

# **The DOE ACTS Collection**

## ***Interfaces, Functionality and Interoperability***

**Fifth DOE ACTS Collection Workshop**  
***Enabling Technologies for High End Computer Simulations***



**Tony Drummond**

**Lawrence Berkeley National Laboratory**

**acts-support@nersc.gov**

**LADrummond@lbl.gov**



# Motivation

*Grand Challenges are ..fundamental problems in science and engineering, with potentially broad social, political, and scientific impact, that could be advanced by applying high performance computer resources*

Office of Science and Technology

- Some grand challenges: electronic structure of materials, turbulence, genome sequencing and structural biology, global climate modeling, speech and language studies, pharmaceutical design, pollution, etc. .



# Motivation

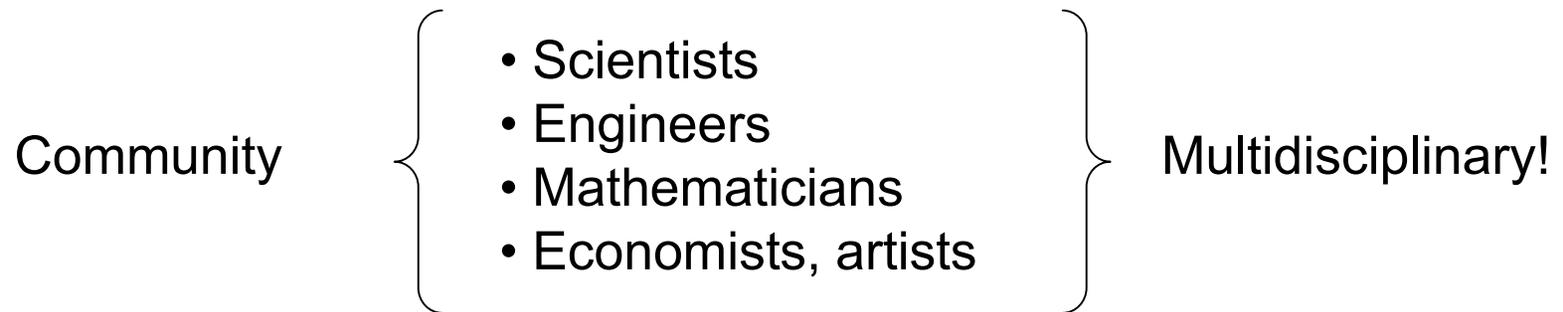
*With the development of new kinds of equipment of greater capacity, and particularly of greater speed, it is almost certain that new methods will have to be developed in order to make the fullest use of this equipment. It is necessary not only to design machines for the mathematics, but also to develop a new mathematics for the machines - 1952, Hartree*

- **Metropolis** (Monte Carlo Methods) grew out of physical chemistry in 1950's through attempts to calculate statistical properties of chemical reactions. Some areas of application: astrophysics, many areas engineering, and chemistry
- **Fast Fourier Transform (FFT)**: implementation of Fourier Analysis. Some areas of application: image and signal processing, seismology, physics, radiology, acoustics and engineering
- **Multigrids**: method for solving a wide variety of PDE. Some areas of application: physics, biophysics and engineering



# Motivation

Computational science: can be characterized by the needs to gain understanding through the analysis of mathematical models using high performing performing computers



## Computer Science

Provides services ranging from Networking and visualization tools to algorithms

## Mathematics:

Credibility of algorithms (error analysis, exact solutions, expansions, uniqueness proofs and theorems)

# Some lessons learned from Earth System Modeling



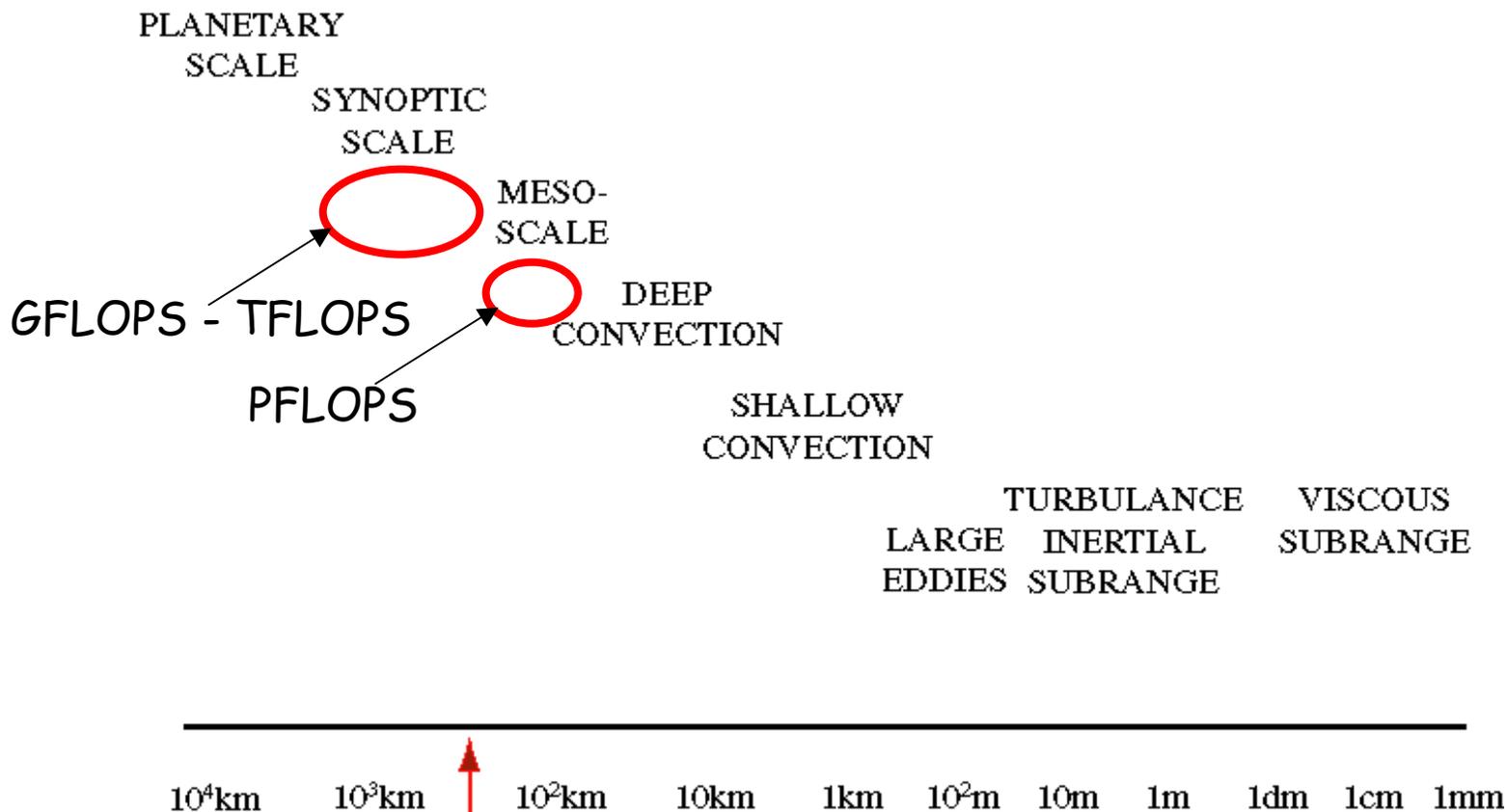
*The DOE ACTS Collection (<http://acts.nersc.gov>)*

*02/24/2004*

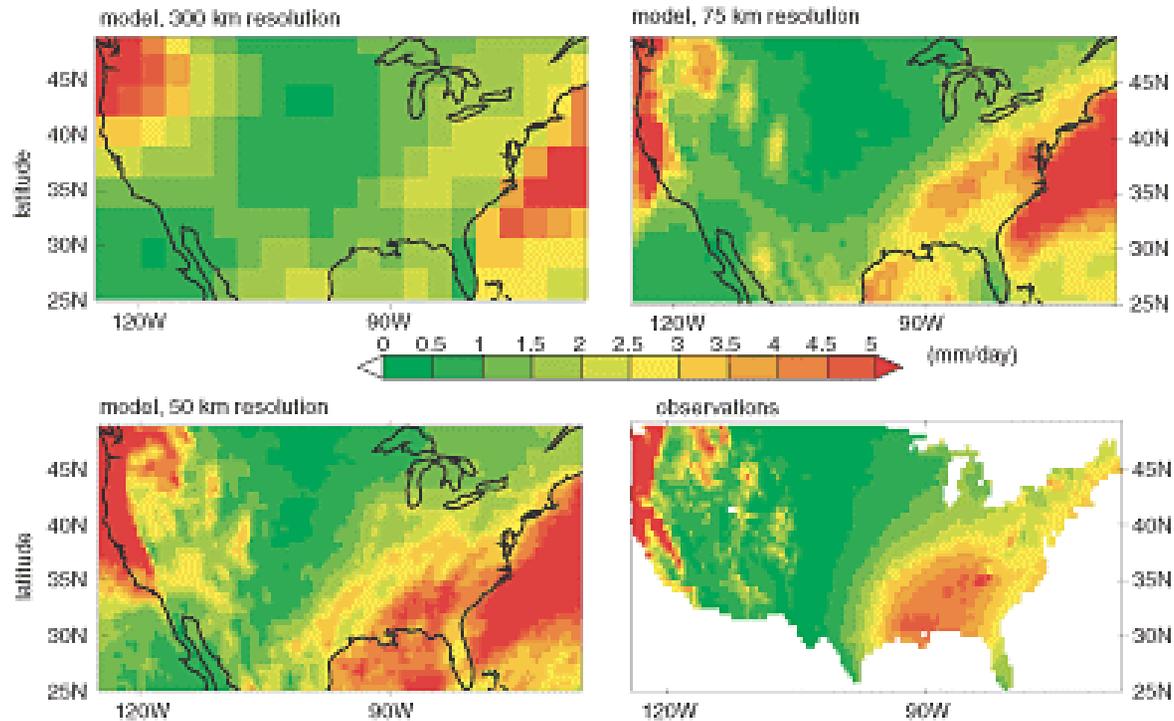
*5*

# Motivation - Example I

## SPECTRUM OF ATMOSPHERIC PHENOMENA



# Motivation - Example I

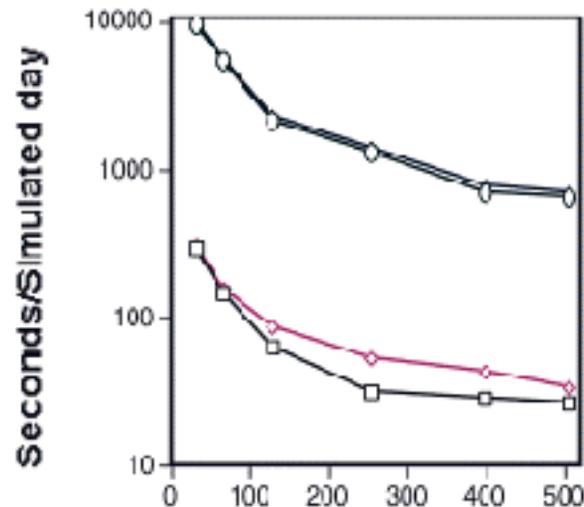


Duffy et. al.,  
Lawrence Livermore National Laboratory

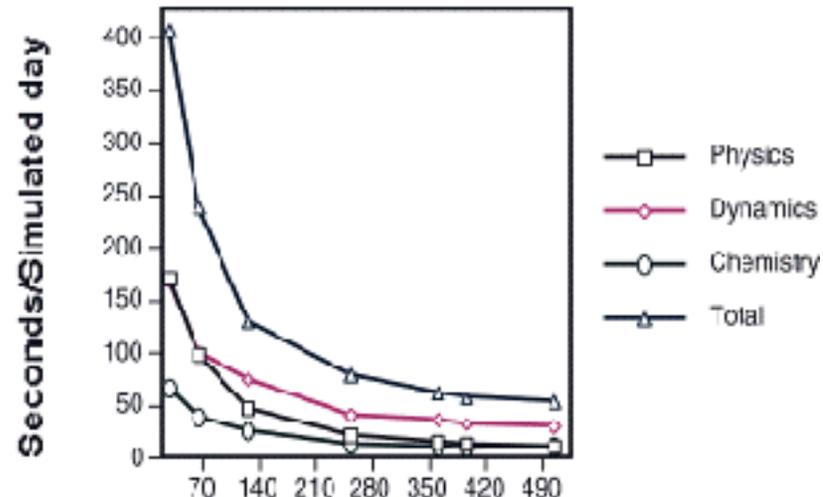
- CCM3 - spectral truncations of T170 and T239
- 50 Km spatial resolution is 32 times more grid cells and takes roughly **200 times longer** to run

# Motivation - Example II

## Coupled Atmosphere/Atmospheric Chemistry



AGCM/ACM  
2.5 long x 2 lat, 30 layers  
25-chemical species



AGCM/ACM  
2.5 long x 2 lat, 30 layers  
2-chemical species

- Non-linear demand for resources (CPU - Memory)
- Multi-disciplinary application is more demanding

## Partial Matrix of Methods and Disciplines

	Climate Change	Material Science	High Enregy Physics	Astrophysics Cosmology	Biology	Chemistry	Fusion
Monte Carlo (Quantum and Classical)	PCM CCSM POP	Quantum MC Classical KMC	FASTER SYNPOL	FASTER SYNPOL		NWChem	
Fast Fourier Transform		VASP Paratec Petot Escan	IMPACT LANGEVIN3D MAD9P ccSHT		SPIDER NAMD	NAMD	WARP GTC
Fast Multipole & Variants		Classical MD	IMPACT LANGEVIN3D QuickPIC		Classical MD	NWChem Classical MD	
Sparse Linear systems	PCM CCSM POP	O(N) Methods	OMEGA3P		SPIDER	pVarDen	NIMROD
Eigenvalue Solvers		DFT FLAPW PW codes	OMEGA3P		DFT SPIDER	NWChem Gaussian QChem	
Dense Linear Solvers		LSMS FLAPW		MADCAP		NWChem Gaussian	GTC
Adaptive Mesh Refinement	BoxLib Paramesh		BoxLib Paramesh	FLASH Paramesh		pVarDen BoxLib	WARP BOX Chombo

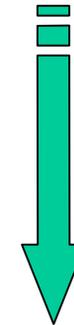
## Partial Matrix of Methods and Disciplines

	Climate	Material	High Enregy	Astrophysics	Biology	Chemistry	Fusion	
Monte Carlo (Quantum Classical)	<b>NWChem: Uses GlobalArrays for managing the distributed computing environments</b>					NWChem		
Fast Fourier Transform		VASP Paratec Petot	IMPACT LANGEVIN3D		SPIDER NAMD	NAMD	WARP GTC	
Fast Multipole & Variants		<b>MADCAP: Uses ScaLAPACK for the solution of large and dense linear systems of equations</b>					em cal	
Sparse Linear systems	PCM CCSM POP	O(N) Methods	OMEGA3P		SPIDER	pVarDen	NIMROD	
Eigenvalue Solvers		DFT FLAPW PW codes	<b>FLAPW: uses ScaLAPACK for the solutions Dense Eigenvalue Problems</b>					NWChem
Dense Linear Solvers		LSMS FLAPW						
Adaptive Mesh Refinement	BoxLib Paramesh		BoxLib Paramesh	FLASH Paramesh		pVarDen BoxLib	WARP BOX Chombo	

# Matrix of Scientific Applications and Tools

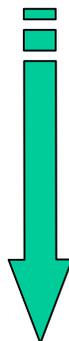
Application	Computational Problem	Software Tools	Highlights
MADCAP	Matrix factorization and triangular solves	ScaLAPACK	<ul style="list-style-type: none"> <li>• 50% peak performance on an IBM SP</li> <li>• Nearly perfect scalability on 1024, 2048, 3072 and 4096 processors</li> <li>• Fast implementation of numerical algorithms</li> </ul>
3-Charged Particles	Solution of large, complex unsymmetric linear systems	SuperLU	<ul style="list-style-type: none"> <li>• Solves systems of equations of order 8.4 million on 64 processors in 1 hour of wall clock time</li> <li>• 30 GFLOPs</li> </ul>
NWChem	Distribute large data arrays, collective operations	Global Arrays and LAPACK	<ul style="list-style-type: none"> <li>• Very good scaling for large problems</li> </ul>

<http://acts.nersc.gov/AppMat>



*Enabling sciences and discoveries... with high performance and scalability...*

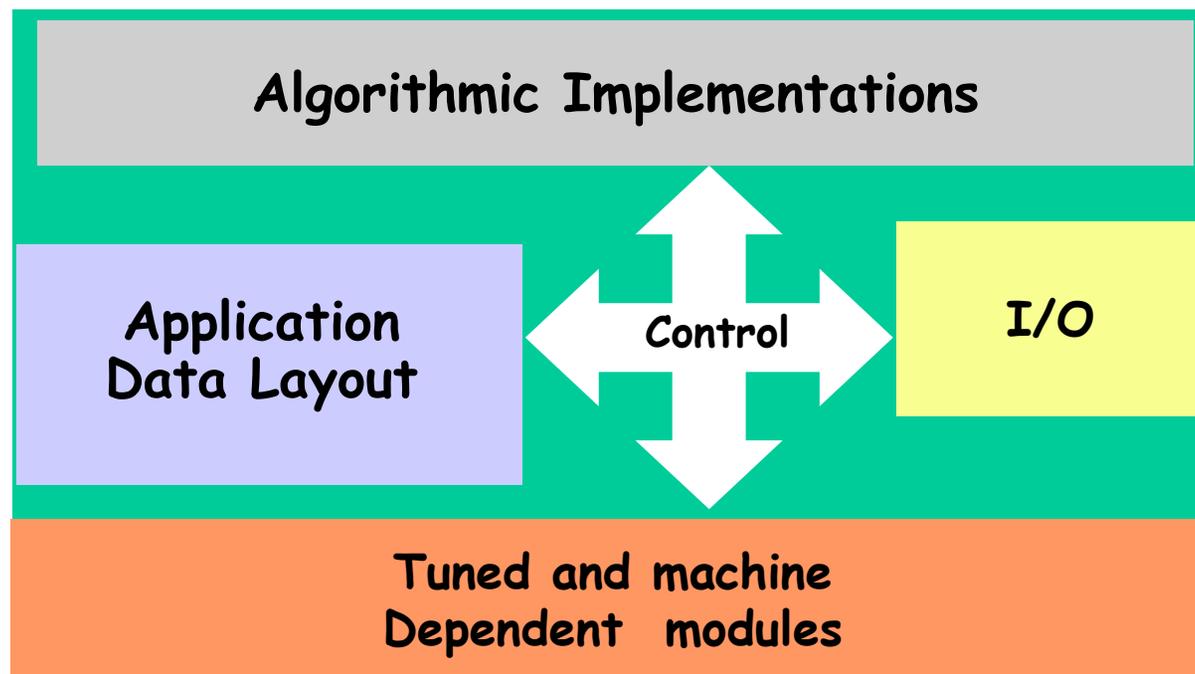
... More Applications ...



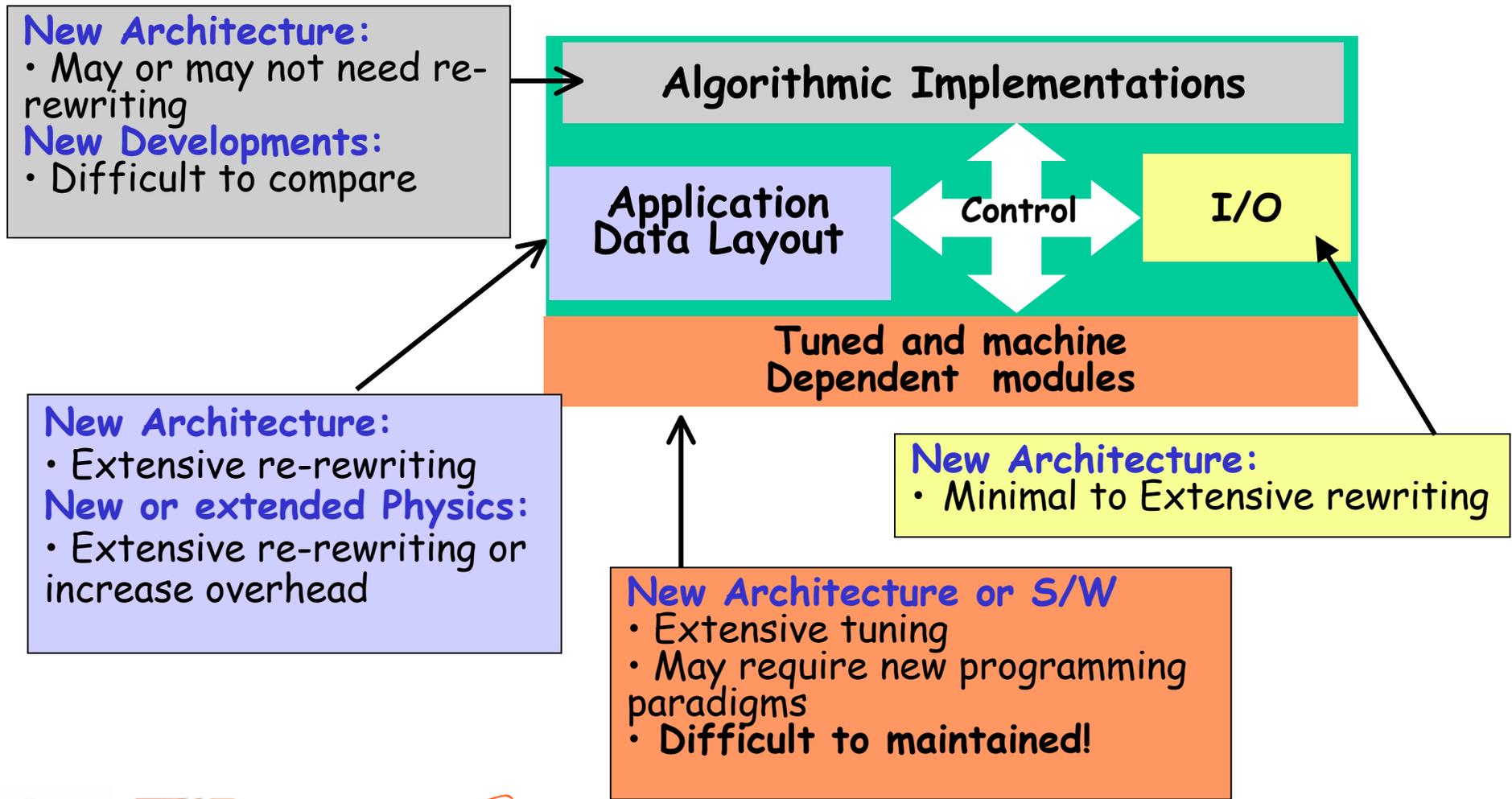
FDN3D &	Unstructured grids, compressible and incompressible Euler and Navier-Stokes equations.	PETSc	<ul style="list-style-type: none"> <li>• Parallelization of legacy code</li> <li>• Gordon Bell price, 0.23 Tflop/s on 3072 procs of ASCI Red</li> </ul>
P-FLAPW	Eigenvalue problems	ScaLAPACK	<ul style="list-style-type: none"> <li>• Study of systems up to 700 atoms (mat size=35,000)</li> <li>• Runs efficiently</li> <li>• Facilitated the study of new problems in materials science such as impurities and disordered systems.</li> </ul>
NIMROD	Quad and triangular high order finite elements, semi-implicit time integration, sparse matrix solvers	SuperLU	<ul style="list-style-type: none"> <li>• Code improvement of 5 fold, equivalent to 3-5 years progress in computing hardware.</li> </ul>

# Motivation- Why do we need software libraries?

## Large Scientific Codes: *A Common Programming Practice*



# Motivation- Why do we need software libraries?

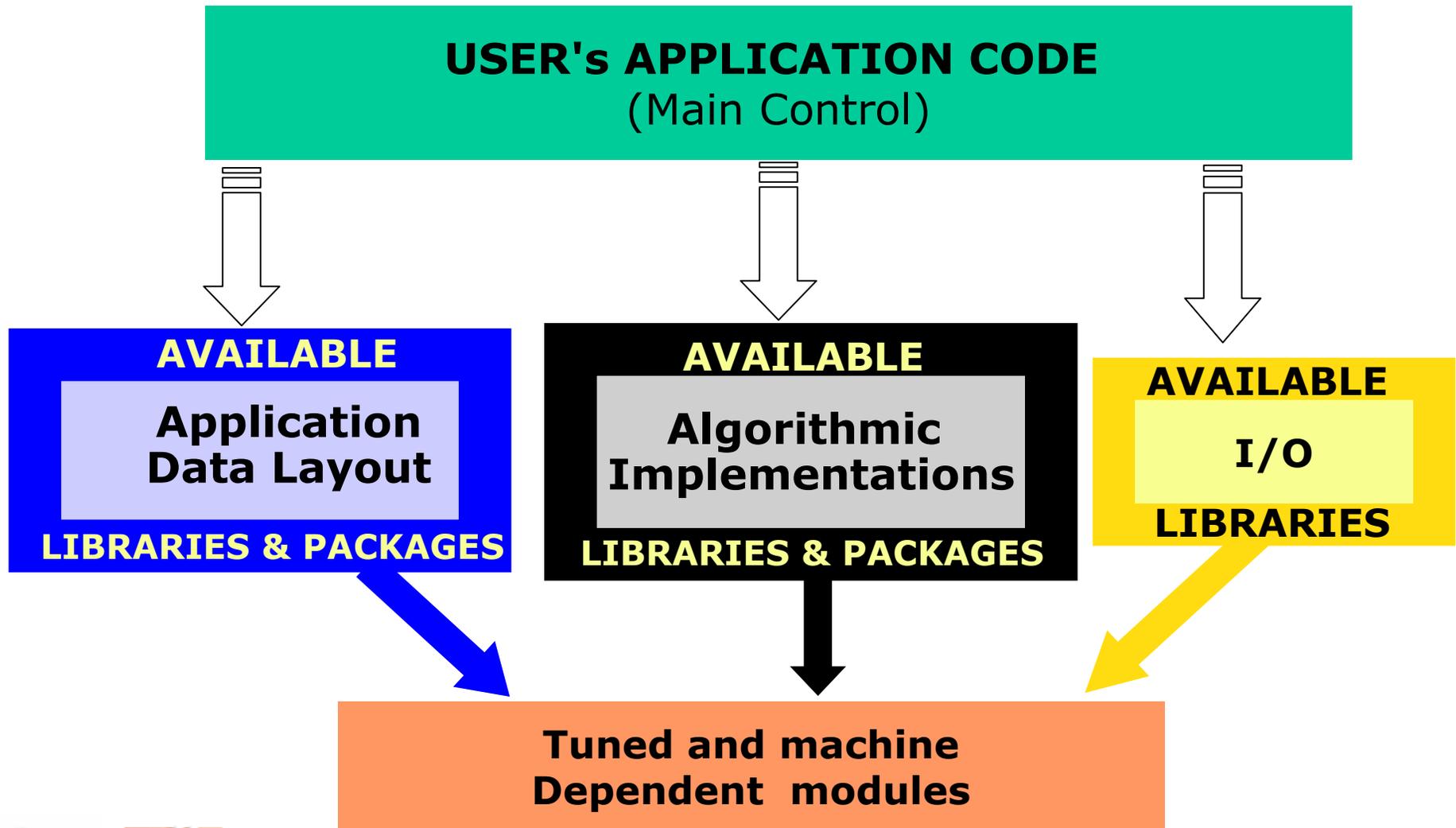


## Some options for New Architectures

OPTION	SOFTWARE IMPACT	COST	TIMELINE
Modification of commodity processors	Minimal	2 or 3 times commodity?	Can be achieved in a few years
U.S.-made vector architecture	Moderate	2 or 3 times commodity at present	Available now
Processor-in-memory (Blue Gene/L)	Extensive	Unknown, 2 to 5 times commodity?	Only prototypes available now
Japanese- made vector architecture	Moderate	2.5 to 3 times commodity at present	Available now
Research Architectures (Streams, VIRAM..)	Extensive or unknown	Unknown	Academic research prototypes only available now

# Motivation- Why do we need software libraries?

## An Alternative Approach

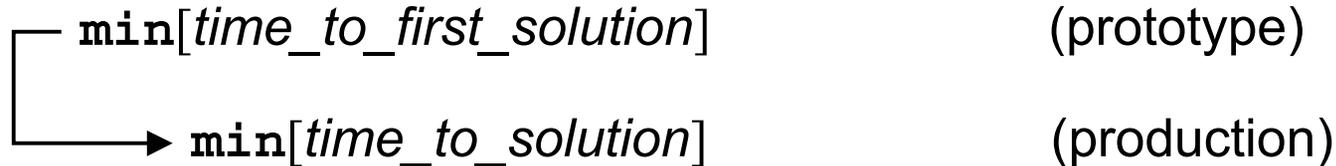


# Old Fashion

**"We need to move away from a coding style suited for serial machines, where every macrostep of an algorithm needs to be thought about and explicitly coded, to a higher-level style, where the compiler and library tools take care of the details. And the remarkable thing is, if we adopt this higher-level approach right now, even on today's machines, we will see immediate benefits in our productivity."**

W. H. Press and S. A. Teukolsky, 1997  
*Numerical Recipes: Does This Paradigm Have a future?*

# Main Objective of Using a Library



- Outlive Complexity
  - Increasingly sophisticated models
  - Model coupling
  - Interdisciplinary(Software Evolution)
- Sustained Performance
  - Increasingly complex algorithms
  - Increasingly complex architectures
  - Increasingly demanding applications(Long-term deliverables)

# Software Interfaces

```
CALL BLACS_GET( -1, 0, ICTXT )
CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL )
:
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
:
:
CALL PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB,
            $          INFO )
```

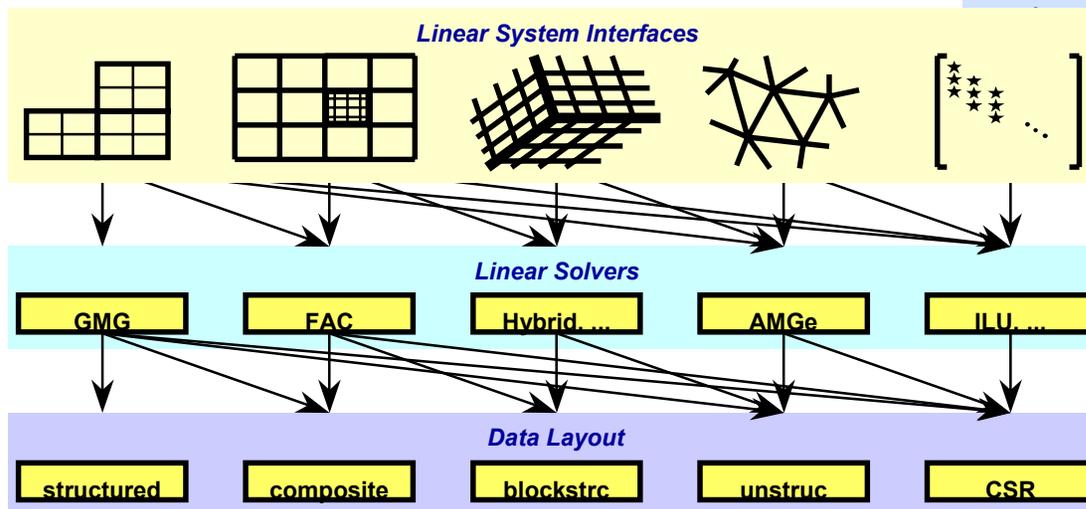
## Language Calls

## Command lines

- -ksp\_type [cg,gmres,bcgs,tfqmr,...]
- -pc\_type [lu,ilu,jacobi,sor,asm,...]

*More advanced:*

- -ksp\_max\_it <max\_iters>
- -ksp\_gmres\_restart <restart>
- -pc\_asm\_overlap <overlap>
- -pc\_asm\_type [basic,restrict,interpolate,none]



## Problem Domain

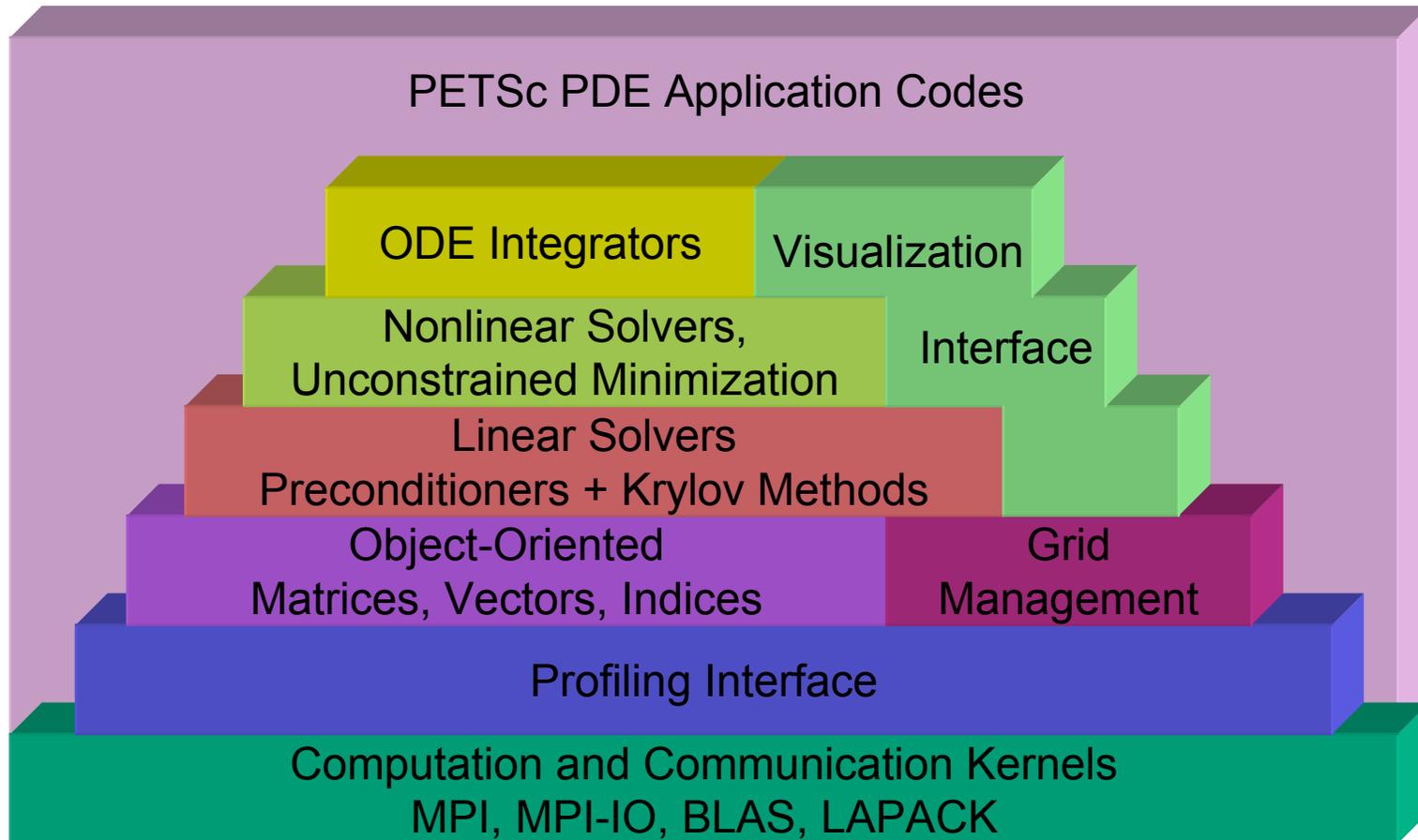
# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations	Direct Methods	LU Factorization	ScaLAPACK(dense) SuperLU (sparse)
		Cholesky Factorization	ScaLAPACK
		LDL <sup>T</sup> (Tridiagonal matrices)	ScaLAPACK
		QR Factorization	ScaLAPACK
		QR with column pivoting	ScaLAPACK
		LQ factorization	ScaLAPACK

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations ( <i>cont..</i> )	Iterative Methods	Conjugate Gradient	AztecOO (Trilinos) PETSc
		GMRES	AztecOO PETSc Hypre
		CG Squared	AztecOO PETSc
		Bi-CG Stab	AztecOO PETSc
		Quasi-Minimal Residual (QMR)	AztecOO
		Transpose Free QMR	AztecOO PETSc

# Structure of PETSc



## PETSc Example - Basic Linear Solver in C

```
SLES sles;          /* linear solver context */
Mat  A;            /* matrix */
Vec  x, b;         /* solution, RHS vectors */
int  n, its;       /* problem dimension, number of iterations */

MatCreate(MPI_COMM_WORLD,PETSC_DECIDE,PETSC_DECIDE,
          n,n,&A); /* assemble matrix */
VecCreate(MPI_COMM_WORLD,PETSC_DECIDE,n,&x);
VecDuplicate(x,&b); /* assemble RHS vector */

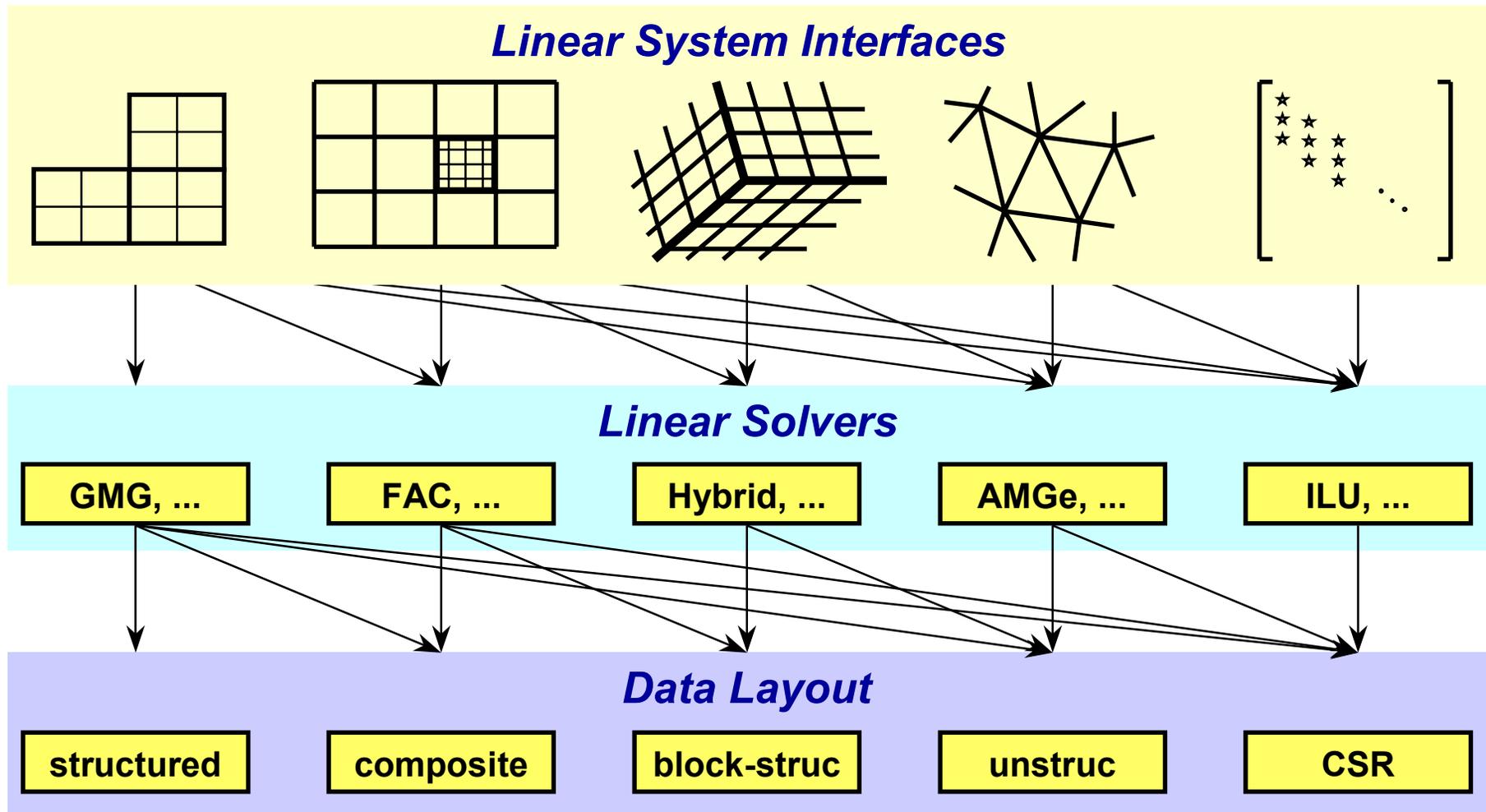
SLESCreate(MPI_COMM_WORLD,&sles);
SLESSetOperators(sles,A,A,DIFFERENT_NONZERO_PATTERN);
SLESSetFromOptions(sles);
SLESSolve(sles,b,x,&its);
SLESDestroy(sles);
```

*Original Matrix from  
Linear System  
Preconditioning matrix*

**A**

SAME\_NON\_ZERO\_PATTERN  
SAME\_PRECONDITIONER

# Hypre Conceptual Interfaces



# INTERFACE TO SOLVERS

List of Solvers and Preconditioners per Conceptual Interface

Solvers	System Interfaces			
	Struct	SStruct	FEI	IJ
Jacobi	X			
SMG	X			
PFMG	X			
BoomerAMG	X	X	X	X
ParaSails	X	X	X	X
PILUT	X	X	X	X
Euclid	X	X	X	X
PCG	X	X	X	X
GMRES	X	X	X	X

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations ( <i>cont..</i> )	Iterative Methods ( <i>cont..</i> )	SYMMLQ	PETSc
		Precondition CG	AztecOO PETSc Hypre
		Richardson	PETSc
		Block Jacobi Preconditioner	AztecOO PETSc Hypre
		Point Jacobi Preconditioner	AztecOO
		Least Squares Polynomials	PETSc

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithms	Library
Systems of Linear Equations ( <i>cont..</i> )	Iterative Methods ( <i>cont..</i> )	SOR Preconditioning	PETSc
		Overlapping Additive Schwartz	PETSc
		Approximate Inverse	Hypre
		Sparse LU preconditioner	AztecOO PETSc Hypre
		Incomplete LU (ILU) preconditioner	AztecOO
		Least Squares Polynomials	PETSc
	MultiGrid (MG) Methods	MG Preconditioner	PETSc Hypre
		Algebraic MG	Hypre
		Semi-coarsening	Hypre

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Linear Least Squares Problems	Least Squares	$\min_x \  b - Ax \ _2$	ScaLAPACK
	Minimum Norm Solution	$\min_x \  x \ _2$	ScaLAPACK
	Minimum Norm Least Squares	$\min_x \  b - Ax \ _2$ $\min_x \  x \ _2$	ScaLAPACK
Standard Eigenvalue Problem	Symmetric Eigenvalue Problem	$Az = \lambda z$ For $A=A^H$ or $A=A^T$	ScaLAPACK (dense) SLEPc (sparse)
Singular Value Problem	Singular Value Decomposition	$A = U\Sigma V^T$ $A = U\Sigma V^H$	ScaLAPACK (dense) SLEPc (sparse)
Generalized Symmetric Definite Eigenproblem	Eigenproblem	$Az = \lambda Bz$ $ABz = \lambda z$ $BAz = \lambda z$	ScaLAPACK (dense) SLEPc (sparse)

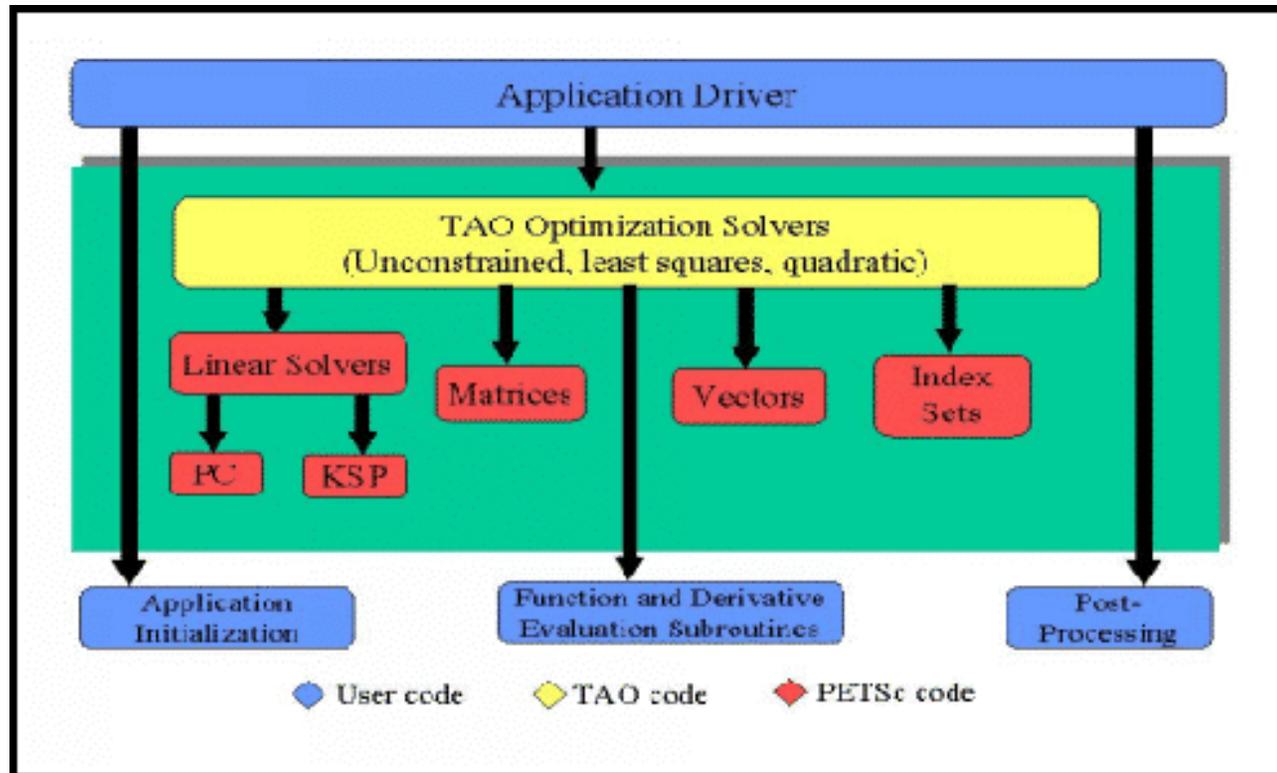
# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Non-Linear Equations	Newton Based	Line Search	PETSc
		Trust Regions	PETSc
		Pseudo-Transient Continuation	PETSc
		Matrix Free	PETSc

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization	Newton Based	Newton	OPT++ TAO
		Finite-Difference Newton	OPT++ TAO
		Quasi-Newton	OPT++ TAO
		Non-linear Interior Point	OPT++ TAO
	CG	Standard Non-linear CG	OPT++ TAO
		Limited Memory BFGS	OPT++
		Gradient Projections	TAO
	Direct Search	No derivate information	OPT++

# TAO - Interface with PETSc



# OPT++ Interfaces

- Four major classes of problems available
  - $NLF0(ndim, fcn, init\_fcn, constraint)$ 
    - Basic nonlinear function, no derivative information available
  - $NLF1(ndim, fcn, init\_fcn, constraint)$ 
    - Nonlinear function, first derivative information available
  - $FDNLF1(ndim, fcn, init\_fcn, constraint)$ 
    - Nonlinear function, first derivative information approximated
  - $NLF2(ndim, fcn, init\_fcn, constraint)$ 
    - Nonlinear function, first and second derivative information available

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization	Newton Based	Newton	OPT++ TAO
		Finite-Difference Newton	OPT++ TAO
		Quasi-Newton	OPT++ TAO
		Non-linear Interior Point	OPT++ TAO
	CG	Standard Non-linear CG	OPT++ TAO
		Limited Memory BFGS	OPT++
		Gradient Projections	TAO
	Direct Search	No derivate information	OPT++

# ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization (cont..)	Semismoothing	Feasible Semismooth	TAO
		Unfeasible semismooth	TAO
Ordinary Differential Equations	Integration	Adam-Moulton (Variable coefficient forms)	CVODE (SUNDIALS) CVODES
	Backward Differential Formula	Direct and Iterative Solvers	CVODE CVODES
Nonlinear Algebraic Equations	Inexact Newton	Line Search	KINSOL (SUNDIALS)
Differential Algebraic Equations	Backward Differential Formula	Direct and Iterative Solvers	IDA (SUNDIALS)

# ACTS Numerical Tools: *Functionality*

Computational Problem	Support	Techniques	Library
Writing Parallel Programs	Distributed Arrays	Shared-Memory	Global Arrays
		Distributed Memory	CUMULVS (viz) Globus (Grid)
		Grid Generation	OVERTURE
		Structured Meshes	CHOMBO (AMR) Hypr OVERTURE PETSc
		Semi-Structured Meshes	CHOMBO (AMR) Hypr OVERTURE
	Distributed Computing	GRID	Globus
		Remote Steering	CUMULVS
		Coupling	PAWS

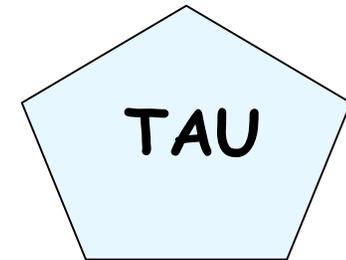
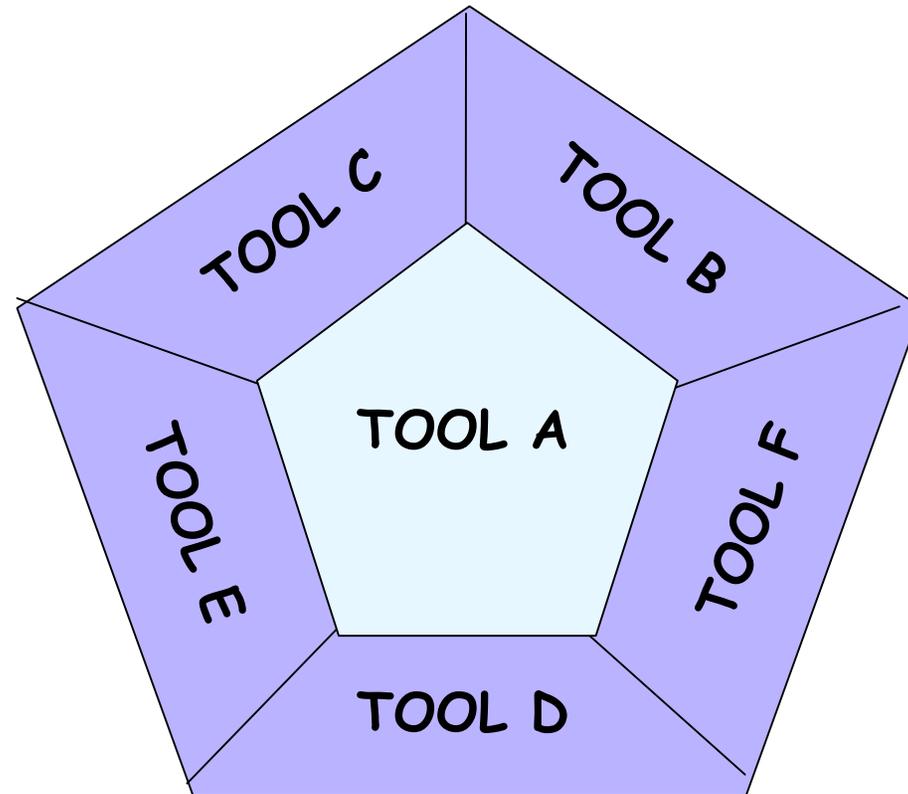
# ACTS Numerical Tools: *Functionality*

Computational Problem	Support	Technique	Library
Writing Parallel Programs (cont.)	Distributed Computing	Check-point/restart	CUMULVS
Profiling	Algorithmic Performance	Automatic instrumentation	PETSc
		User Instrumentation	PETSc
	Execution Performance	Automatic Instrumentation	TAU
		User Instrumentation	TAU
Code Optimization	Library Installation	Linear Algebra Tuning	ATLAS
Interoperability	Code Generation	Language	BABEL CHASM
		Components	CCA

# Tool Interoperability Tool-to-Tool

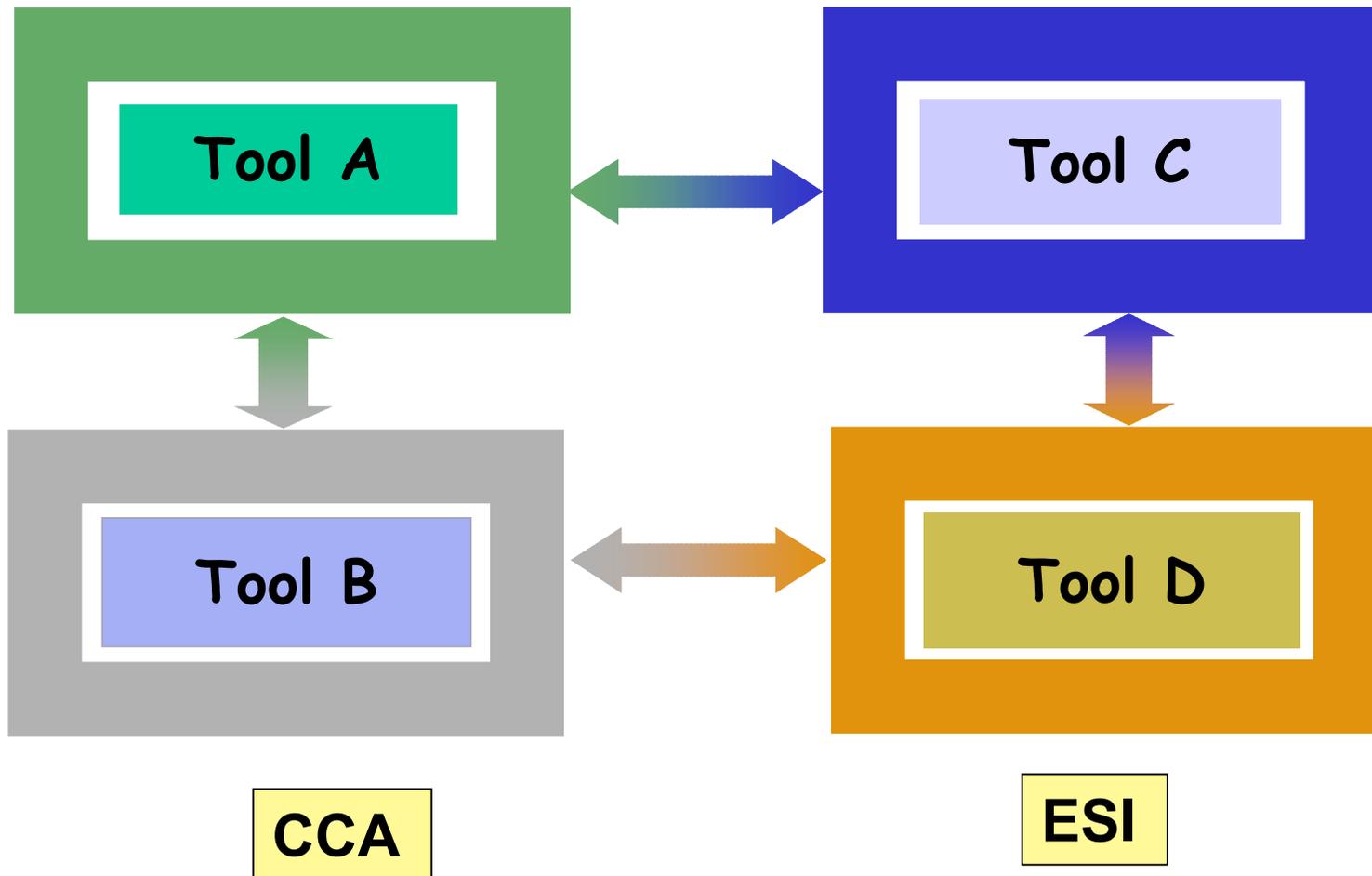


Ex 1



Ex 2

# Component Technology!



# PSE's and Frameworks

PMatlab

PyACTS

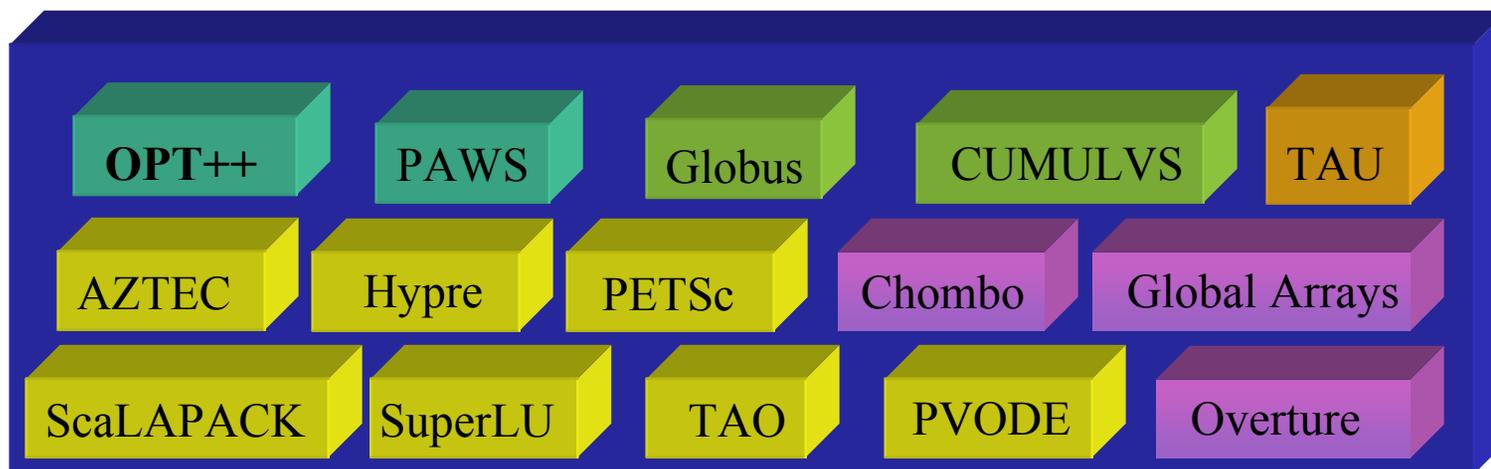


*View\_field(T1)*

$$Ax = b$$

$$Az = \lambda z$$

$$A = U\Sigma V^T$$



# PyACTS

```
Terminal
Window Edit Options Help
% run 4
-----
User: nkang                Repo: mpccc
Job Name: <none specified>  Group: mpccc
Class Of Service: interactive  Job Class: interactive
Job Accepted: Wed Jul 30 09:42:16 2003
-----

llsubmit: Processed command file through Submit Filter: "/usr/common/nsg/etc/sub
filter".
>>>
import sys
>>>
sys.path.append('/u6/nkang/kn/pyacts_1/build/lib.aix-5.1-mpi-2.2')
>>>
import scalapack
>>>
scalapack.ex2("ex2_mat", "ex2_rhs", "sol", 6, 1, 2, 2, 2, 1)

Scalapack Example Program #2 (C-version) -- 07/24/2003
Solving AX=B
where A is a 6 by 6 matrix,
B is a 6 by 1 matrix,
with a block size of 2
Running on 4 processes, where the process grid is 2 by 2
INFO code returned by PDGESV = 0

According to the normalized residual the solution is correct.
||AX-B|| / (||X||*||A||*eps*N) = 1.25878215e-01

The solution is written to file sol

End of test.
>>>
```

