UNIVERSITY OF MINNESOTA TWIN CITIES

Solving large eigenvalue problems in electronic structure calculations

Yousef Saad

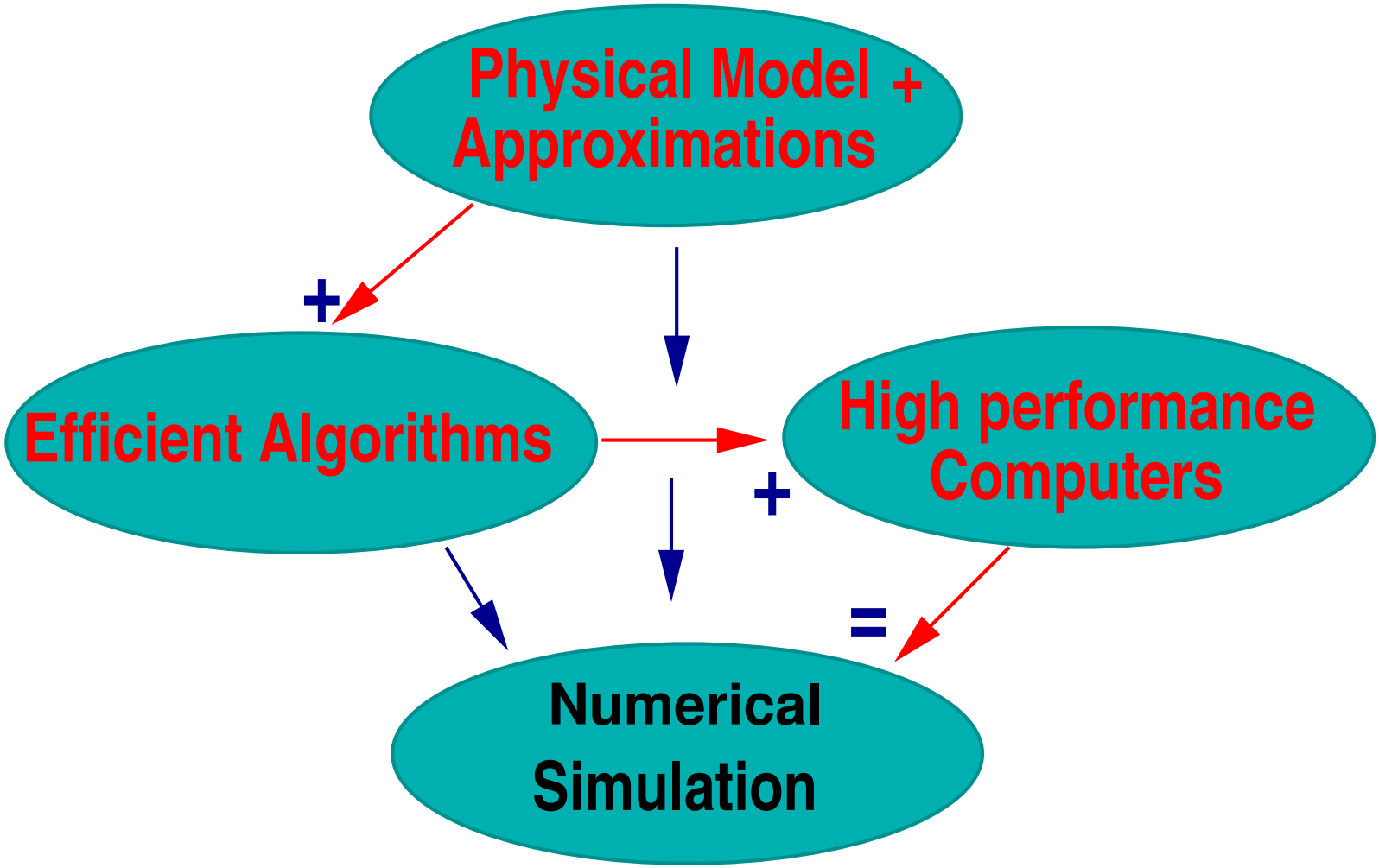
Department of Computer Science and Engineering

University of Minnesota

University of Queensland
Brisbane, Sept. 21, 2007

General preliminary comments

- **Ingredients of an effective numerical simulation:**



Most of the gains in speed combine advances from all 3 areas: simplifications from physics, effective numerical algorithms, and powerful hardware+software tools.

- More than ever a successful physical simulation must be cross-disciplinary.
- In particular, computational codes have become too complex to be handled by 'one-dimensional' teams.
- This talk: Algorithms – mostly 'diagonalization' –
- Will illustrate the above with experience of cross - disciplinary collaboration

Electronic structure and Schrödinger's equation

- Determining matter's electronic structure can be a major challenge:

  Number of particules is large [a macroscopic amount contains $\approx 10^{23}$ electrons and nuclei] and the physical problem is intrinsically complex.

- Solution via the many-body Schrödinger equation:

$$H\Psi = E\Psi$$

- In original form the above equation is very complex

- Hamiltonian H is of the form :

$$H = -\sum_i \frac{\hbar^2 \nabla_i^2}{2M_i} - \sum_j \frac{\hbar^2 \nabla_j^2}{2m} + \frac{1}{2}\sum_{i,j} \frac{Z_i Z_j e^2}{|\vec{R}_i - \vec{R}_j|} - \sum_{i,j} \frac{Z_i e^2}{|\vec{R}_i - \vec{r}_j|} + \frac{1}{2}\sum_{i,j} \frac{e^2}{|\vec{r}_i - \vec{r}_j|}$$

- $\Psi = \Psi(r_1, r_2, \ldots, r_n, R_1, R_2, \ldots, R_N)$ depends on coordinates of all electrons/nuclei.

- Involves sums over all electrons / nuclei and their pairs

- Note $\nabla_i^2 \Psi$ is Laplacean of Ψ w.r.t. variable r_i. Represents kinetic energy for i-th particle.

A hypothetical calculation: [with a "naive approach"]

- 10 Atoms each having 14 electrons [Silicon]
- ... a total of 15*10= 150 particles
- ... Assume each coordinate will need 100 points for discretization..
- ... you will get

$$\#\,\text{Unknowns} = \underbrace{100}_{part.1} \times \underbrace{100}_{part.2} \times \cdots \times \underbrace{100}_{part.150} = 100^{150}$$

- Methods based on this basic formulation are limited to a few atoms – useless for real compounds.

The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to the explanation of the main features of complex atomic systems without too much computations.

Dirac, 1929

- **In 1929 quantum mechanics was basically understood**
- **Today, the desire to have approximate practical methods is still alive**

Approximations/theories used

- **Born-Oppenheimer approximation: Neglects motion of nuclei [much heavier than electrons]**

- **Density Functional Theory: observable quantities are uniquely determined by ground state charge density.**

Kohn-Sham equation:

$$\left[-\frac{\nabla^2}{2} + V_{ion} + \int \frac{\rho(r')}{|r-r'|}dr' + \frac{\delta E_{xc}}{\delta \rho}\right]\Psi = E\Psi$$

The three potential terms

Effective Hamiltonian is of the form

$$-\frac{\nabla^2}{2} + V_{ion} + V_H + V_{xc}$$

- **Hartree Potential V_H = solution of Poisson equation:**

$$\nabla^2 V_H = -4\pi\rho(r)$$

where

$$\rho(r) = \Sigma_{i=1}^{occup} |\psi_i(r)|^2$$

- **Solve by CG or FFT once a distribution ρ is known.**

- **V_{xc} (exchange & correlation) approximated by a potential induced by a local density. [LDA]. Valid for slowly varying $\rho(r)$.**

The ionic potential: pseudopotentials

Potential V_{ion} is handled by pseudopotentials: replace effect of core (inner shell) electrons of the system by a smooth effective potential (Pseudopotential).

- **Must be smooth and replicate the same physical effect as that of all-electron potential. V_{ion} = sum of a local term and low-rank projectors for each atom.**

$$V_{ion} = V_{loc} + \sum_a P_a$$

Roughly speaking in matrix form: $P_a = \Sigma U_{lm} U_{lm}^{\top}$

- **A small-rank matrix localized around each atom.**

In the end:

$$\left[-\frac{\nabla^2}{2} + V_{ion} + V_H + V_{xc}\right]\Psi(r) = E\Psi(r)$$

With

- **Hartree potential (local)**

$$\nabla^2 V_H = -4\pi\rho(r)$$

- V_{xc} **depends on functional. For LDA:**

$$V_{xc} = f(\rho(r))$$

- V_{ion} **= nonlocal – does not explicitly depend on** ρ

$$V_{ion} = V_{loc} + \Sigma_a P_a$$

- V_H **and** V_{xc} **depend nonlinearly on eigenvectors:**

$$\rho(r) = \Sigma_{i=1}^{occup} |\psi_i(r)|^2$$

Self Consistence

- The potentials and/or charge densities must be self-consistent: Can be viewed as a nonlinear eigenvalue problem. Can be solved using different viewpoints

- **Nonlinear eigenvalue problem:** Linearize + iterate to self-consistence

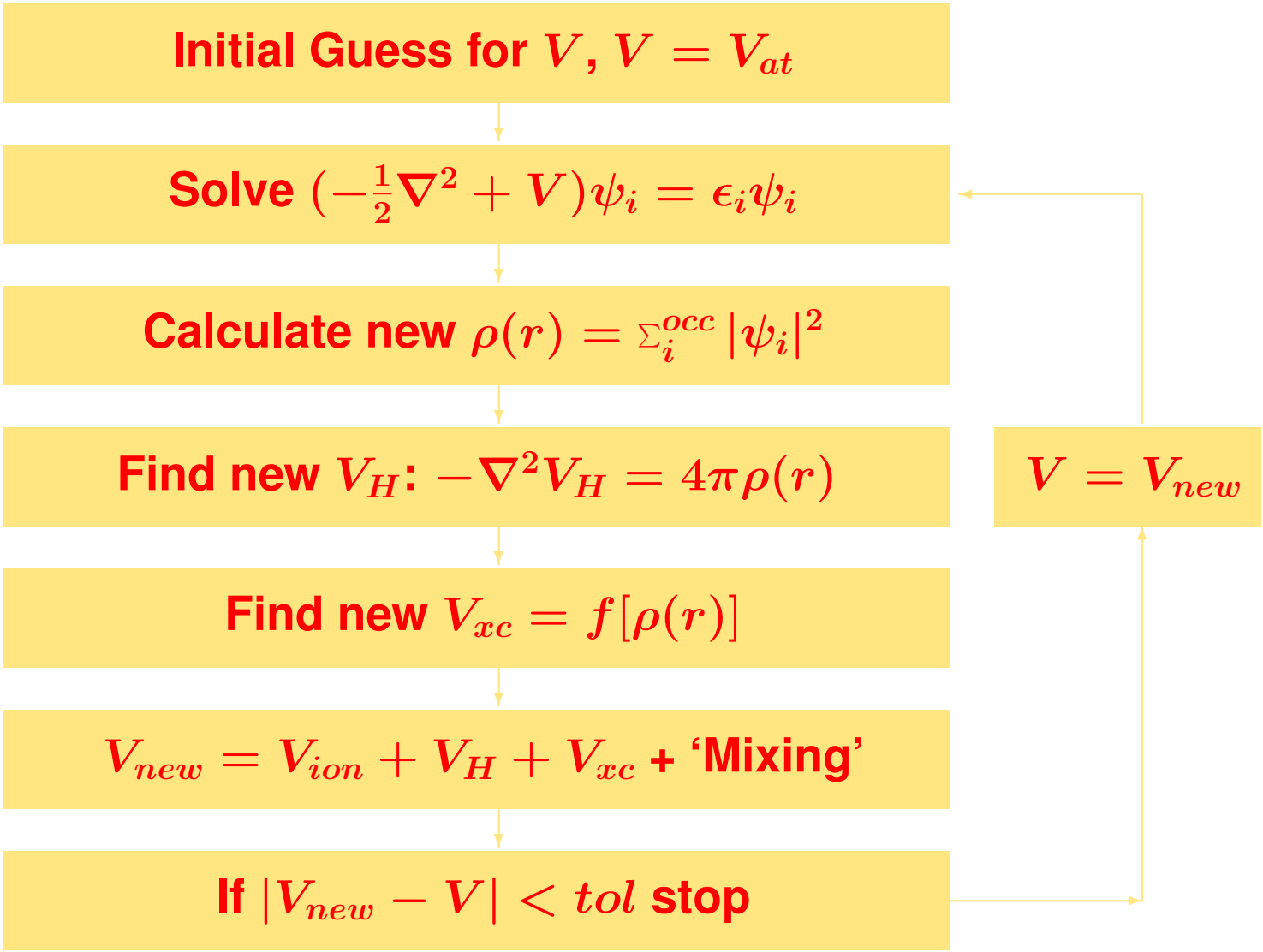
- **Nonlinear optimization:** minimize energy [again linearize + achieve self-consistency]

The two viewpoints are more or less equivalent

- Preferred approach: Broyden-type quasi-Newton technique

- Typically, a small number of iterations are required

Self-Consistent Iteration



- **Most time-consuming part = computing eigenvalues / eigenvectors.**

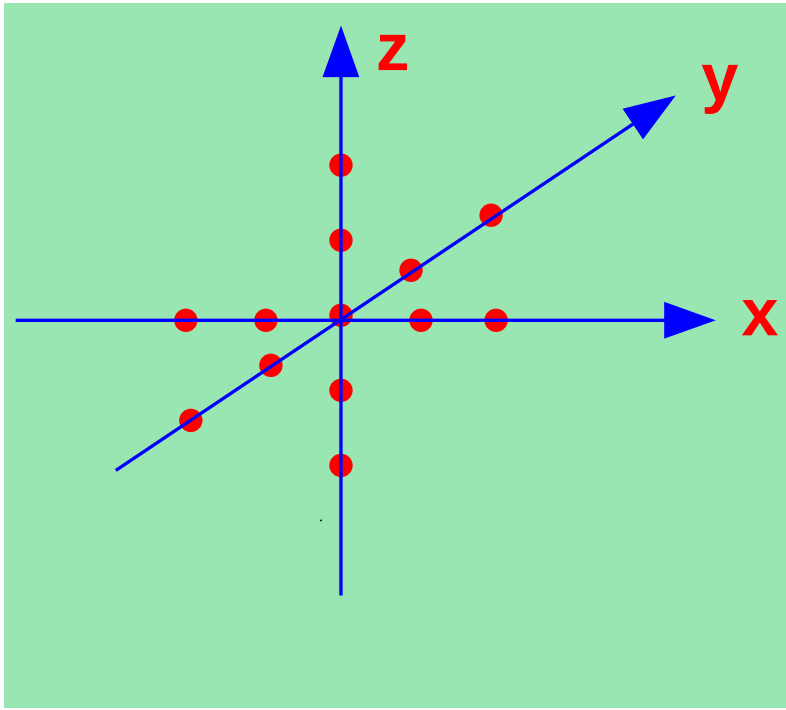
Characteristic : Large number of eigenvalues /-vectors to compute [occupied states]. For example Hamiltonian matrix size can be $N = 1,000,000$ and the number of eigenvectors about 1,000.

- **Self-consistent loop takes a few iterations (say 10 or 20 in easy cases).**

Real-space Finite Difference Methods

- Use High-Order Finite Difference Methods [Fornberg & Sloan '94]
- Typical Geometry = Cube – regular structure.
- Laplacean matrix need not even be stored.

Order 4 Finite Difference Approximation:



PARSEC

- PARSEC = Pseudopotential Algorithm for Real Space Electronic Calculations
- Represents $\approx$ 15 years of gradual improvements
- Runs on sequential and parallel platforms – using MPI.
- Efficient diagonalization -
- Takes advandate of symmetry

PARSEC - A few milestones

- Sequential real-space code on Cray YMP [up to ’93]
- Cluster of SGI workstations [up to ’96]
- CM5 [’94-’96] Massive parallelism begins
- IBM SP2 [Using PVM]
- Cray T3D [PVM + MPI] $\sim$ ’96; Cray T3E [MPI] – ’97
- IBM SP with +256 nodes – ’98+
- SGI Origin 3900 [128 processors] – ’99+
- IBM SP + F90 - PARSEC name given, ’02
- ’05: PARSEC released
- Now: Runs on most machines (SGI Altix, IBM SP, Cray XT, ...)

Main contributors to code:

Early days:

N. Troullier

K. Wu (*)

X. Jing

H. Kim

A. Stathopoulos (*)

I. Vassiliev

More recent:

M. Jain

L. Kronik

R. Burdick (*)

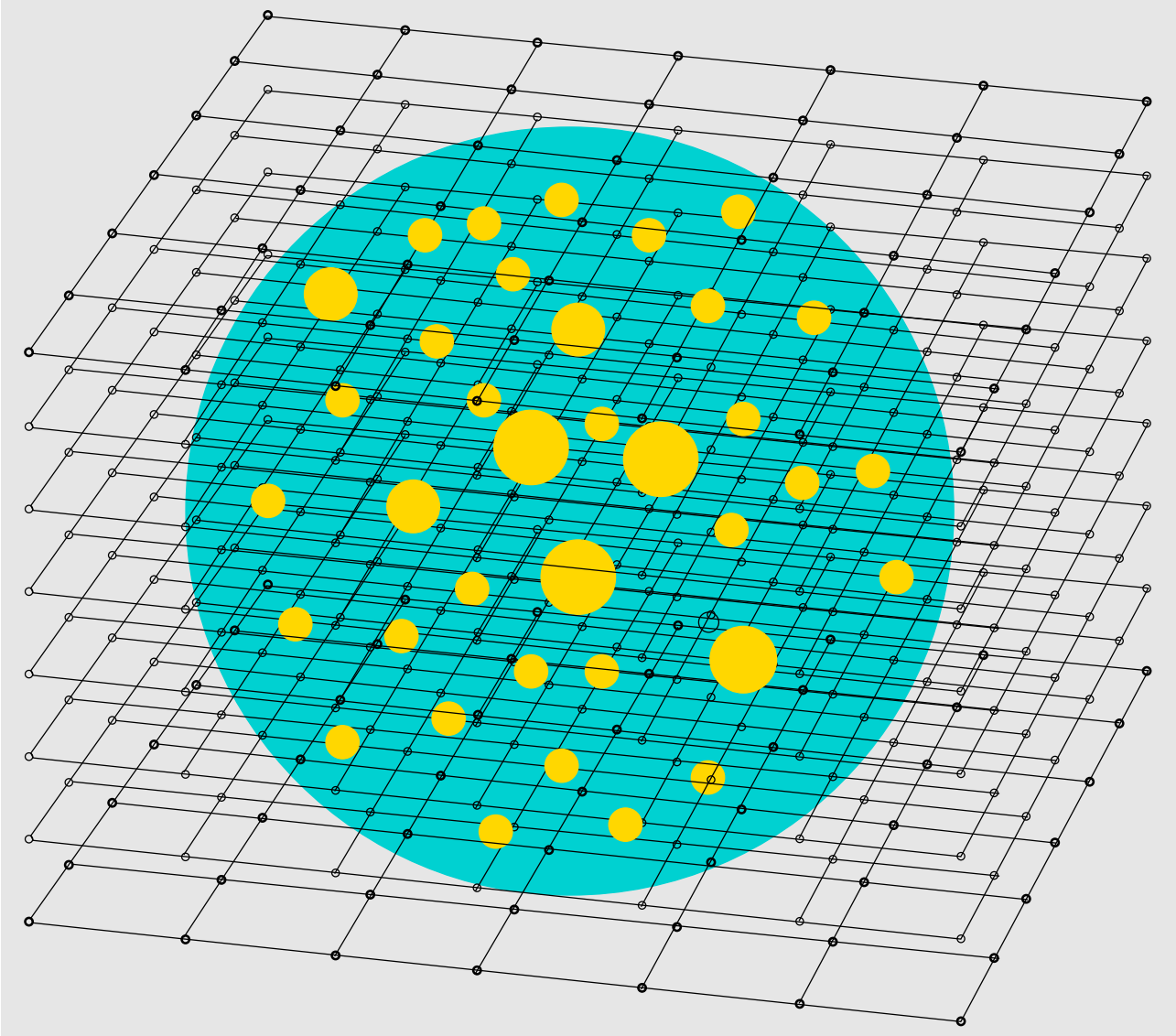
M. Alemany

M. Tiago

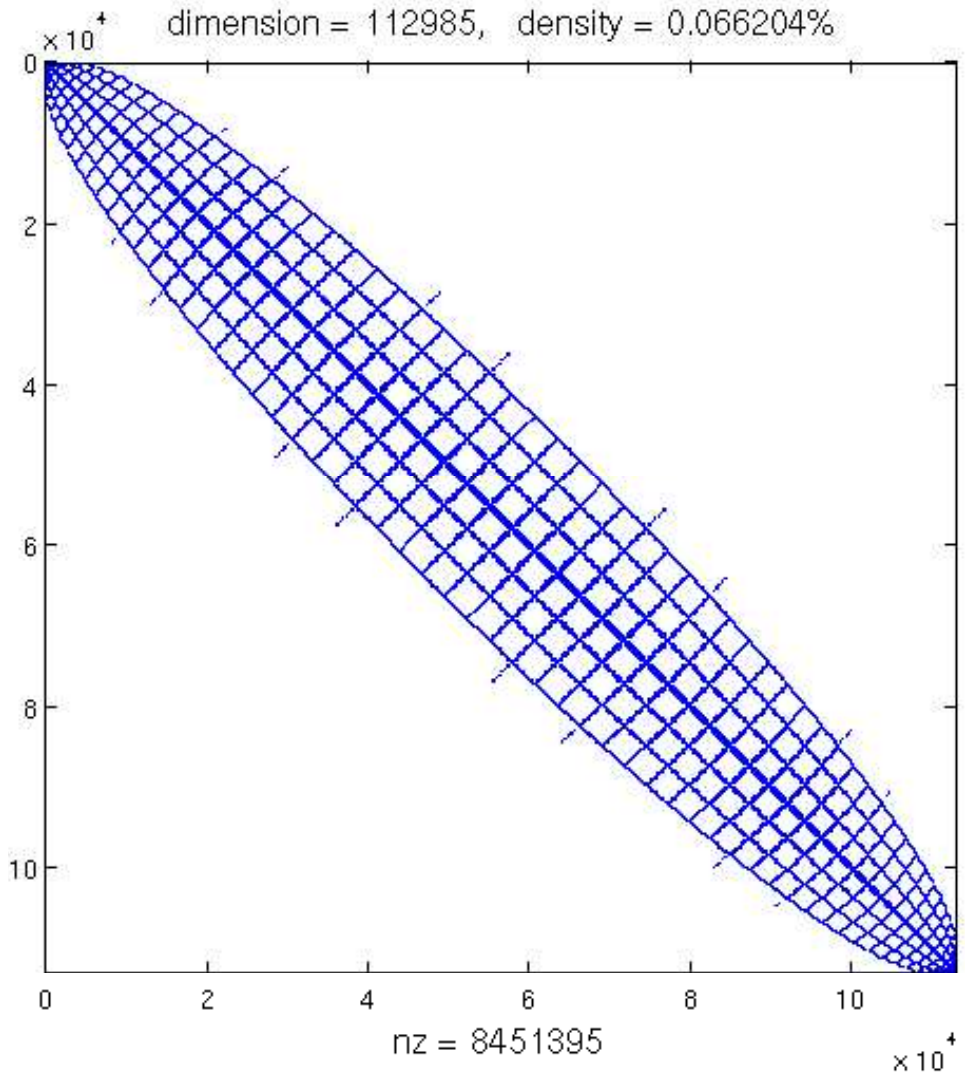
Y. Zhou (*)

(*) = Computer-Scientist

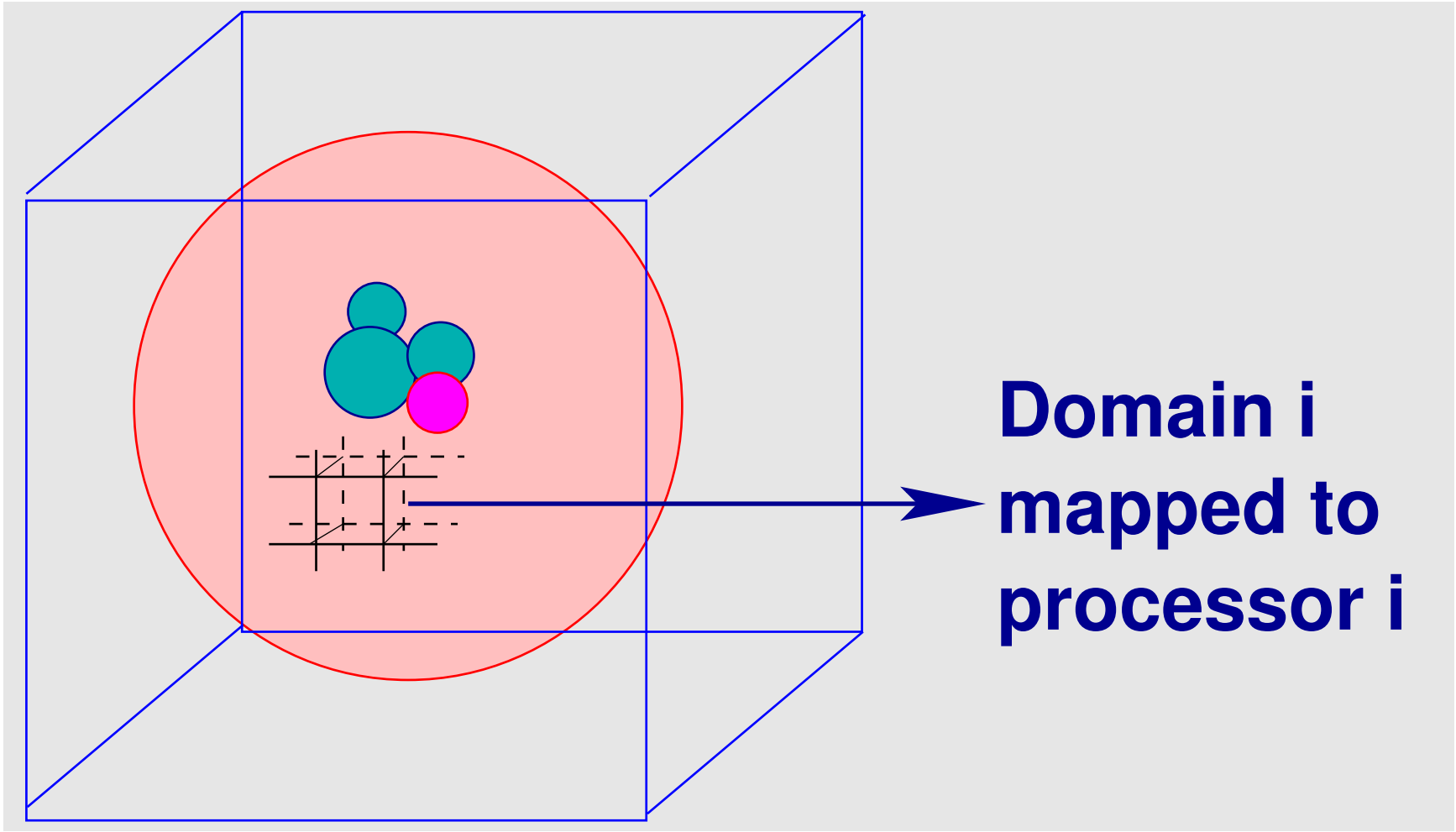
The physical domain



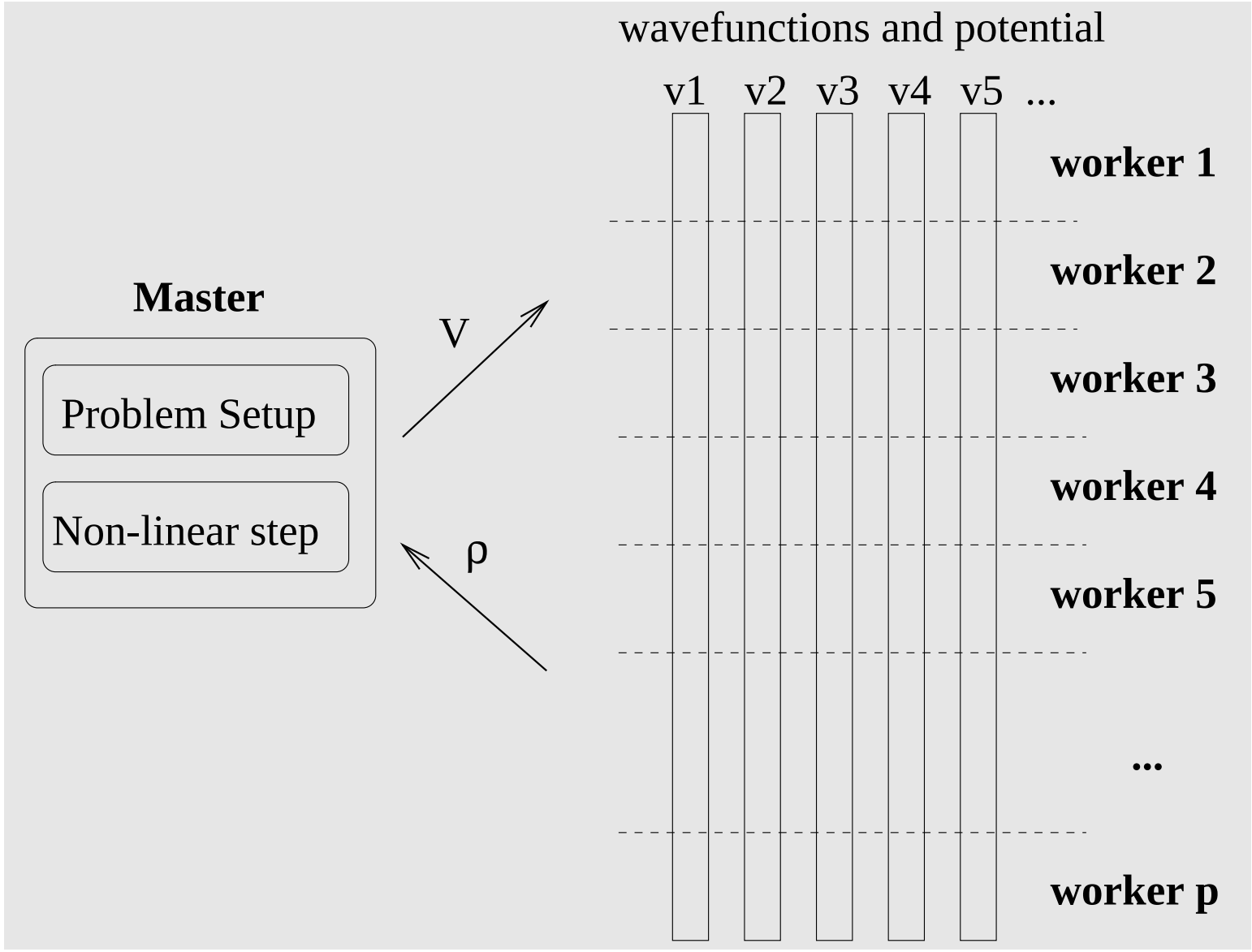
Pattern of resulting matrix for Ge99H100:



Domain Mapping:



A domain decomposition approach is used


wavefunctions and potential
v1
v2
v3
v4
v5
...
worker 1
worker 2
worker 3
worker 4
worker 5
...
worker p
Master
Problem Setup
Non-linear step
V
ρ

Sample calculations done: Quantum dots

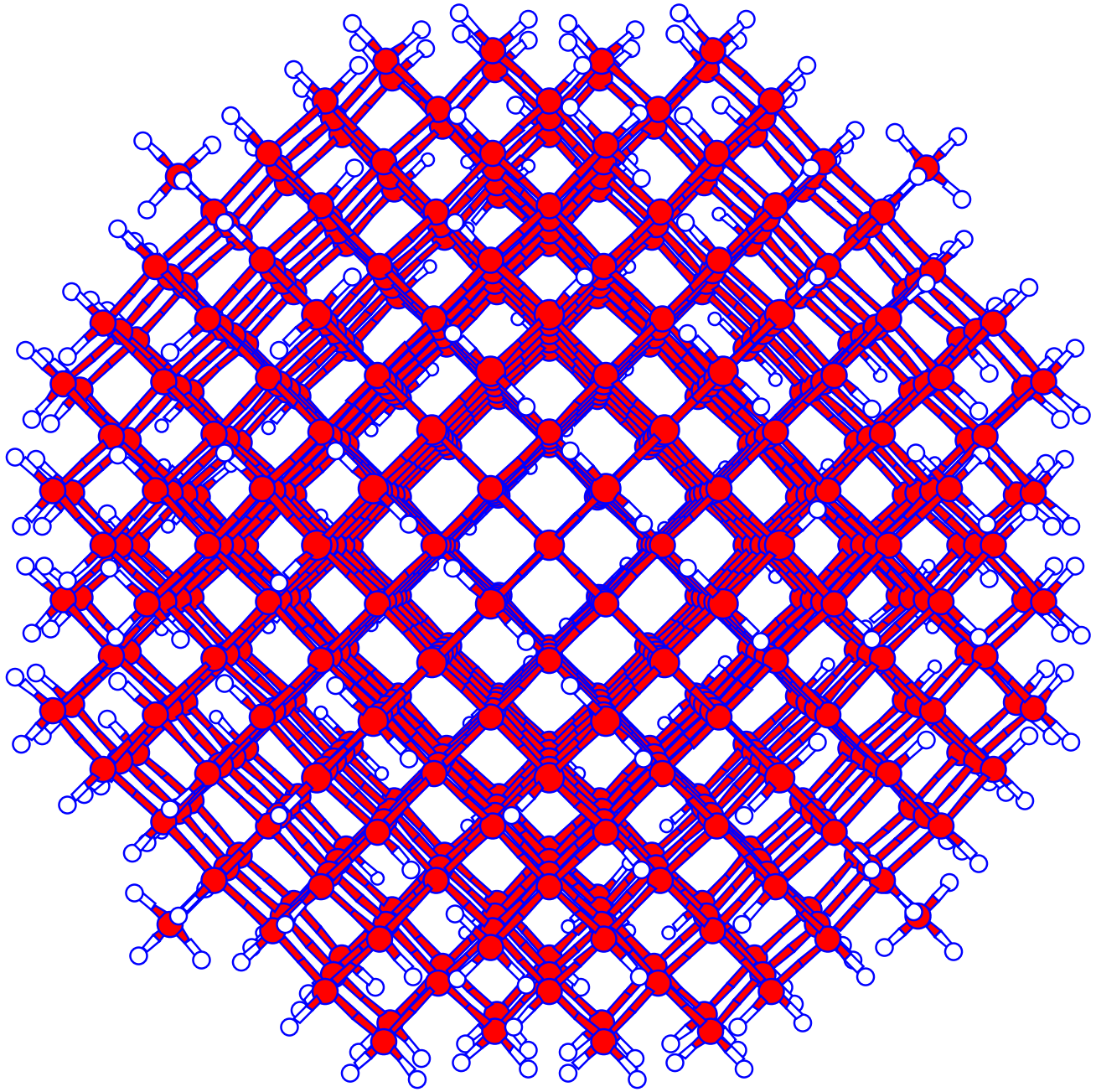
- **Small silicon clusters ($\approx 20 - 100 Angst.$ Involve up to a few hundreds atoms)**

- **$Si_{525}H_{276}$ leads to a matrix size of $N \approx 290,000$ and $n_{states} = 1,194$ eigenpairs.**
- **In 1997 this took $\sim$ 20 hours of CPU time on the Cray T3E, using 48 processors.**
- **TODAY: 2 hours on one SGI Madison proc. (1.3GHz)**
- **Could be done on a good workstation!**

- **Gains: hardware and algorithms**

- **Algorithms: Better diagonalization + New code exploits symmetry**

$Si_{525}H_{276}$



Matlab version: RSDFT (work in progress)

- Goal is to provide (1) prototyping tools (2) simple codes for teaching Real-space DFT with pseudopotentials
- Can do small systems on this laptop – [Demo later..]
- Idea: provide similar input file as PARSEC –
- Many summer interns helped with the project. This year and last year's interns:

Virginie Audin, Long Bui, Nate Born, Amy Coddington, Nick Voshell, Adam Jundt, ...

+ ... others who worked with a related visualization tool.

Diagonalization methods in PARSEC

1. At the beginning there was simplicity: DIAGLA

- In-house parallel code developed circa 1995 (1st version)
- Uses a form of Davidson's method with various enhancements
- Feature: can use part of a subspace from previous scf iteration
- One issue: Preconditioner required in Davidson – but not easy in real-space methods [in contrast with planewaves]

2. Later ARPACK was added as an option

- State of the art public domain diagonalization package
- Could be a few times faster than DIAGLA - Also: quite robust.

But...

- .. cannot reuse previous eigenvectors for next SCF iteration.
- .. Not easy to modify to adjust to our needs
- .. Really a method not designed for large eigenspaces

3. Recent work: focus on eigenspaces – more on this shortly

4. Also explored - AMLS

- Domain-decomposition type approach
- Very complex (to implement) ...
- and ... accuracy not sufficient for DFT approaches [Although this may be fixed]

Current work on diagonalization

Focus:

- **Compute eigen-space - not individual eigenvectors.**
- **Take into account outer (SCF) loop**
- **Future: eigenvector-free or basis-free methods**

Motivation:

Standard packages (ARPACK) do not easily take advantage of specificity of problem: self-consistent loop, large number of eigenvalues, ...

Example: Partial Reorth. Lanczos - PLAN

- Compute eigenspace instead of individual eigenvectors
- No full reorthogonalization: reorthogonalize when needed
- No restarts – so, much larger basis needed in general

Ingredients: (1) test of loss of orthogonality (recurrence relation) and (2) stopping criterion based on charge density instead of eigenvectors.

Partial Reorth. Lanczos - (Background)

- **Recall the Lanczos recurrence:**

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

- **Scalars β_{j+1}, α_j selected so that $v_{j+1} \perp v_j$, $v_{j+1} \perp v_{j-1}$, and $\|v_{j+1}\|_2 = 1$.**

- **In theory this is enough to guarantee that $\{v_j\}$ is orthonormal. + we have:**

$$V^T A V = T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}$$

- In practice: Loss of orthogonality takes place as soon as first eigenvalues start to converge [C. Paige, 1970s]

Remedy: reorthogonalize.

- Partial reorthogonalization: reorthogonalize only when deemed necessary.

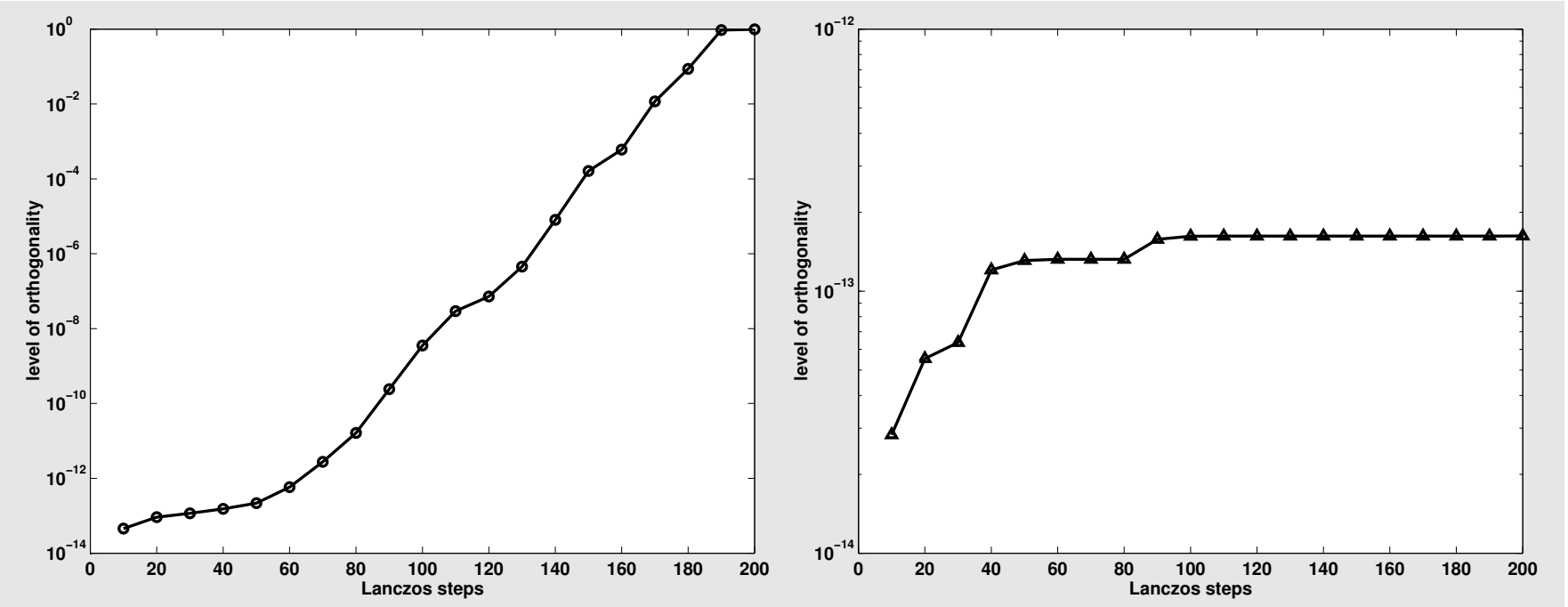
- Uses an inexpensive recurrence relation

- Work done in the 80's [Parlett, Simon, and co-workers] + more recent work [Larsen, '98]

- Package: PROPACK [Larsen] V 1: 2001, most recent: V 2.1 (Apr. 05)

- In tests with real-space Hamiltonians from PARSEC, need for re-orthogonalization not too strong.

Reorthgonalization: a small example



Levels of orthogonality of the Lanczos basis for the Hamiltonian ($n = 17077$) corresponding to $\mathrm{Si_{10}H_{16}}$. Left: no reorthogonalization. Right: partial reorth. (34 in all)

Second ingredient: avoid computing and updating eigenvalues / eigenvectors. Instead:

Test how good is the underlying eigenspace without knowledge of individual eigenvectors. When converged – then compute the basis (eigenvectors). Test: sum of occupied energies has converged = a sum of eigenvalues of a small tridiagonal matrix. Inexpensive.

➨ See:

"Computing Charge Densities with Partially Reorthogonalized Lanczos", C. Bekas, Y. Saad, M. L. Tiago, and J. R. Chelikowsky; to appear, CPC.

Partial Reorth. Lanczos vs. ARPACK for $Ge_{99}H_{100}$.

	Partial Lanczos				ARPACK			
n_o	$A*x$	orth	mem.	secs	$A*x$	rest.	mem.	secs
248	3150	109	2268	2746	3342	20	357	16454
350	4570	184	3289	5982	5283	24	504	37371
496	6550	302	4715	13714	6836	22	714	67020

- Matrix-size = 94,341; # nonzero entries = 6,332,795
- Number of occupied states : neig=248.
- Requires more memory but ...
- ... Still manageable for most systems studied [can also use secondary storage]
- ... and gain a factor of 4 to 6 in CPU time

CHEBYSHEV FILTERING

Chebyshev Subspace iteration

- **Main ingredient: Chebyshev filtering**

Given a basis $[v_1, \ldots, v_m]$, 'filter' each vector as

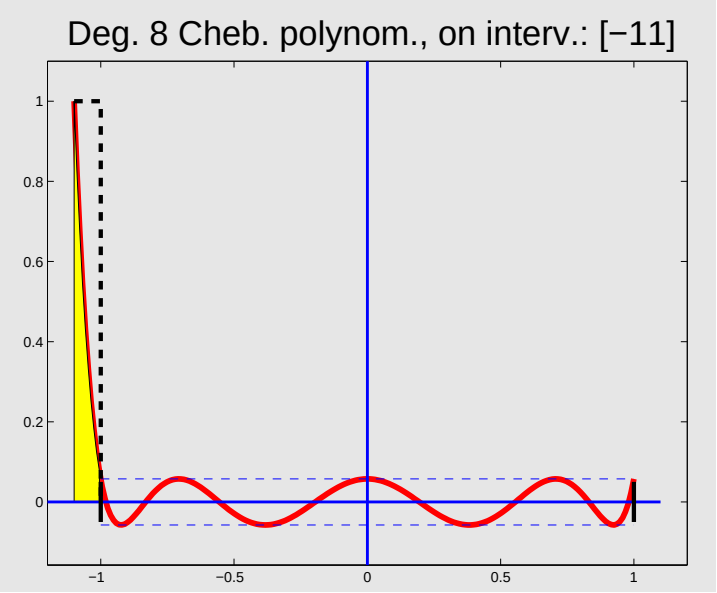
$$\hat{v}_i = P_k(A)v_i$$

- **p_k = Low deg. polynomial. Enhances wanted eigencomponents**

The filtering step is not used to compute eigenvectors accurately

SCF & diagonalization loops merged

Important: convergence still good and robust



Main step:

Previous basis $V = [v_1, v_2, \cdots, v_m]$

$\downarrow$

Filter $\hat{V} = [p(A)v_1, p(A)v_2, \cdots, p(A)v_m]$

$\downarrow$

Orthogonalize $[V, R] = qr(\hat{V}, 0)$

- The basis V is used to do a Ritz step (basis rotation)

$$C = V^T A V \rightarrow [U, D] = eig(C) \rightarrow V := V * U$$

- Update charge density using this basis.
- Update Hamiltonian — repeat

- **In effect: Nonlinear subspace iteration**

- **Main advantages: (1) very inexpensive, (2) uses minimal storage (m is a little $\geq$ # states).**

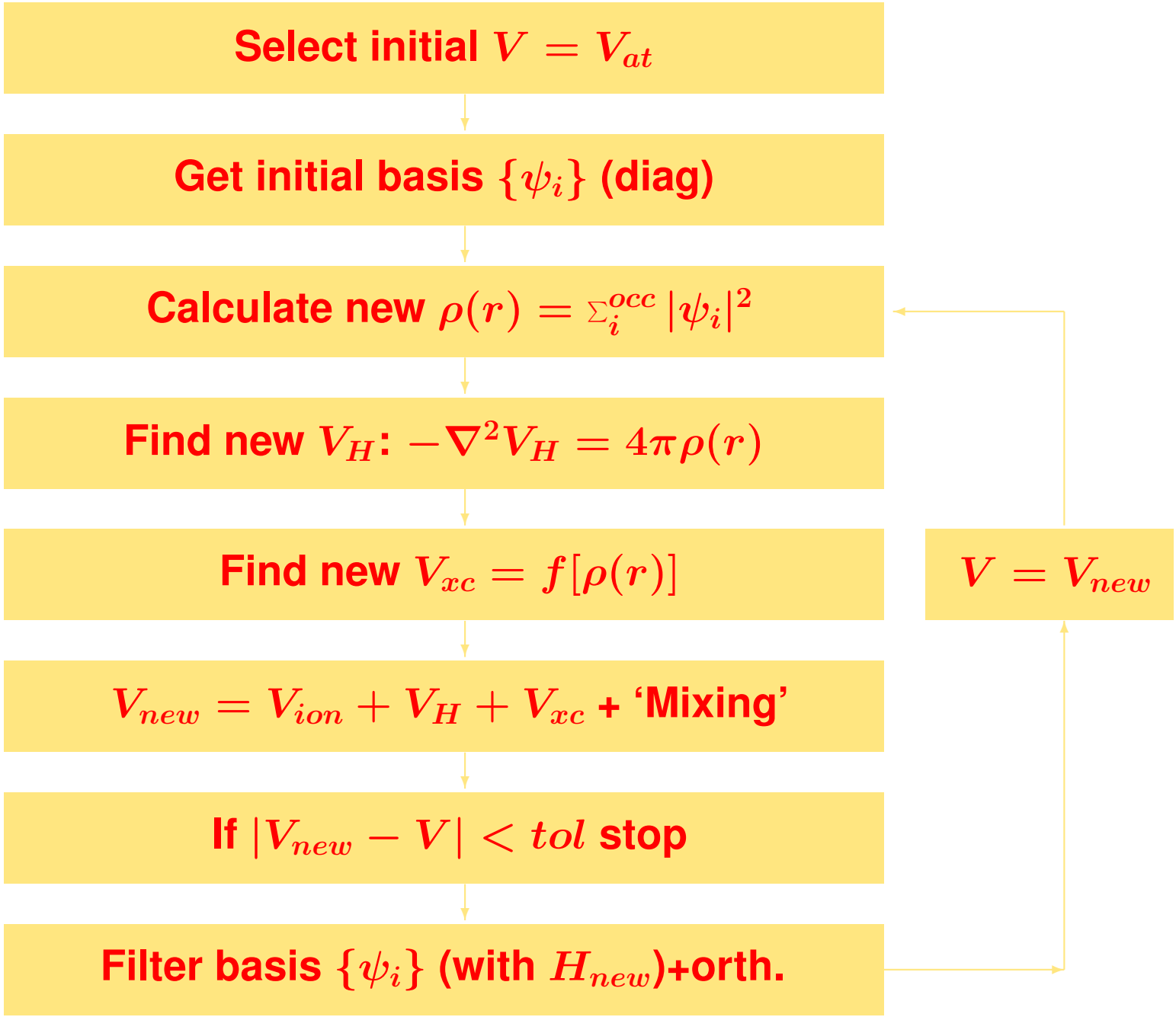
- **Filter polynomials: if $[a,\ b]$ is interval to dampen, then**

$$p_k(t) = \frac{C_k(l(t))}{C_k(l(c))}; \quad \text{with} \quad l(t) = \frac{2t-b-a}{b-a}$$

- $c \approx$ **eigenvalue farthest from** $(a+b)/2$ **– used for scaling**

- **3-term recurrence of Chebyshev polynommial exploited to compute** $p_k(A)v$. **If** $B = l(A)$, **then** $C_{k+1}(t) = 2tC_k(t) - C_{k-1}(t) \rightarrow$

$$w_{k+1} = 2Bw_k - w_{k-1}$$


Select initial $V = V_{at}$
Get initial basis $\{\psi_i\}$ (diag)
Calculate new $\rho(r) = \Sigma_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 V_H = 4\pi\rho(r)$
Find new $V_{xc} = f[\rho(r)]$
$V_{new} = V_{ion} + V_H + V_{xc}$ + 'Mixing'
If $|V_{new} - V| < tol$ stop
Filter basis $\{\psi_i\}$ (with H_{new})+orth.
$V = V_{new}$

Reference:

Yunkai Zhou, Y.S., Murilo L. Tiago, and James R. Chelikowsky, Parallel Self-Consistent-Field Calculations with Chebyshev Filtered Subspace Iteration, Phy. Rev. E, vol. 74, p. 066704 (2006).

[See http://www.cs.umn.edu/∼saad]

Chebyshev Subspace iteration - experiments

model	size of H	n_{state}	n_{symm}	$n_{H-reduced}$
$Si_{525}H_{276}$	292,584	1194	4	73,146
$Si_{65}Ge_{65}H_{98}$	185,368	313	2	92,684
$Ga_{41}As_{41}H_{72}$	268,096	210	1	268,096
Fe_{27}	697,504	520×2	8	87,188
Fe_{51}	874,976	520×2	8	109,372

Test problems

- **Tests performed on an SGI Altix 3700 cluster (Minnesota supercomputing Institute). [CPU = a 1.3 GHz Intel Madison processor. Compiler: Intel FORTRAN `ifort`, with optimization flag `-O3`]**

method	# $A * x$	SCF its.	CPU(secs)
ChebSI	124761	11	5946.69
ARPACK	142047	10	62026.37
TRLan	145909	10	26852.84

$Si_{525}H_{276}$, Polynomial degree used is 8. Total energies agreed to within 8 digits.

method	# $A * x$	SCF its.	CPU (secs)
ChebSI	42919	13	2344.06
ARPACK	51752	9	12770.81
TRLan	53892	9	6056.11

$Si_{65}Ge_{65}H_{98}$, Polynomial degree used is 8. Total energies same to within 9 digits.

method	# $A * x$	SCF its.	CPU (secs)
ChebSI	138672	37	12923.27
ARPACK	58506	10	44305.97
TRLan	58794	10	16733.68

$Ga_{41}As_{41}H_{72}$. Polynomial degree used is 16. The spectrum of each reduced Hamiltonian spans a large interval, making the Chebyshev filtering not as effective as other examples. Total energies same to within 9 digits.

method	# $A * x$	SCF its.	CPU (secs)
ChebSI	363728	30	15408.16
ARPACK	750883	21	118693.64
TRLan	807652	21	83726.20

Fe_{27}, Polynomial degree used is 9. Total energies same to within ~ 5 digits.

method	# $A * x$	SCF its.	CPU (secs)
ChebSI	474773	37	37701.54
ARPACK	1272441	34	235662.96
TRLan	1241744	32	184580.33

Fe_{51}, Polynomial degree used is 9. Total energies same to within ~ 5 digits.

Larger tests

- **Large tests with Silicon and Iron clusters –**

Legend:

- n_{state} : number of states
- n_H : size of Hamiltonian matrix
- # $A * x$: number of total matrix-vector products
- # SCF : number of iteration steps to reach self-consistency
- $\frac{total\_eV}{atom}$: total energy per atom in electron-volts
- 1st CPU : CPU time for the first step diagonalization
- total CPU : total CPU spent on diagonalizations to reach self-consistency

Silicon clusters [symmetry of 4 in all cases]

$Si_{2713}H_{828}$

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
5843	1400187	14	-86.16790	7.83 h.	19.56 h.

PEs = 16. H Size = 1,074,080. $m = 17$ for Chebyshev-Davidson; $m = 10$ for CheFSI.

Note: First diagonalization by TRLan costs 8.65 hours. Fourteen steps would cost $\approx 121h$.

$Si_{4001}H_{1012}$

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
8511	1652243	12	-89.12338	18.63 h.	38.17 h.

PEs = 16. n_H =1,472,440. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFSI.

Note: 1st step diagonalization by TRLan costs 34.99 hours

$Si_{6047}H_{1308}$

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
12751	2682749	14	-91.34809	45.11 h.	101.02 h.

PEs = 32. n_H =2,144,432. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFSI.

Note: comparisons with TRlan no longer available

$Si_{9041}H_{1860}$

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
19015	4804488	18	-92.00412	102.12 h.	294.36 h.

PEs = 48; n_H =2,992,832. $m = 17$ for Chebyshev-Davidson; $m = 8$ for CheFSI.

Iron clusters [symmetry of 12 in all cases]

Fe_{150}

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
900×2	2670945	66	-794.95991	2.34 h.	19.67 h.

PEs = 16. n_H =1,790,960. $m = 20$ for Chebyshev-Davidson; $m = 18$ for CheFSI.

Fe_{286}

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
1716×2	7405332	100	-795.165	10.19	71.66 h.

PEs = 16. n_H =2,726,968 $m = 20$ for Chebyshev-Davidson; $m = 17$ for CheFSI.

Fe_{388}

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
2328×2	18232215	187	-795.247	16.22	247.05 h.

Fe_{388}

#PE= 24. $n_H = 3332856$. $m = 20$ for Chebyshev-Davidson; $m = 18$ for CheFSI.

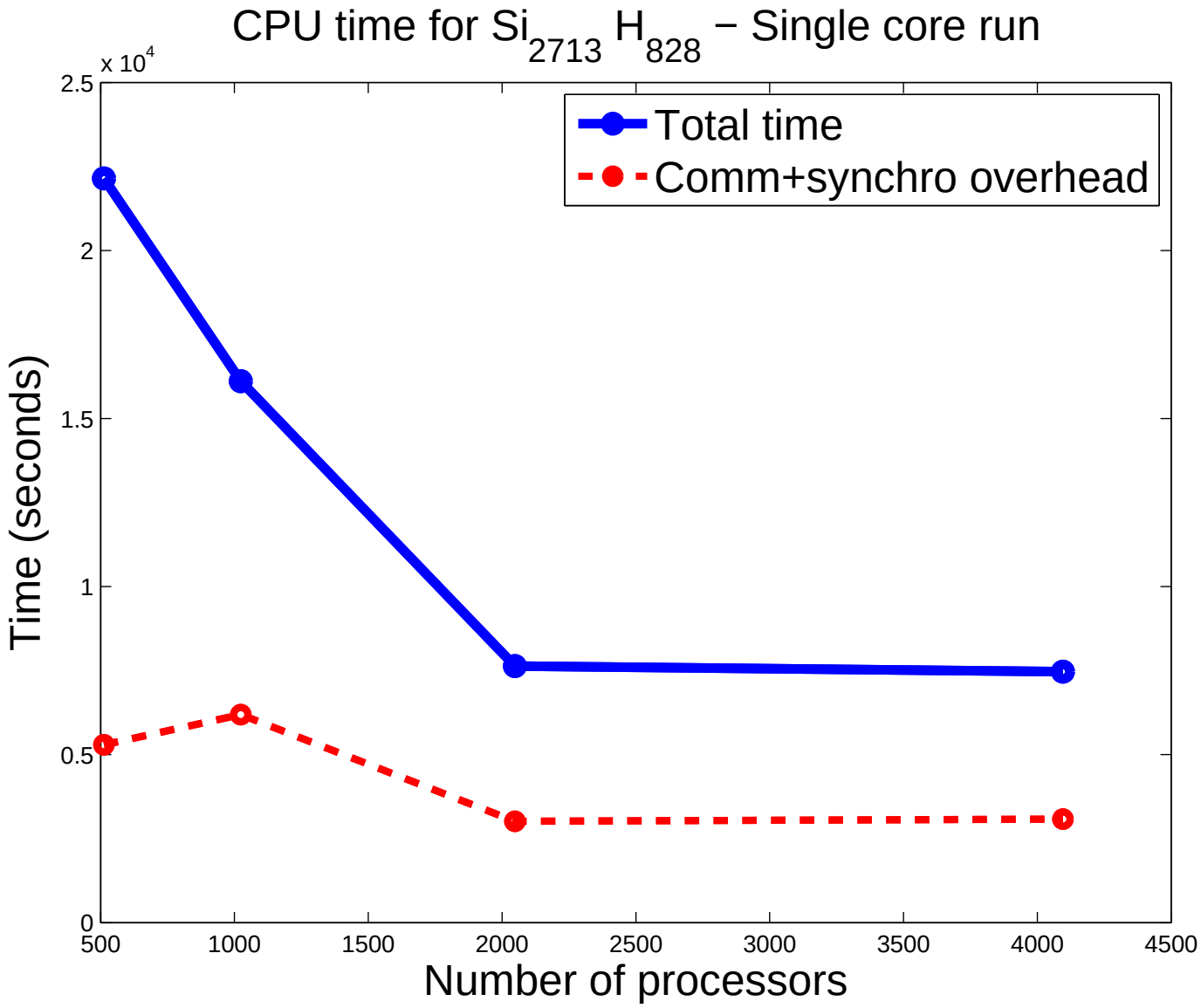
Reference:

M. L. Tiago, Y. Zhou, M. M. G. Alemany, YS, and J.R. Chelikowsky, The evolution of magnetism in iron from the atom to the bulk, Physical Review Letters, vol. 97, pp. 147201-4, (2006).

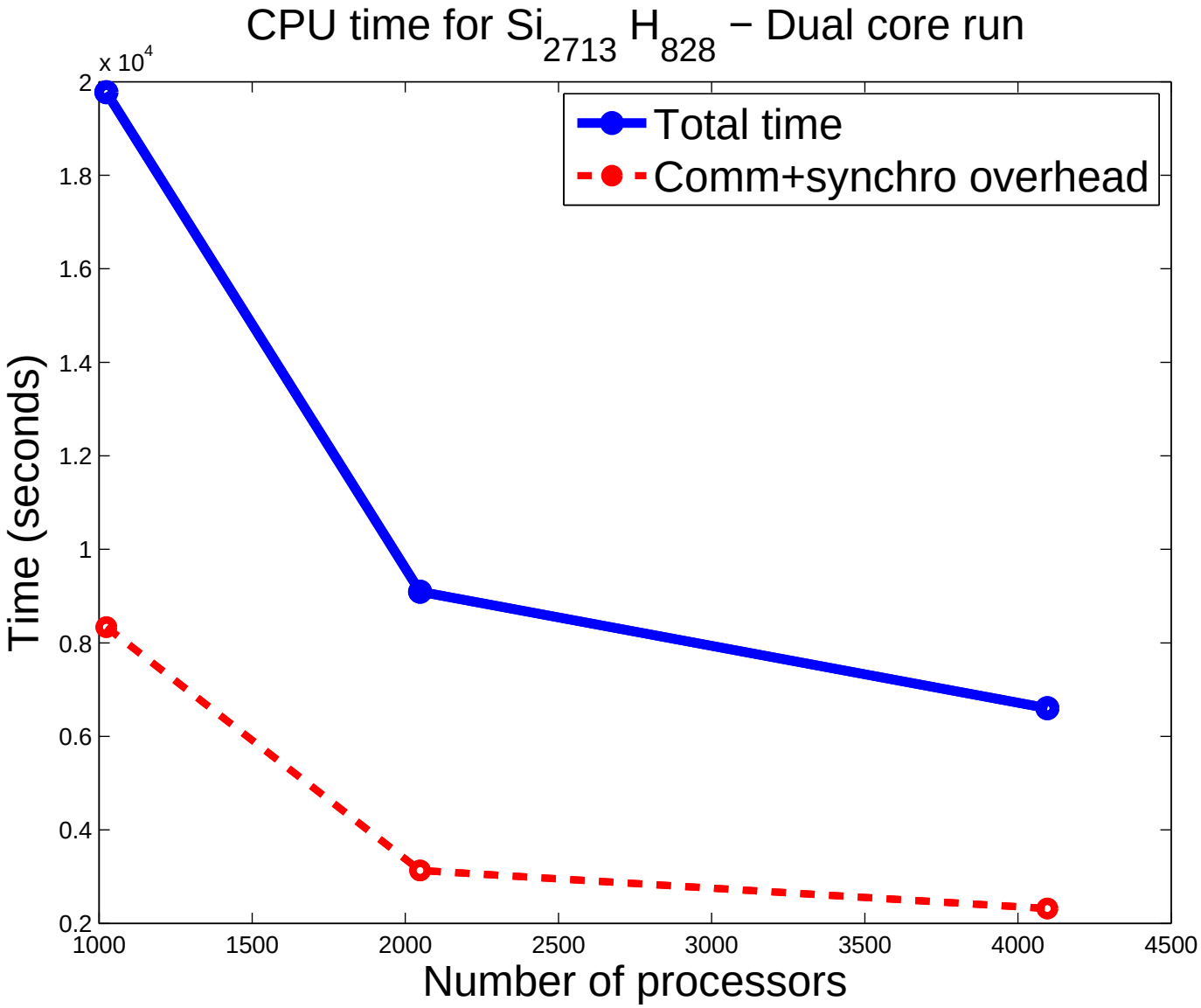
Scalability

- Recent runs on the Cray X T4 at Oak Ridge national labs.
- Dual-core AMD Opteron processors (2.6 Ghz) with 4GB mem/ node.
- 3-D Torus topology
- Total number of processors = 5294 [aggregate peak flops = 100 Tflops]
- Systems tested: $Si_{2713}H_{828}$ – [$\approx 5,000$ eigenvectors computed]

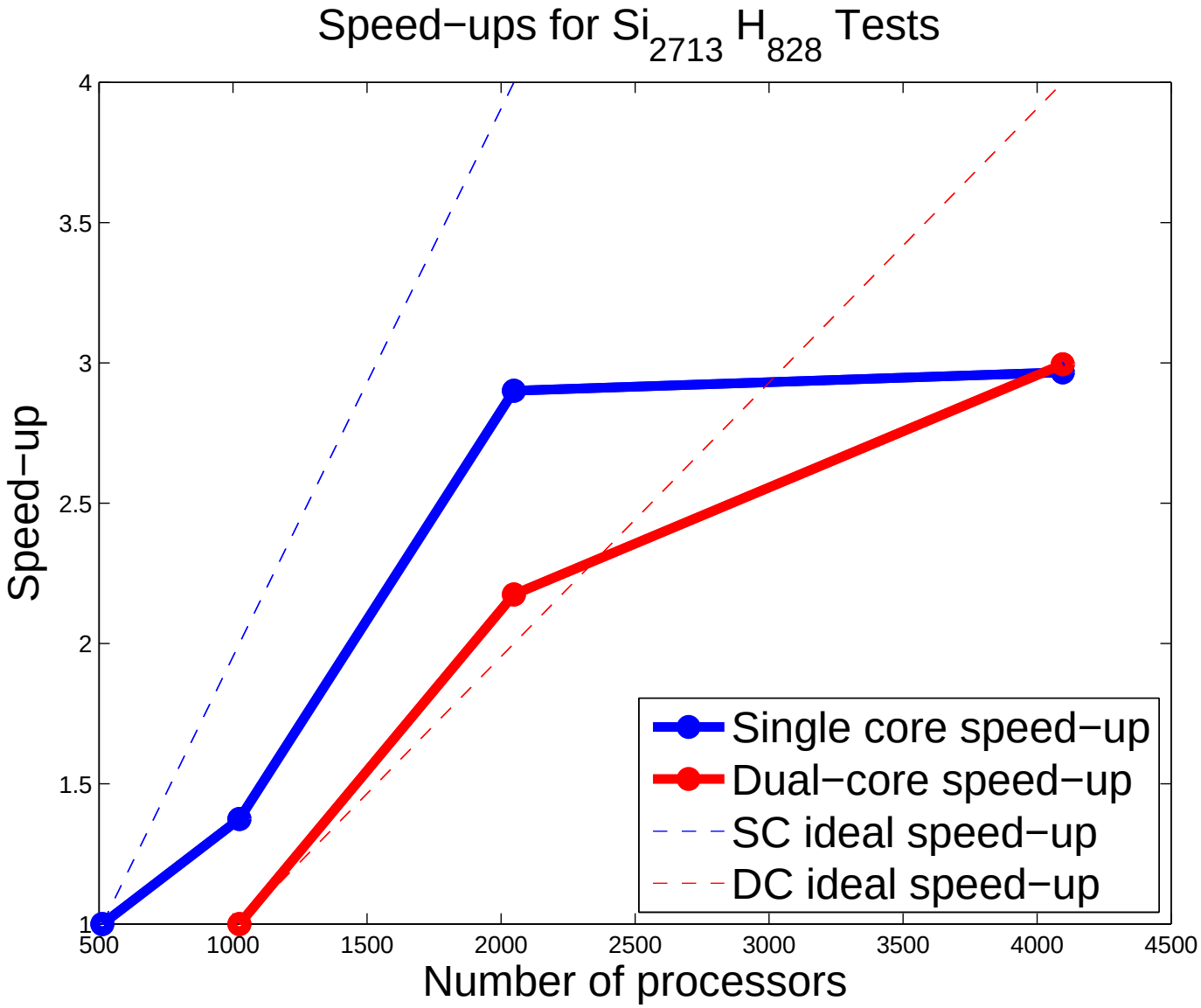
Time vs. processors - Single core



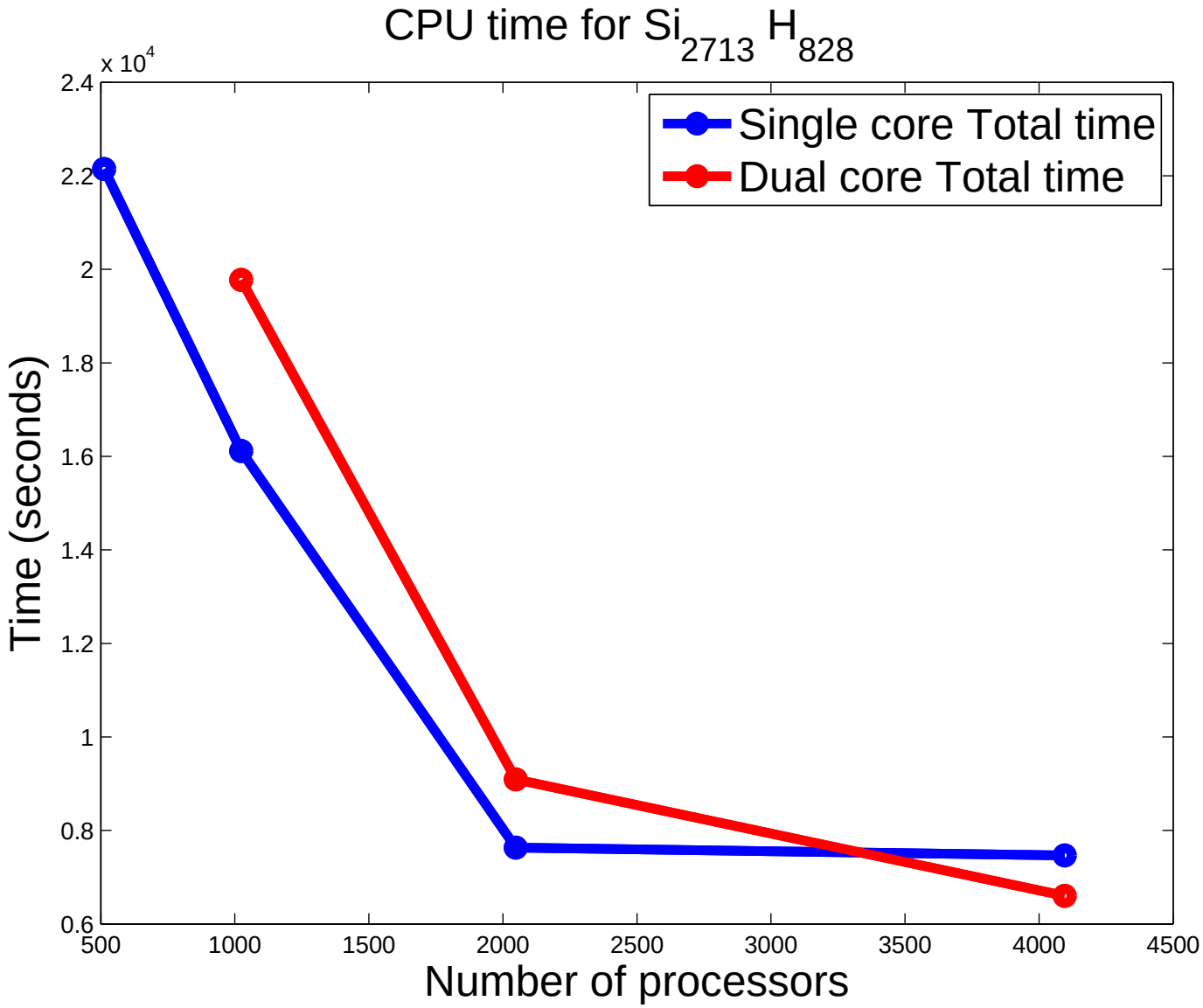
Time vs. processors - Dual core



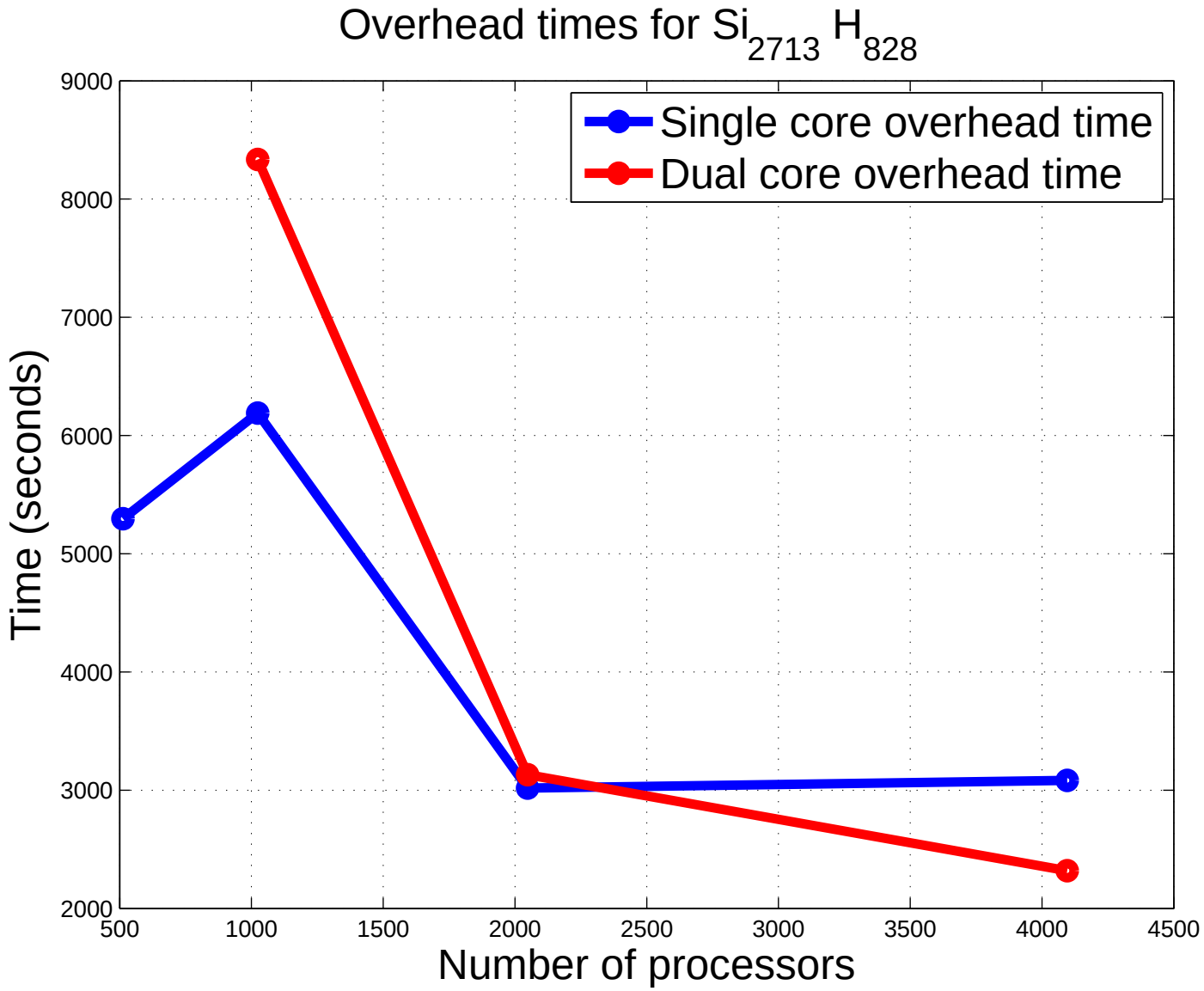
Speed-ups


Speed−ups for Si2713 H828 Tests
Speed−up
Number of processors
4
3.5
3
2.5
2
1.5
1
500
1000
1500
2000
2500
3000
3500
4000
4500
Single core speed−up
Dual−core speed−up
SC ideal speed−up
DC ideal speed−up

Times - comparisons SC/DC



Overhead - comparisons SC/DC



Eigenvalue-free methods

- Diagonalization-based methods seem like an over-kill. There are techniques which avoid diagonalization.. ["Order n" methods,..]

- Main ingredient used: obtain charge densities by other means than using eigenvectors, e.g. exploit density matrix $\rho(r, r')$

- Filtering ideas – approximation theory ideas, ..

Some Approaches

$$P = f(H)$$

where f is a step function. Approximate f by, e.g., a polynomial

- Result: can obtain columns of P inexpensively via:

$$Pe_j \approx p_k(H)e_j$$

- Exploit sparsity of P (especially in planewave basis)- ideas of "probing" allow to compute several columns of P at once.

- Statistical approach: well-known ideas on estimating traces of a matrix adapted to estimate diagonals

Example 1 : Statistical approach

- Let a sequence of random vectors $v^1, \ldots, v^s$ with entries satisfying a normal distribution. Diagonal of a matrix B can be approximated by

$$D^s = \left[\sum_{k=1}^{s} v^k \odot Bv^k\right] \oslash \left[\sum_{k=1}^{s} v^k \odot v^k\right]$$

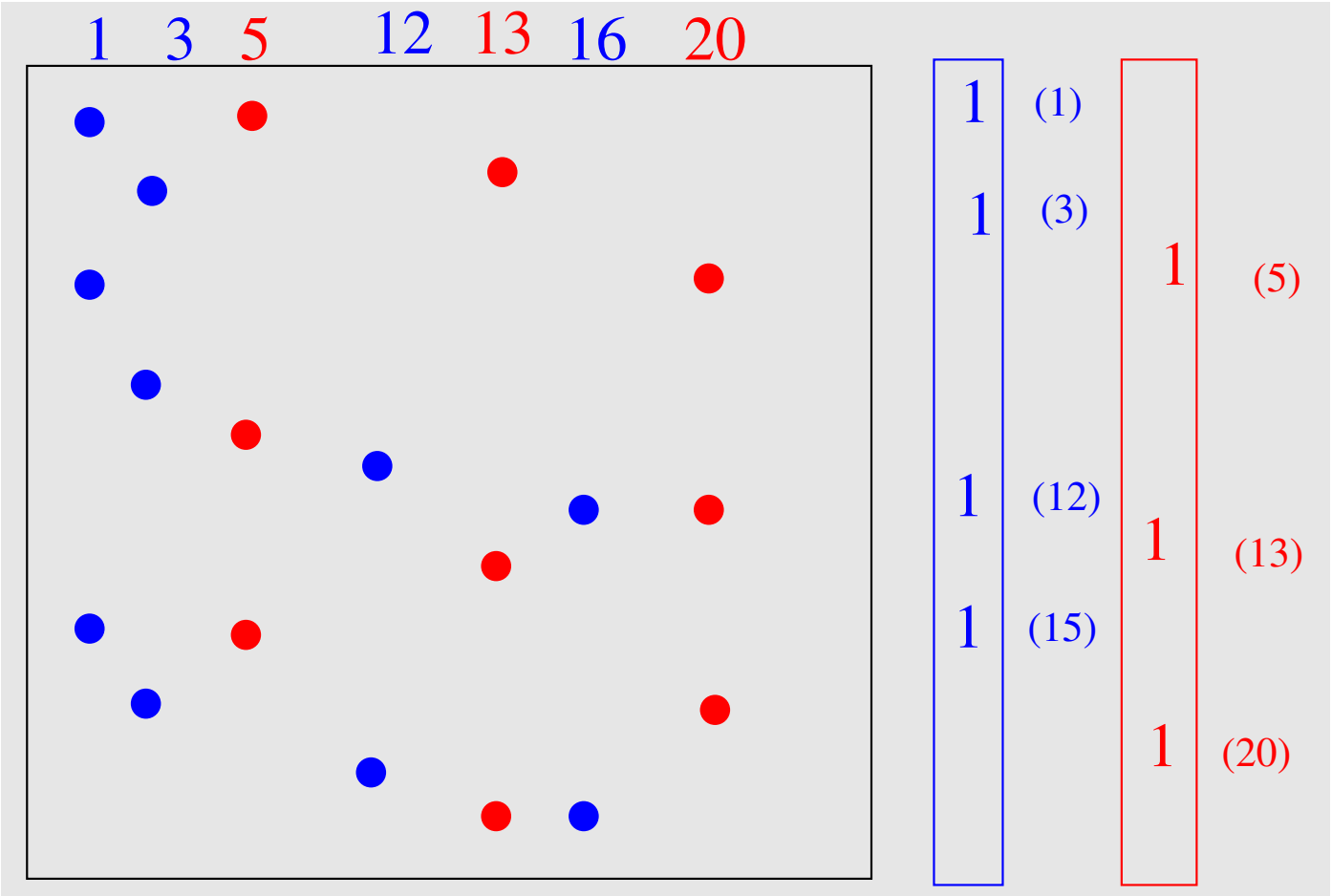
in which $\odot$ is a componentwise product of vectors, and similarly $\oslash$ represents a componentwise division of vectors.

- Deterministic approach: For a banded matrix (bandwidth p), there exists p vectors such the above formula yields the exact diagonal.

- These methods would require computing $p_k(H)v$ for several v's. Generally: method is expensive unless bandwith is small.

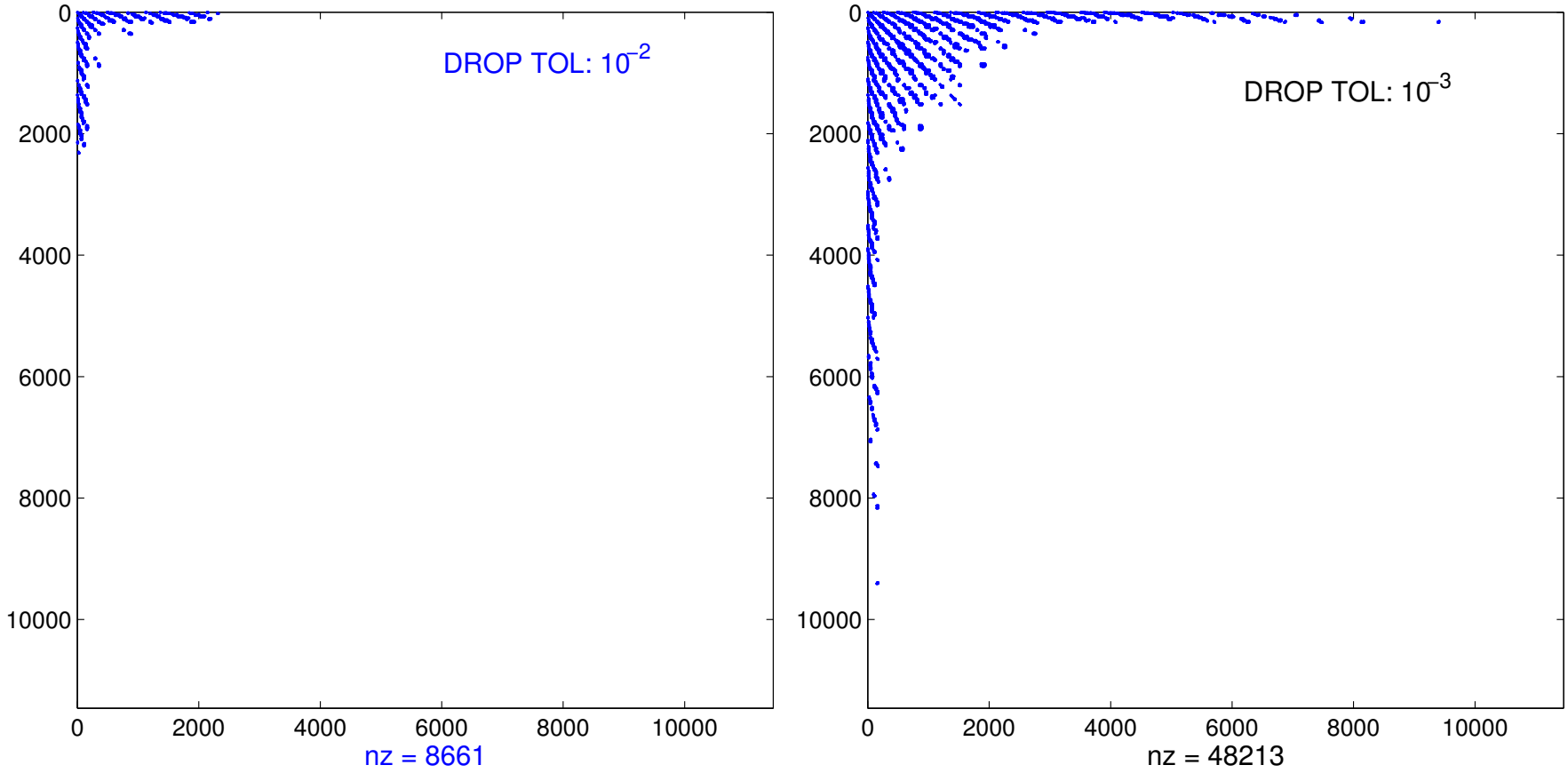
Example 2: density matrix in PW basis

- **Recall** $P = f(H) = \{\rho(r, r')\}$
- **Consider the expression in the** G**-basis**
- **Most entries are small –**
- **Idea: use technique of "probing" or "CPR" or "Sparse Jacobian" estimators ....**



Probing in action: blue columns can be computed at once by one matrix-vector product. Then red columns can be coumputed the same way

Density matrix for Si64 Using dropping of 0.01 and 0.001

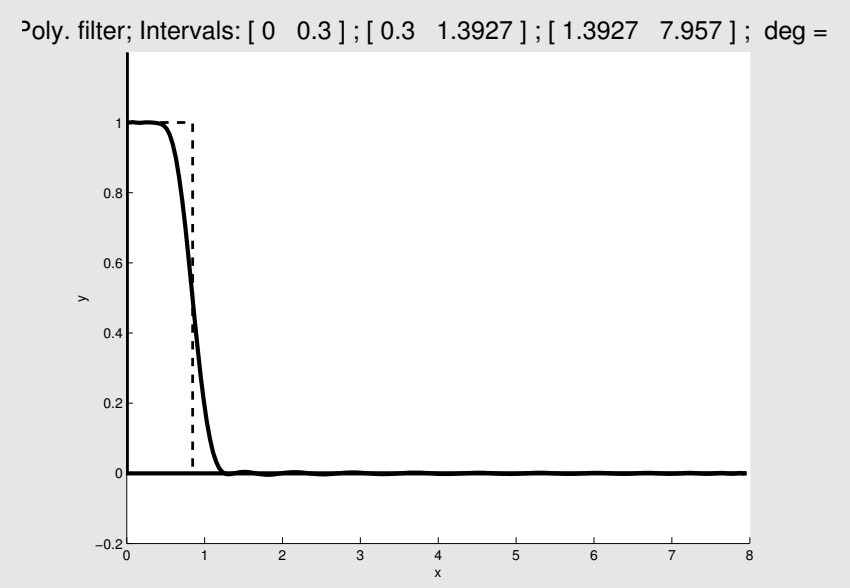


Summary & Conlusion

- Good progress made by shifting emphasis from eigenvectors to subspaces

- Next big step: completely avoid diagonalization ['linear scaling' methods w. density matrix formalism]

- Use of better polynomials?



- Also important: better 'mixing' methods..

- **My URL:**

URL: http://www.cs.umn.edu/∼saad

- **My e-mail address:**

e-mail: saad@cs.umn.edu

- **PARSEC's site:**

http://www.ices.utexas.edu/parsec/index.html

THANK YOU FOR YOUR ATTENTION!