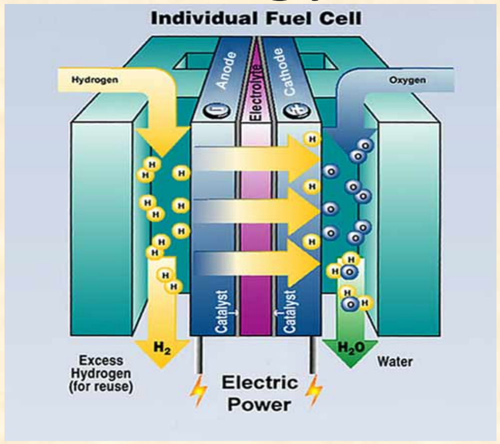
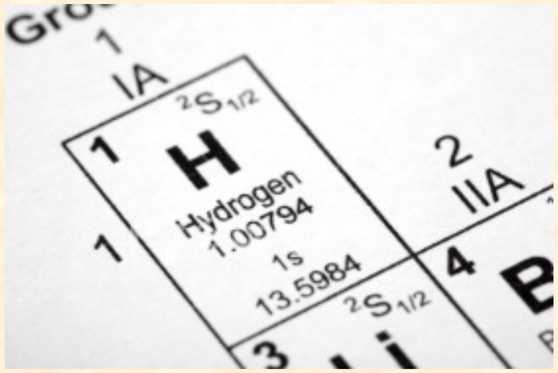
LONI Louisiana Optical Network Initiative

Hydrogen Saturation, Jacobian Iteration, and LA-SiGMA: The David Howe Story

Hydrogen Transport : Promises and Problems

- Hydrogen fuel cells promise large amounts of ***reliable and clean*** energy.





Hydrogen Transport : Promises and Problems

- While net-energy costs of hydrogen capture pose large hydrogen-switch barriers, ***safe and low cost*** transportation methods also must be developed

Hydrogen Transport : Promises and Problems

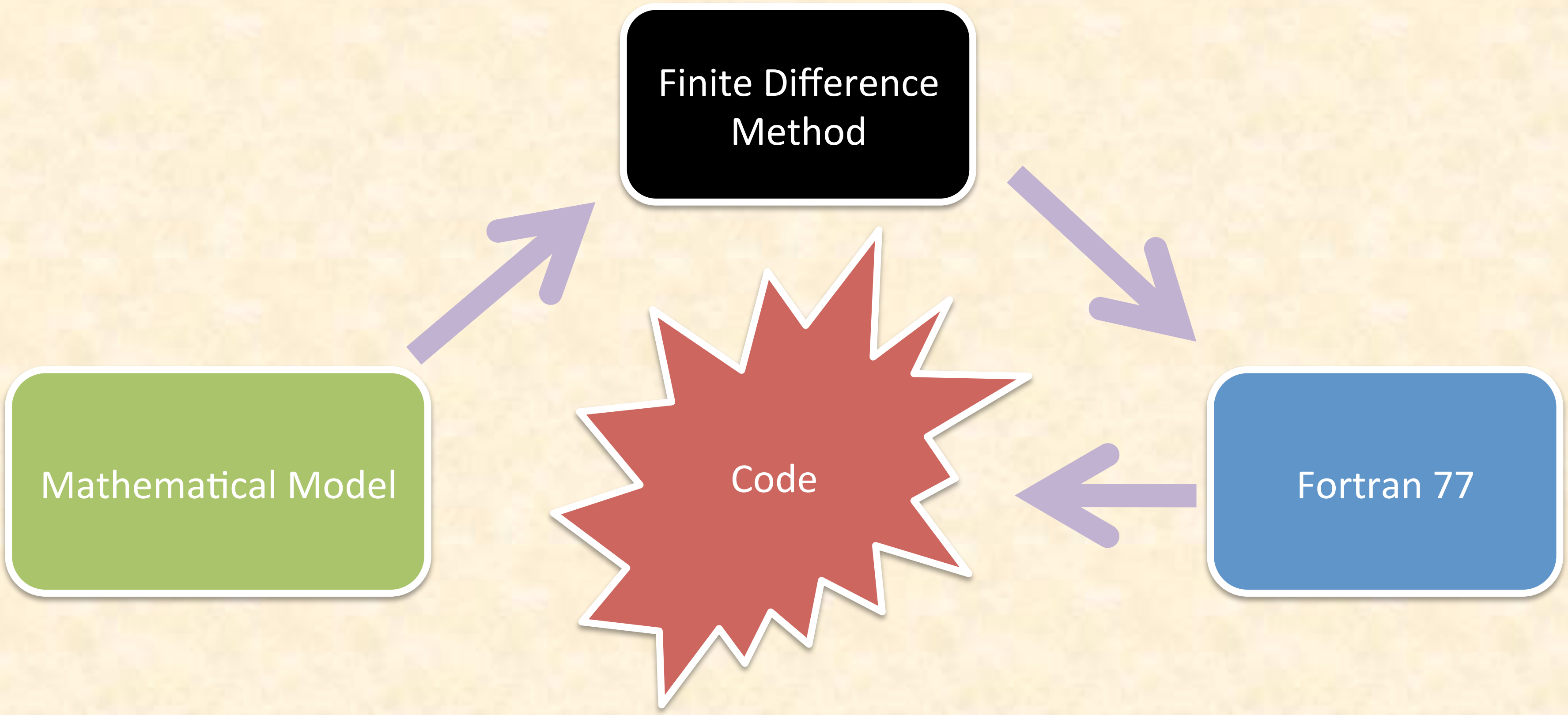
- A viable transportation method involves saturating metallic rods with hydrogen gas- most notably ***Lanthanum Nickel ($LaNi_5$)***

Absorbtion/Desorption at 300-325K

High $g_{hyrdogen}/g_{metal}$

Desirable pressure equilibrium

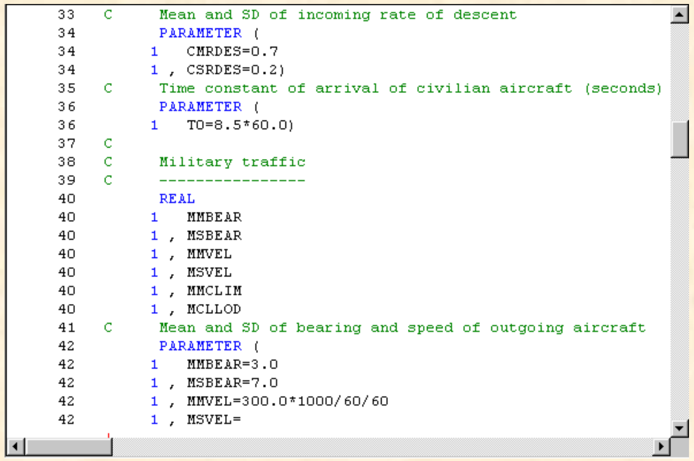
How can we know the specifics? How can we find even better options?



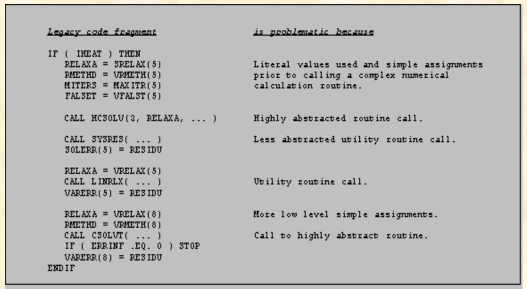
Hydrogen Desorption Modeling

- Using mathematical equations such as the heat equation and numerical methods such as ***Jacobian Iteration***, it was possible to formulate a program which modeled the desorption of hydrogen from $LaNi_5$ rods.

[illegible]



```
C     Mean and SD of incoming rate of descent
      PARAMETER (
     1   CMRDES=0.7
     1 , CSRDES=0.2)
C     Time constant of arrival of civilian aircraft (seconds)
      PARAMETER (
     1   TO=8.5*60.0)
C
C     Military traffic
C     ----------------
      REAL
     1   MMBEAR
     1 , MSBEAR
     1 , MMVEL
     1 , MSVEL
     1 , MMCLIM
     1 , MCLLOD
C     Mean and SD of bearing and speed of outgoing aircraft
      PARAMETER (
     1   MMBEAR=3.0
     1 , MSBEAR=7.0
     1 , MMVEL=300.0*1000/60/60
     1 , MSVEL=
```

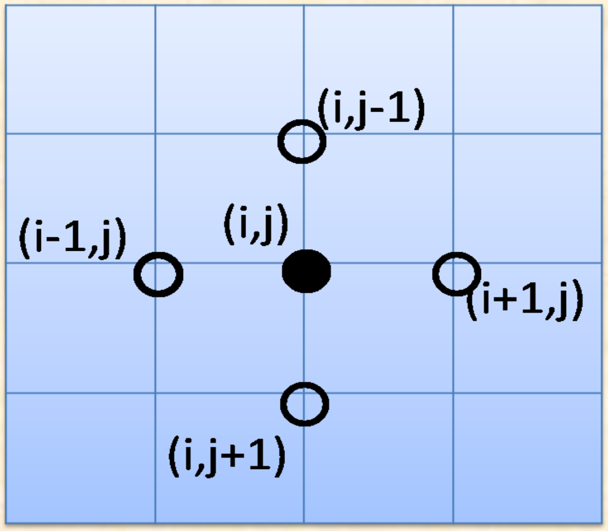


```
Legacy code fragment                 is problematic because

IF ( IHEAT ) THEN
   RELAXA = SRELAX(5)                Literal values used and simple assignments
   RMETHD = VRMETH(5)                prior to calling a complex numerical
   MITERS = MAXITR(5)                calculation routine.
   FALSET = VFALST(5)

   CALL HCSOLV(3, RELAXA, ... )      Highly abstracted routine call.

   CALL SYSRES( ... )                Less abstracted utility routine call.
   SOLERR(5) = RESIDU

   RELAXA = VRELAX(5)
   CALL LINRLX( ... )                Utility routine call.
   VARERR(5) = RESIDU

   RELAXA = VRELAX(8)                More low level simple assignments.
   RMETHD = VRMETH(8)
   CALL CSOLVT( ... )                Call to highly abstract routine.
   IF ( ERRINF .EQ. 0 ) STOP
   VARERR(8) = RESIDU
ENDIF
```

Figure 1 : Mixed levels of abstraction.

Laplace Equation

- In order to extrapolate interior conditions and multiple time-steps from a given set of initial conditions, a special method of Jacobian Iteration can be used. Given a mesh and Laplace's equation with boundary conditions…



$$\boxed{\Delta\varphi = 0}$$

$$0 \le y \le 1$$
$$u(0,y) = 50$$
$$u(1,y) = 0$$

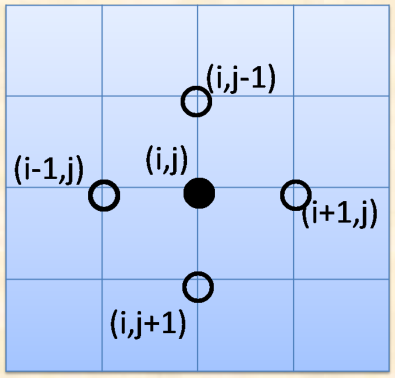
$$0 \le x \le 1$$
$$u(x,0) = \begin{matrix} 50 + 100x \\ 200(1-x) \end{matrix}$$
$$u(x,1) = 0$$

Finite Difference Method

- …and the following Laplacian obeying numerical scheme…

$$\boxed{u(x_{i+1}, y_j) = u(x_i + \Delta x, y_j) = u(x_i, y_j) + \Delta x \bullet \left.\frac{\partial u}{\partial x}\right|_{i,j} + \frac{\Delta x^2}{2} \bullet \left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} + \ldots}$$

$$\boxed{u(x_{i+1}, y_j) = u(x_i + \Delta x, y_j) = u(x_i, y_j) - \Delta x \bullet \left.\frac{\partial u}{\partial x}\right|_{i,j} + \frac{\Delta x^2}{2} \bullet \left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} - \ldots}$$



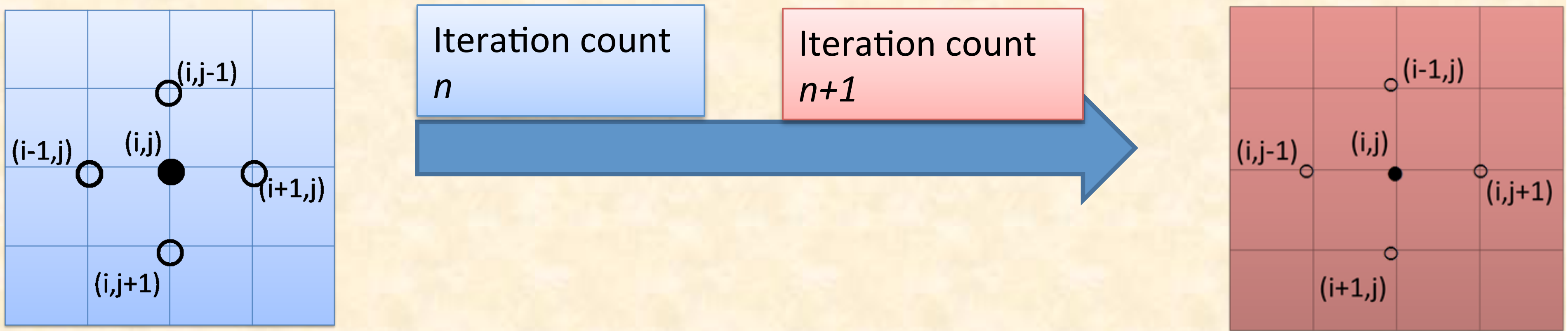
$$u(x_{i+1}, y_j) + u(x_{i-1}, y_j)$$

$$\boxed{\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + O(\Delta x^2)}$$

Finite Difference Scheme

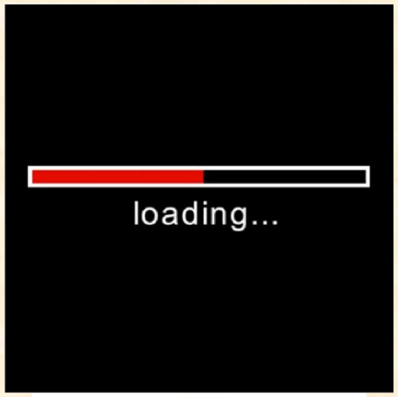
- …iteration over all mesh points gives the Jacobian form

$$\boxed{u_{i,j}^{n+1} = \frac{1}{4}\left(u_{i+1,j}^{n} + u_{i-1,j}^{n} + u_{i,j+1}^{n} + u_{i,j-1}^{n}\right)}$$



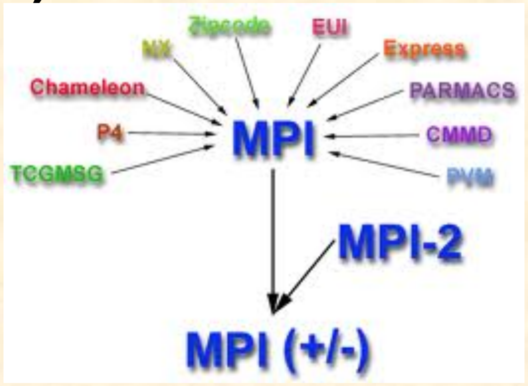
Jacobi, Fortran, and MPI : Formulation and Results

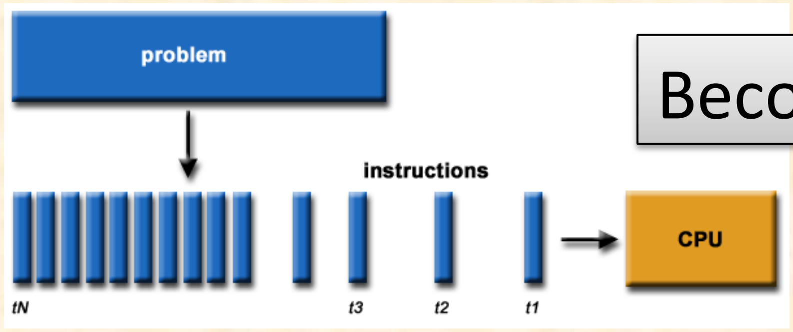
- This numerical technique is essential in computations within the hydrogen rod, but also takes up the ***vast majority of computing time***



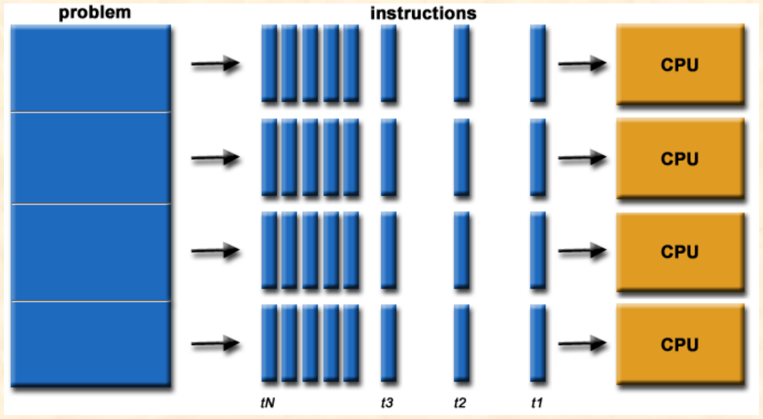
Jacobi, Fortran, and MPI : Formulation and Results

- In order to reduce the time needed for computation, ***MPI*** was introduced





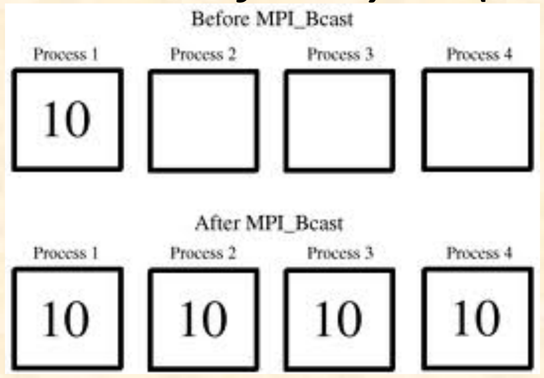
Becomes…

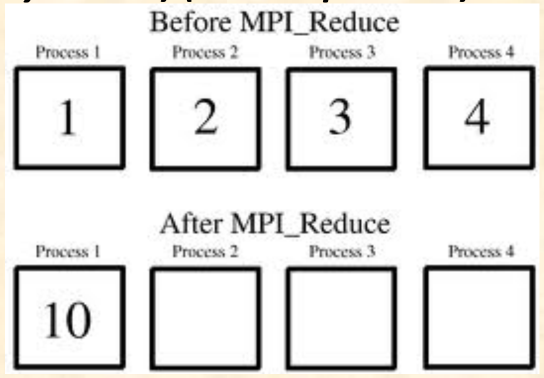


Jacobi, Fortran, and MPI : Formulation and Results

- Our Jacobian iteration program
  1. Ran a 10^4 x 10^4 mesh
  2. Incorporated MPI_Bcast and MPI_Reduce functionalities, as well as MPI_Isend and MPI_Recv for communication
  3. Utilized the following formalism to break up jobs between processors

DO j = myid(Nz/nprocs)+1 , (myid+1)(Nz/nprocs)*





```
do i=1,Nx
 do j=1,Ny
      un(i,j)=0.25d0*(uo(i+1,j)+uo(
+1)+uo(i,j-1))
        enddo
        enddo


       errmax=0.0d0
       do i=1,Nx
       do j=1,Ny
       if(errmax.LT.abs(uo(i,j)-un(i,j)))then
       errmax=abs(uo(i,j)-un(i,j))
       endif
       enddo
       enddo
       if(mod(counter,100)==0)
print*,counter,uo(1,30)
       if(errmax.LE.tol) goto 111
       do i=1,Nx
       do j=1,Ny
       uo(i,j)=un(i,j)
       enddo
       enddo
       counter=counter+1
       goto 22
```

Jacobian

Set Tolerance

Compute error

Replace previous array with new

Update iteration count

```
do j=myid*(Nx/nprocs)+1,(myid+1)*(Nx/nprocs)
do i=1,Ny
      un(i,j)=0.25d0*(uo(i+1,j)+uo(i-1,j)+uo(i,j+1)+uo(i,j-1))
      enddo
      enddo

      local_err=0.0d0
      global_err=0.0d0

      do j=myid*(Nx/nprocs)+1,(myid+1)*(Nx/nprocs)
      do i=1,Ny
      if(local_err.LT.abs(uo(i,j)-un(i,j)))then
      local_err=abs(uo(i,j)-un(i,j))
      endif
      enddo
      enddo
c       if(myid==0)then
c       do j=1,Nx
c       print*,uo(30,j),un(31,j)
c       enddo
c       endif

      call mpi_reduce(local_err,global_err,1,MPI_DOUBLE_PRECISION,
     &               MPI_MAX,0,mpi_comm_world,ierr)
c       if(myid==1.AND.mod(counter,100)==0)
c     &           print*,'before recv global_err=',counter,global_err
      call mpi_bcast(global_err,1,mpi_double_precision,0,
     &   mpi_comm_world,ierr)
      if(myid==0.AND.mod(counter,100)==0)
     &       print*,'after recv global_err=',counter,uo(1,30)

c       if(mod(counter,100)==0.AND.myid==0)print*,counter,uo(2,2)
      if(global_err.LE.tol) goto 111
      do j=myid*(Nx/nprocs)+1,(myid+1)*(Nx/nprocs)
      do i=1,Ny
      uo(i,j)=un(i,j)
      enddo
      enddo
if(nprocs.gt.1)then
      if(myid.ne.0)then
       call mpi_isend(uo(1,myid*(Nx/nprocs)+1),Ny,mpi_double_precision,
     &               myid-1,11,mpi_comm_world,request,ierr)

       call mpi_recv(uo(1,myid*(Nx/nprocs)),Ny,mpi_double_precision,
     &               myid-1,12,mpi_comm_world,status,ierr)
       call mpi_wait(request,status,ierr)

      endif

      if(myid.ne.(nprocs-1))then
       call mpi_isend(uo(1,(myid+1)*(Nx/nprocs)),Ny,
     &     mpi_double_precision,myid+1,12,mpi_comm_world,request,ierr)

       call mpi_recv(uo(1,(myid+1)*(Nx/nprocs)+1),Ny,
     &     mpi_double_precision,myid+1,11,mpi_comm_world,status,ierr)
       call mpi_wait(request,status,ierr)
      endif
      endif
      counter=counter+1
C       if(counter==10000)goto 11
      goto 22
C       goto 22
```

Array Breakup

Jacobian

Set Tolerance

Gather all local errors and select for largest

Replace previous array with new

Communication

Update iteration count

Jacobi, Fortran, and MPI : Formulation and Results

- Running on the LONI supercomputer at the Queen Bee cluster, the program was executed on 1,2,4,8,16,32,64, and 128 processors

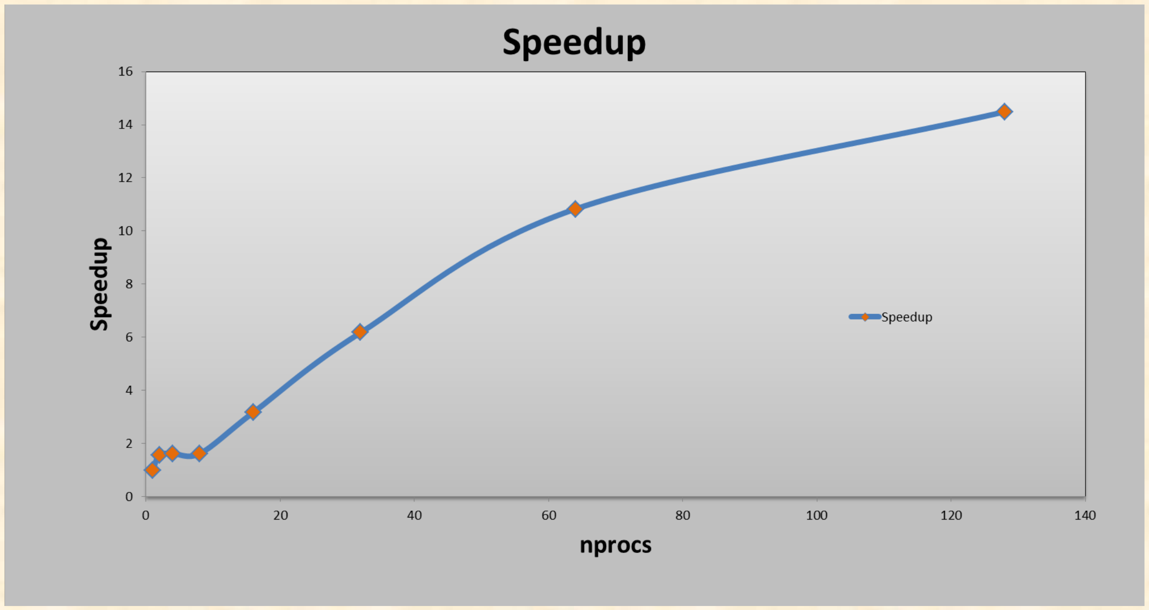


Fig. 1) Number of Processors vs. the relative speedup

Speedup was capable of reaching 14.47x with 128 processors compared with single processor runs. Serial trials ran for 10 hours and 15 minutes whereas 128 processors needed only 42 minutes.

Jacobi, Fortran, and MPI : Formulation and Results

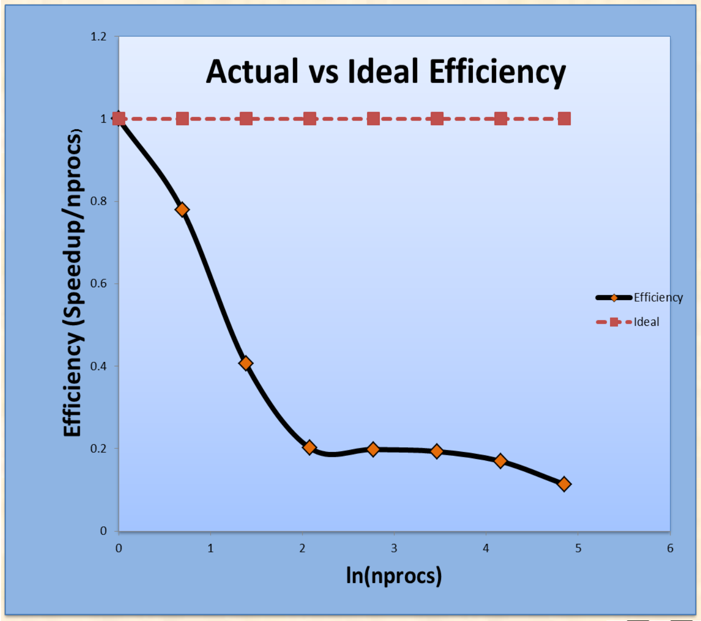


Fig. 2) Natural log of Number of Processors vs. Efficiency. Red line depicts ideal utilization of all processors used.

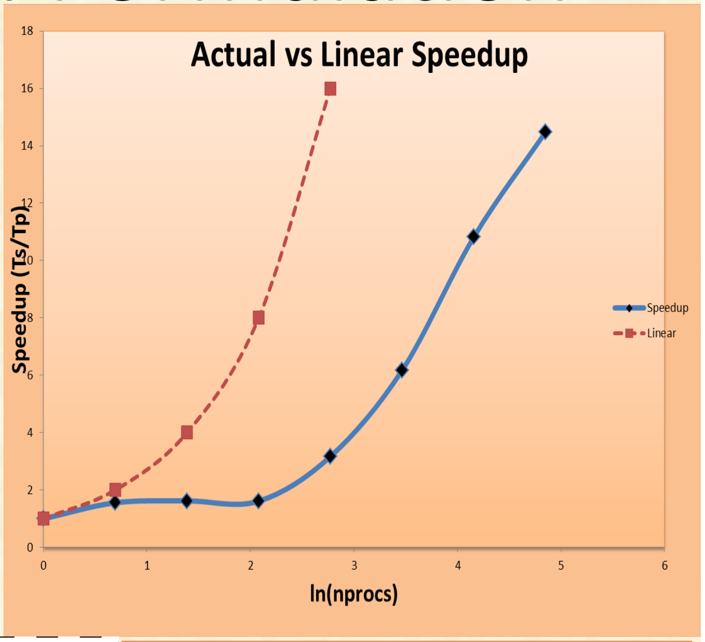
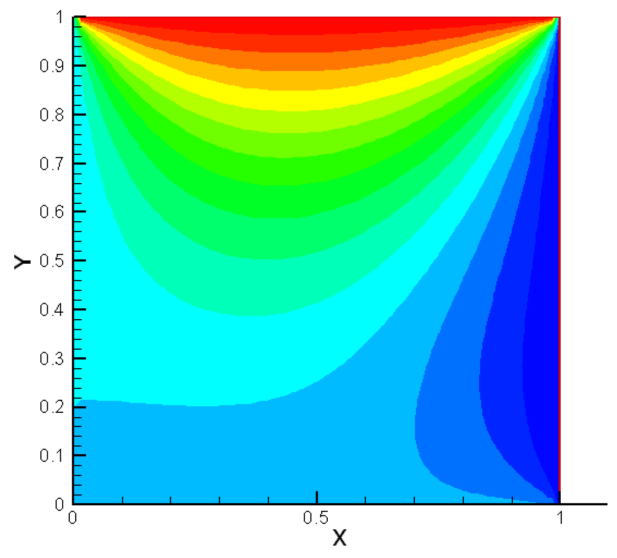


Fig. 3) Natural log of number of processors vs. relative speedup. Red line depicts ideal utilization of all processors used .

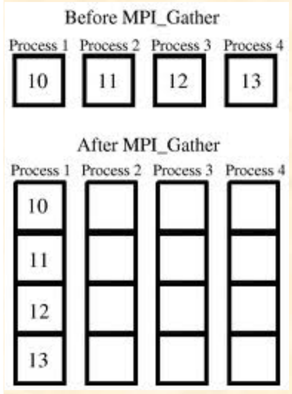


Hydrogen Desorption Modeling: Revisited

- Serial versions of the hydrogen desorption model (which use Jacobian iteration techniques extensively) took ~32 hours to complete.
  - If MPI could be correctly implemented into this program, wall-times could be reduced to ***~2 hours***

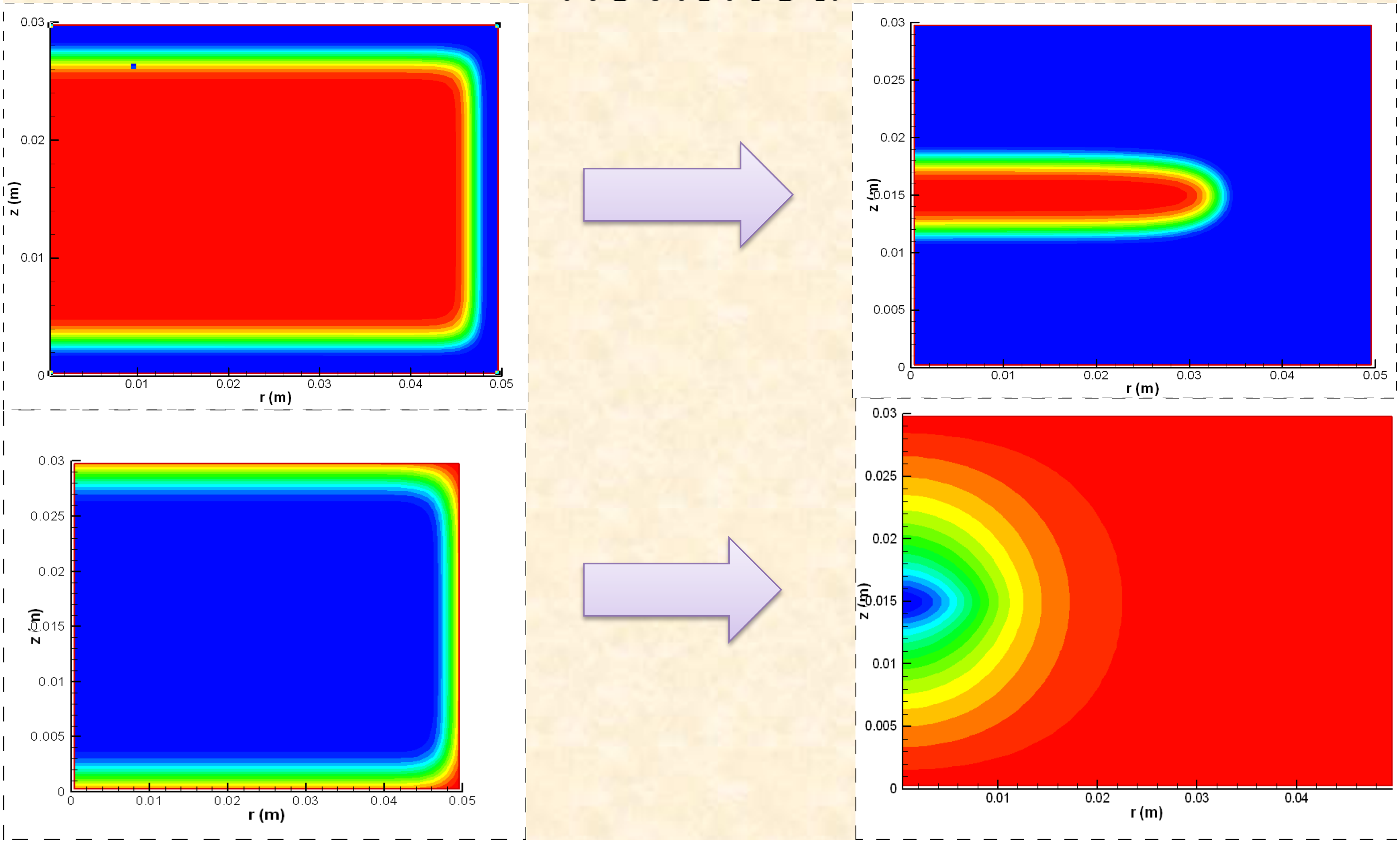
Hydrogen Desorption Modeling: Revisited

- MPI implementation to this program
  1. Ran fifteen 40x40 arrays, 5 of which required Jacobian iteration techniques to compute
  2. Iterated through 10,800,000 time-steps
  3. Utilized MPI_Bcast,MPI_Reduce, MPI_Gather, and MPI_Barrier functions
  4. Utilized the following scheme to break jobs between processors



```
if (myid .eq. 0) then  j=2,Nz/nprocs
else if (myid .eq. nprocs-1) then
            do j=(Nz/nprocs)*(nprocs-1)+1,Nz-1
Else do j=myid*(Nz/nprocs)+1,(myid+1)*(Nz/nprocs)
```

Hydrogen Desorption Modeling: Revisited



The Future of Hydrogen Desorption Modeling

- Once MPI techniques are successfully implemented into models for hydrogen desorption from $LaNi_5$, modifications to the structure of said compound can be tested readily and with no extra experimental costs

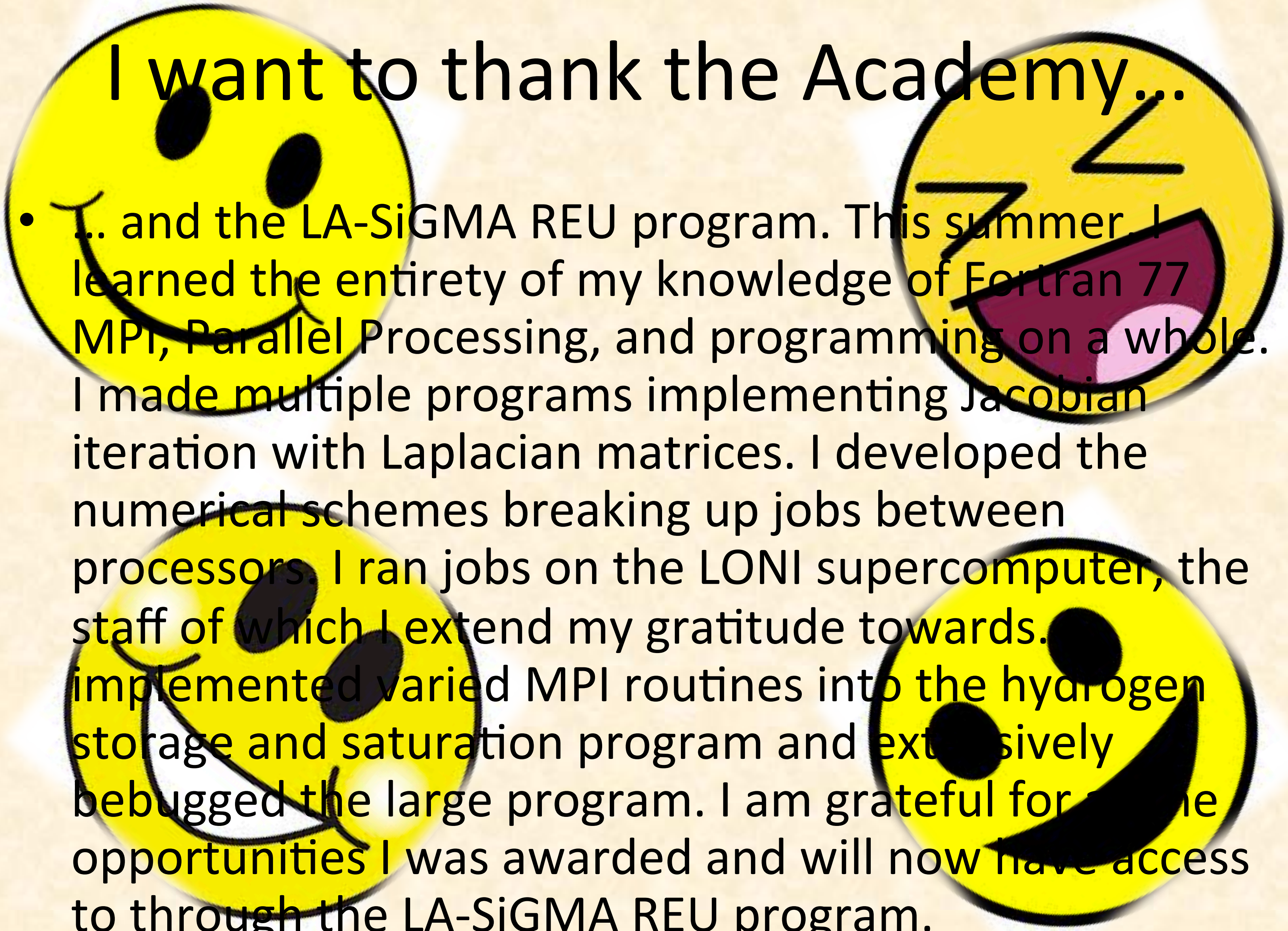
Mg?

Mg_2Ni?

$LaNi_5$?

FeTi?

Dopants?



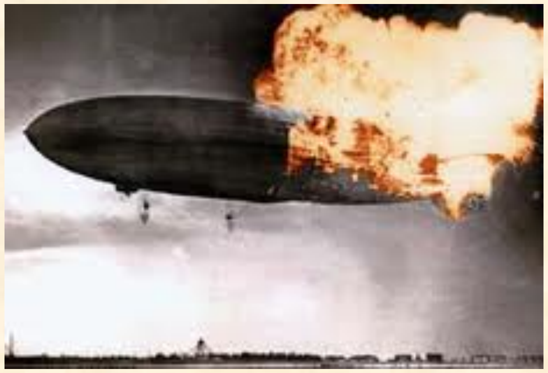
I want to thank the Academy…

- … and the LA-SiGMA REU program. This summer, I learned the entirety of my knowledge of Fortran 77 MPI, Parallel Processing, and programming on a whole. I made multiple programs implementing Jacobian iteration with Laplacian matrices. I developed the numerical schemes breaking up jobs between processors. I ran jobs on the LONI supercomputer, the staff of which I extend my gratitude towards. I implemented varied MPI routines into the hydrogen storage and saturation program and extensively bebugged the large program. I am grateful for all the opportunities I was awarded and will now have access to through the LA-SiGMA REU program.

Questions and Farewell

- Hope all your brains feels like…

- And not like…



This material is based upon work supported by the National Science Foundation under the NSF EPSCoR Cooperative Agreement No. EPS-1003897 with additional support from the Louisiana Board of Regents.