Project-1

- **Write a computer program to solve the heat conduction equation on a single block of steel**
  - k = 18.8 W/m K, ρ = 8000 kg/m^3, c_p = 500 J/(kg K)
  - Solve the transient heat conduction equation using the general, explicit control volume scheme on a non-uniform computational grid.
- **Use a computational grid:**
  - That is IMAX x JMAX in dimensions
    - Where IMAX=JMAX=101 AND IMAX=JMAX=501 (2 cases)
  - That has the following non-uniform distribution:

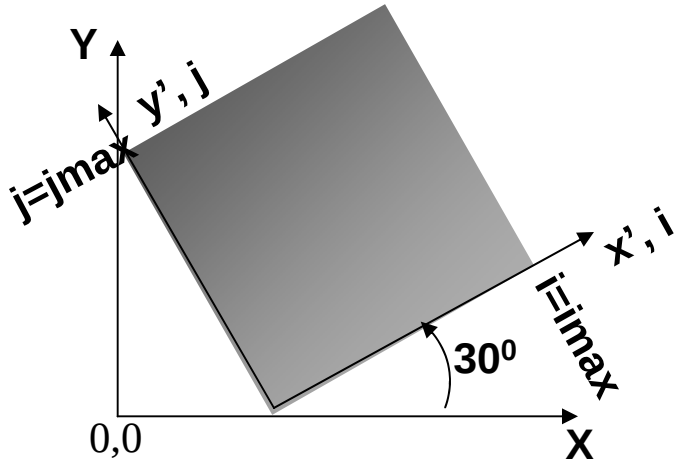
rot= 30.*3.141592654/180.

xp = cos[0.5*π(imax-i)/(imax-1)]

yp = cos[0.5*π(jmax-j)/(jmax-1)]

x(i,j) = xp*cos(rot)+(1.-yp)*sin(rot)

y(i,j) = yp*cos(rot)+xp*sin(rot)



Project-1

- **Find the temperature distribution on a 1m x 1m block with Dirichlet boundary conditions:**
  - T=5.[sin(πxp)+1.] at j=jmax
  - T=abs(cos(πxp))+1. at j=0
  - T=3.yp+2. at i=0 and i=imax

Note that xp and yp are used in these equations!

- **Iterate until the maximum residual magnitude drops below TOLER=1.0 x 10^{-5}**
  - Hint: The number of iterations will depend on the equation and the algorithm (~10,000 for 101 grid dimension)
- **Initialize the temperature field to a uniform value of 3.5**
- **Write out your converged solution to a (PLOT3D or equivalent) file that can be used for plotting or restart capability**
  - an example plot3d routine is on smartsite. Additional information can be found on the internet.

Project-1

- **Run your code on a cluster node that is not being used via qsub**
  - Compiling of code can only be performed on wopr

- **Use qsub to submit your job (that will automatically go to one of the nodes compute-0-0 through compute-0-6)**

Project-1

- **Due Wednesday October 16th (PAY CLOSE ATTENTION TO WHAT IS ASKED FOR!):**
  - Statement of problem, equations, and algorithm(s) used
  - Listing of Fortran or C code (You may program this and all projects in C if you prefer)
  - Output of run. Print out the iteration number, the magnitude of the maximum residual, and the indicies where the maximum residual was located
  - The CPU time from start of iteration to convergence

```
      integer clock_start,clock_end,clock_max,clock_rate
      real*4 wall_time
!     call system time to determine flow solver wall time
      call system_clock(count_max=clock_max,count_rate=clock_rate)
      call system_clock(clock_start)
!     determine total wall time for solver
      call system_clock(clock_end)
      wall_time=float(clock_end-clock_start)/float(clock_rate)
      print*,'solver wall clock time (seconds)',wall_time
```

  - Make sure that you link to the system library for the clock routines by adding –lrt during the linking step in your makefile
  - Plot of computational grid
  - Plot of converged temperature fields