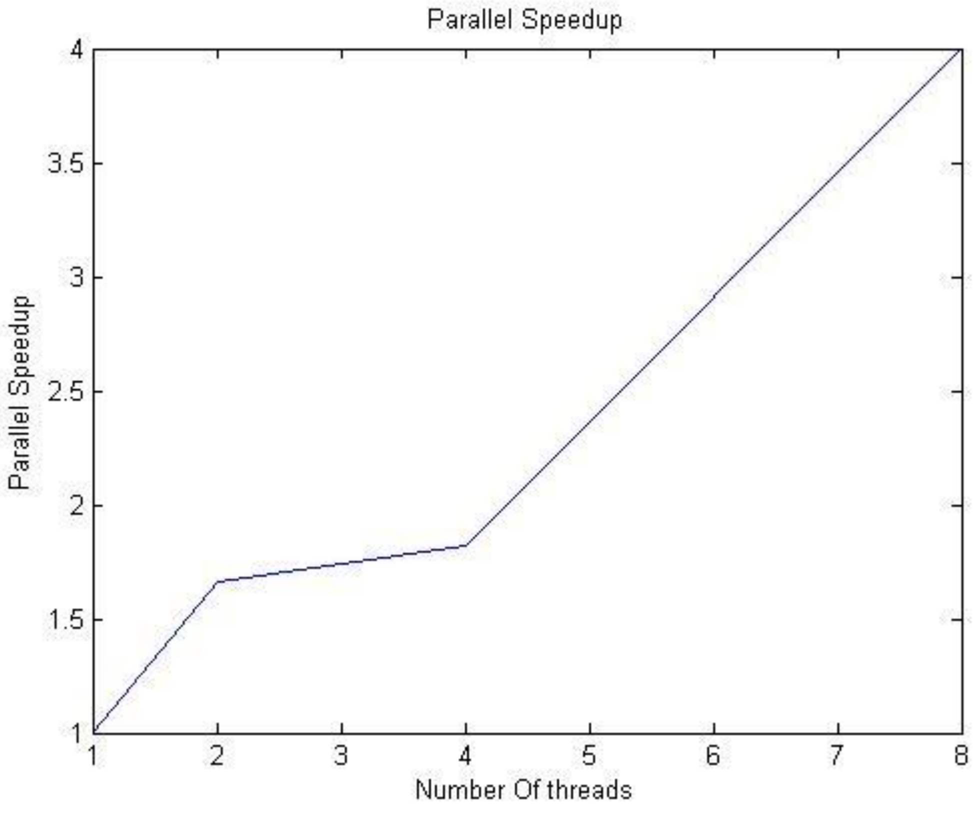
c. The code is attached in the appendix

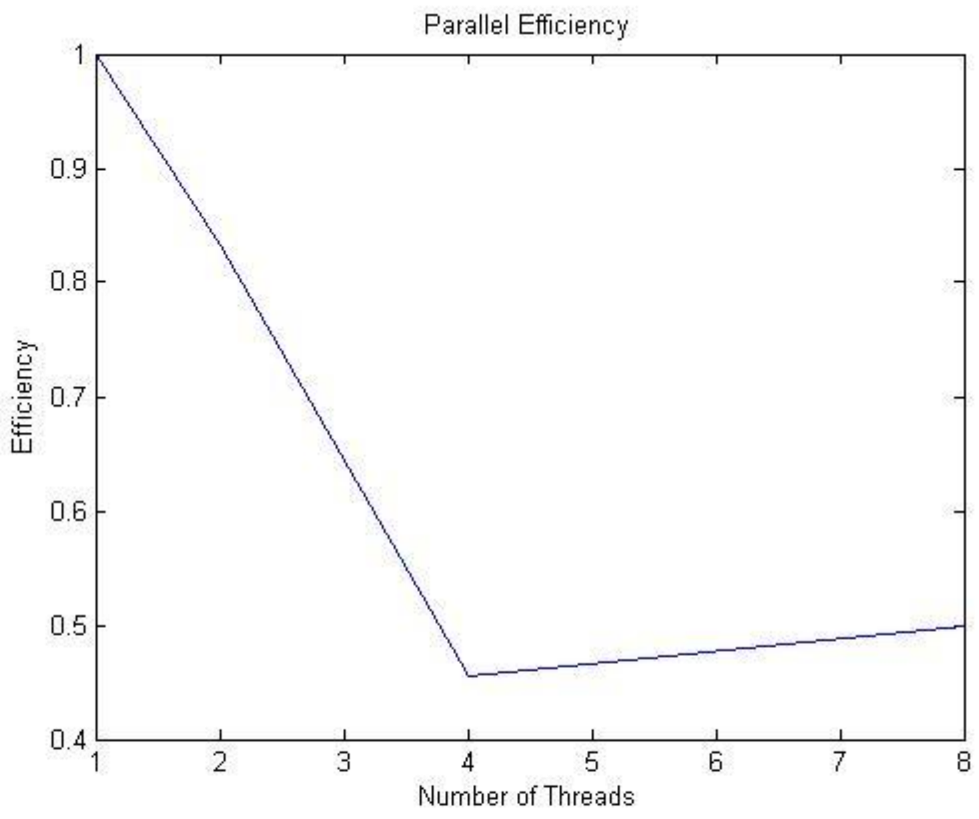
Note: the parallel code was submitted to CCR on node d09n09s02, with 8 threads with variation of number of threads 1,2,4 and 8 threads. Just one node was requested on CCR

d. The graphs after parallelizing the solver with openMP are given below and as expected gives a better parallel speedup and execution time as compared to just one thread, which is also the sequential code.

The grid size was 2000 for all these cases.







Parallel efficiency graph increases for 8 threads which is an indication that for the number of grid size used, it is best to use 4 threads to get the maximum efficiency.

## Appendix 1

```
program lap

implicit none

double precision :: dx,dy,dt, pi=3.14, norm
integer :: Nx,Ny,i,j,n,Nt,nitr
double precision, dimension(100) ::x,y
double precision, dimension(100,100) :: phi,phio
do i=1,100
   do j=1,100
      phio(i,j)=1.0
   end do
end do
open(1,file='result.txt')
n=100
dt=0.01
Nt=int(n/dt)
dx=0.01
dy=0.01
Nx=100
Ny=100

do i=1,Nx
   x(i)=i*dx
   y(i)=i*dy
   !print *,x(i)
end do
print *,x(100)
timeloop: do nitr=1,Nt

   do i=1,100
      phio(1,i)=sin(pi*x(i))
      phio(Nx,i)=sin(pi*x(i))*exp(-pi)
      phio(:,1)=0.0
      phio(:,Ny)=0.0
   end do

   do i=2,Nx-1
      do j=2,Ny-1
         phi(i,j)=(phio(i+1,j)+phio(i,j+1)+phio(i-1,j)+phio(i,j-1))/4

         !print *,norm
         IF (abs(phio(i,j)-phi(i,j))<(10**-5)) exit timeloop
      end do
   end do

   phio=phi
end do timeloop
do i=1,Nx
   do j=1,Ny
      !write(1,*) phi(i,j)
      !print *,phi(i,j)
   end do
   !write(1,*)phi(i,:)
   !write(1,*)''
end do
close(1)
!print *,nitr
end program
```

## Appendix 2

```
program lap
Use omp_lib
implicit none

double precision :: dx,dy,dt, pi=3.14, norm, t_start,t_end
integer :: Nx,Ny,i,j,n,Nt,nitr, Np, Nperprocess
double precision, dimension(2000) ::x,y
double precision, dimension(2000,2000) :: phi,phio
integer, dimension(8) :: Nthreads
Nthreads=(/ 1, 2, 3, 4, 5, 6, 7, 8/)
do i=1,100
   do j=1,100
      phio(i,j)=1.0
   end do
end do
open(1,file='result.txt')
n=1000000
dt=0.01
Nt=int(n/dt)
dx=0.01
dy=0.01
Nx=2000
Ny=2000

do ntr=1,size(Nthreads)
call OMP_SET_NUM_THREADS(Nt)
!$OMP PARALLEL DO
 myid = OMP_GET_THREAD_NUM()
 Nt = omp_get_num_threads()
 Np=omp_get_num_procs()
 Nperprocess= N/Nt
  t_start = OMP_GET_WTIME()


do i=1,Nx
   x(i)=i*dx
   y(i)=i*dy
   !print *,x(i)
end do

timeloop: do nitr=1,Nt

   do i=1,2000
      phio(1,i)=sin(pi*x(i))
      phio(Nx,i)=sin(pi*x(i))*exp(-pi)
      phio(:,1)=0.0
      phio(:,Ny)=0.0
   end do

   do i=2,Nx-1
      do j=2,Ny-1
         phi(i,j)=(phio(i+1,j)+phio(i,j+1)+phio(i-1,j)+phio(i,j-1))/4

         !print *,norm
         IF (abs(phio(i,j)-phi(i,j))<(10**-5)) exit timeloop
      end do
   end do

   phio=phi
end do timeloop
do i=1,Nx
   do j=1,Ny
      !write(1,*) phi(i,j)
      !print *,phi(i,j)
   end do
   !write(1,*)phi(i,:)
   !write(1,*)''
end do
 t_end = OMP_GET_WTIME()
 tot_time = t_end - t_start

end do
close(1)
!print *,nitr
end program
```