Diagonalization algorithms in real space methods for electronic structure calculations

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

PMAA-06

IRISA, Rennes, Sept. 8th, 2006

Current & recent team members team: (CS side)

- Russ Burdick — Masters student - Scient. computing
- Shiv Gowda — Masters student - Scient. computing
- Emmanuel Lorin de La Grandmaison — Post-doc [Now at CRM,Montreal]
- Efi Kokiopoulou — PhD student – Now in Lausanne
- Costas Bekas — Post-Doc – Now in Lausanne
- Yunkai Zhou — Post-Doc – Now in Southern Methodist Univ.
- Susanne Shontz — Post-Doc – (Penn State)
- Prakash Dayal — Post-Doc
- Adam Jundt — PhD Student, Comp. Science

▶▶ Plus many from Jim Chelikowsky's team (Materials science side)

Main issues of interest:

- Solve the eigenvalue problem efficiently [specificity: large number of eigenvalues]
- Find alternatives [avoid eigenvectors, eigenvalues]
- Solve various related computational problems [TDDFT, computation of dielectric matrix, ...]

Today's talk:

- General discussion on real-space methods
- PARSEC (code)
- Techniques for diagonalization + some results

Electronic structure and Schrödinger's equation

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

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

- Solution via the many-body Shrödinger 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.

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

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

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

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

Several approximations/theories used

- Born-Oppenheimer approximation: Neglect motion of nuclei [Much heavier than electrons]
- Replace many electrons by one electron systems: each electron sees only average potentials from other particles
- Density Functional Theory [Hohenberg-Kohn '65]: Observables determined by ground state charge density
- Consequence: An equation of the form

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + v_0(r) + \int \frac{\rho(r')}{|r-r'|}dr' + \frac{\delta E_{xc}}{\delta \rho}\right]\Psi = E\Psi$$

- v_0 = external pot., E_{xc} = exchange-correlation energy

Density Function Theory - Kohn-Sham Eqns.

▶▶ Result of Density Functional Theory [Hohenberg-Kohn, Kohn-Sham]:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{tot}[\rho(r), r]\right]\Psi(r) = E\Psi(r)$$

With

$$V_{tot} = V_{ion} + V_H + V_{xc}$$

- V_H = Hartree potential — local
- V_{xc} = Exchange & Correlation potential — local (LDA)
- V_{ion} = Ionic potential — Non-local

▶▶ Electron Density:

$$\rho(r) = \Sigma_i^{occup} |\Psi_i(r)|^2$$

Kohn-Sham equations → nonlinear eigenvalue Pb

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{tot}[\rho(r)]\right]\Psi_i(r) = E_i\Psi_i(r), i = 1, ..., i^o$$

$$V_{tot} = V_H + V_{xc} + V_{ion}$$

$$\rho(r) = \sum_i^o |\Psi_i(r)|^2$$

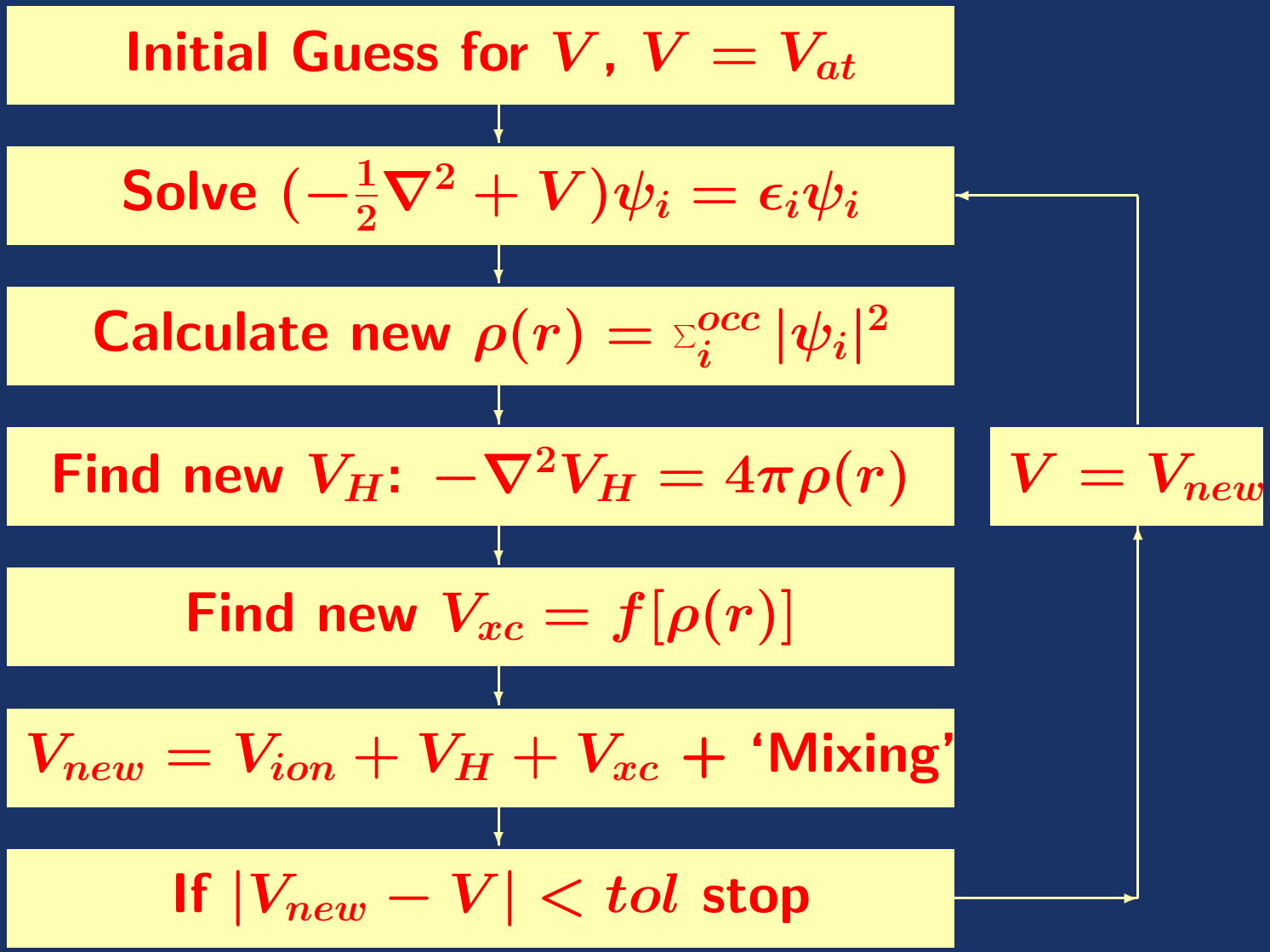
$$\nabla^2 V_H = -4\pi\rho(r)$$

- Both V_{xc} and V_H, depend on ρ.
- Potentials & charge densities must be **self-consistent.**

Can be viewed as a nonlinear eigenvalue problem

- Broyden-type quasi-Newton technique used
- Typically, a small number of iterations are required

Self-Consistent Iteration

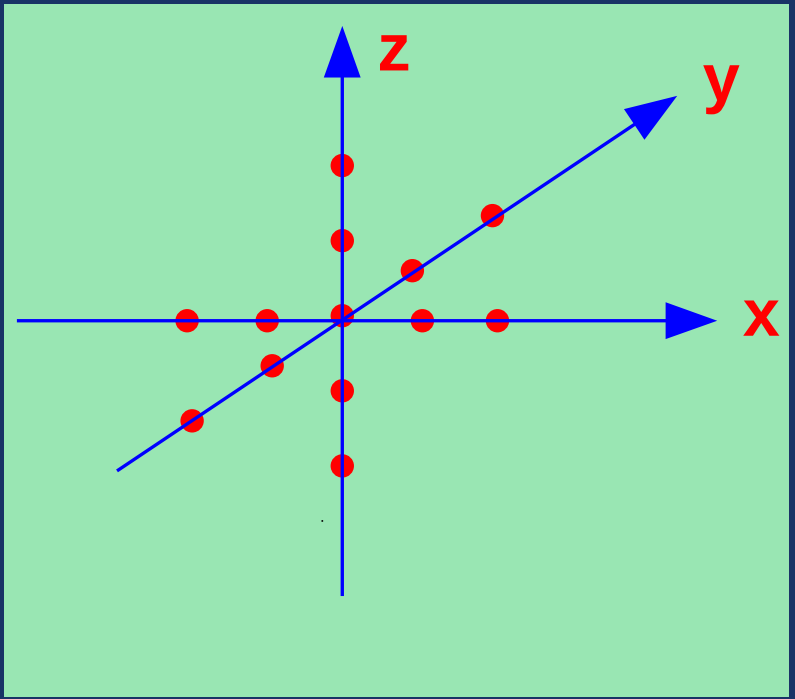


- Most time-consuming part: **diagonalization**
- Difficulty: large number of wanted eigenvalues/-vectors [number of occupied states].
- In this situation, cost of typical diagonalization codes dominated by orthogonalization

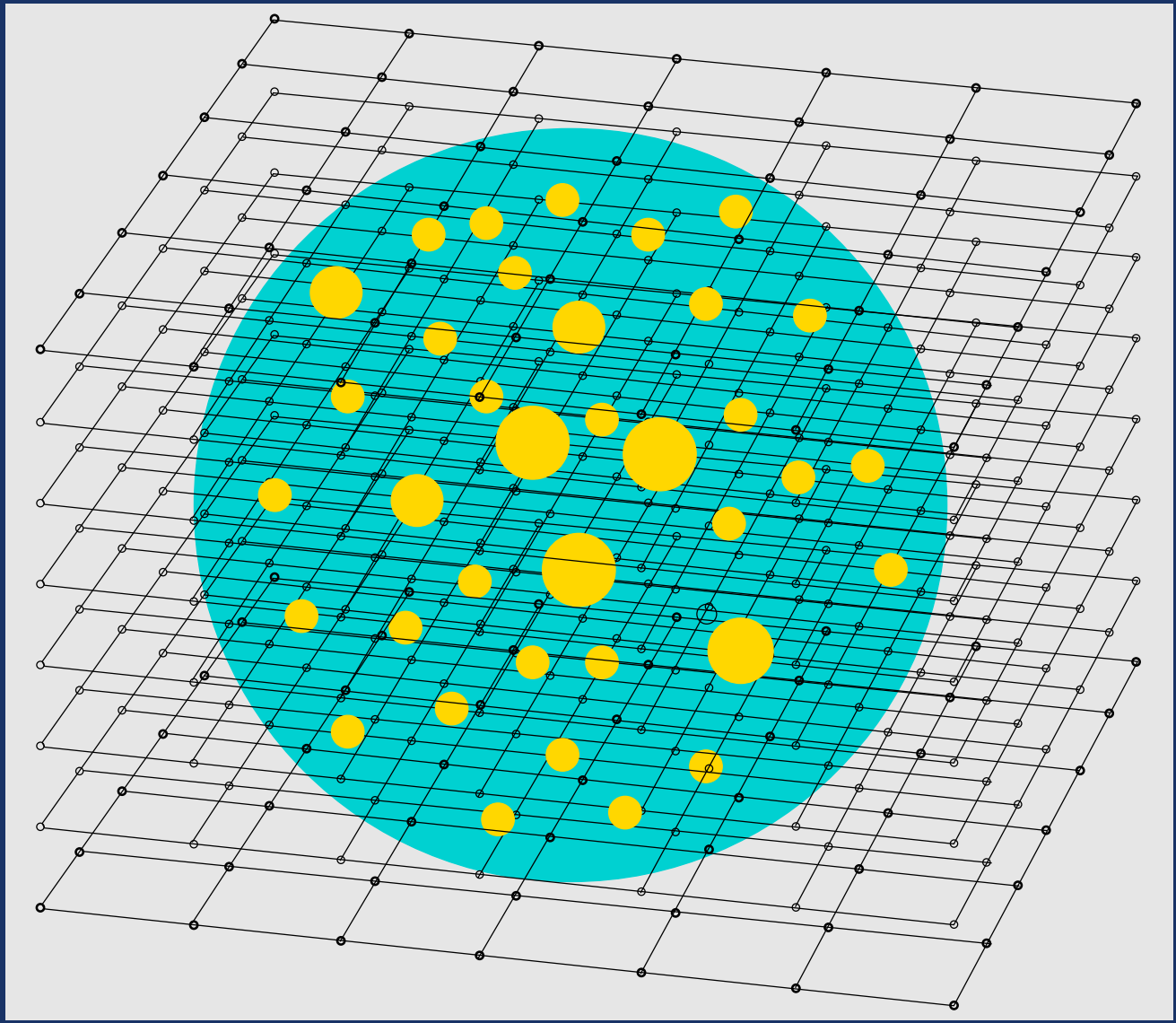
Real-space Finite Difference Methods

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

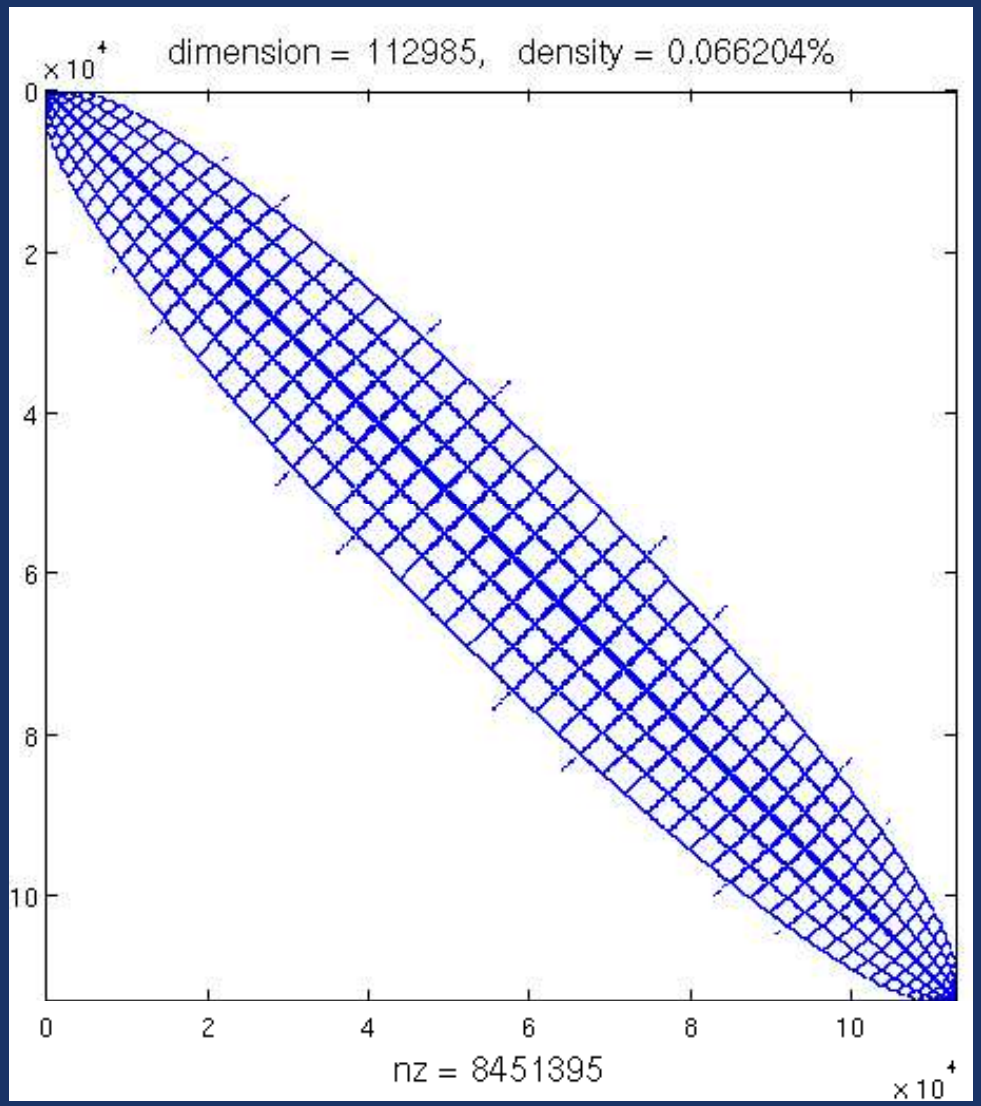
Order 4 Finite Difference Approximation:



The physical domain



Pattern of resulting matrix for Ge99H100:



PARSEC - A few milestones

- **PARSEC** = Pseudopotential Algorithm for Real Space Electronic Calculations
- 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
- Current: SGI Altix, IBM SP4 – PARSEC released

Sample calculations done: Quantum dots

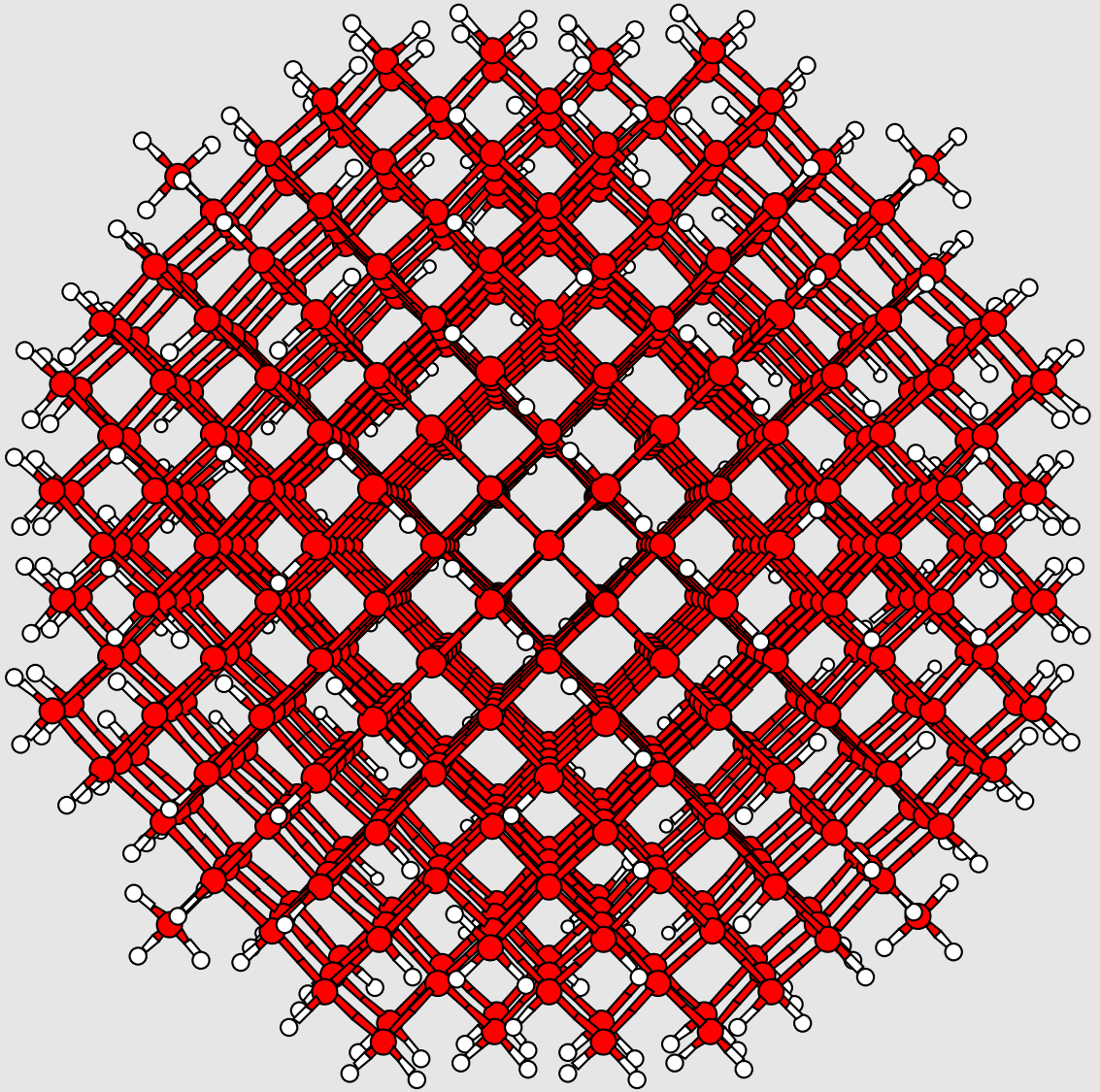
▶▶ Small silicon clusters ($\approx 20 - 100 Angst$. Involve up to a few hundreds atoms)

- $Si_{525}H_{276}$ 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

$Si_{525}H_{276}$



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 eigen-values, ...

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

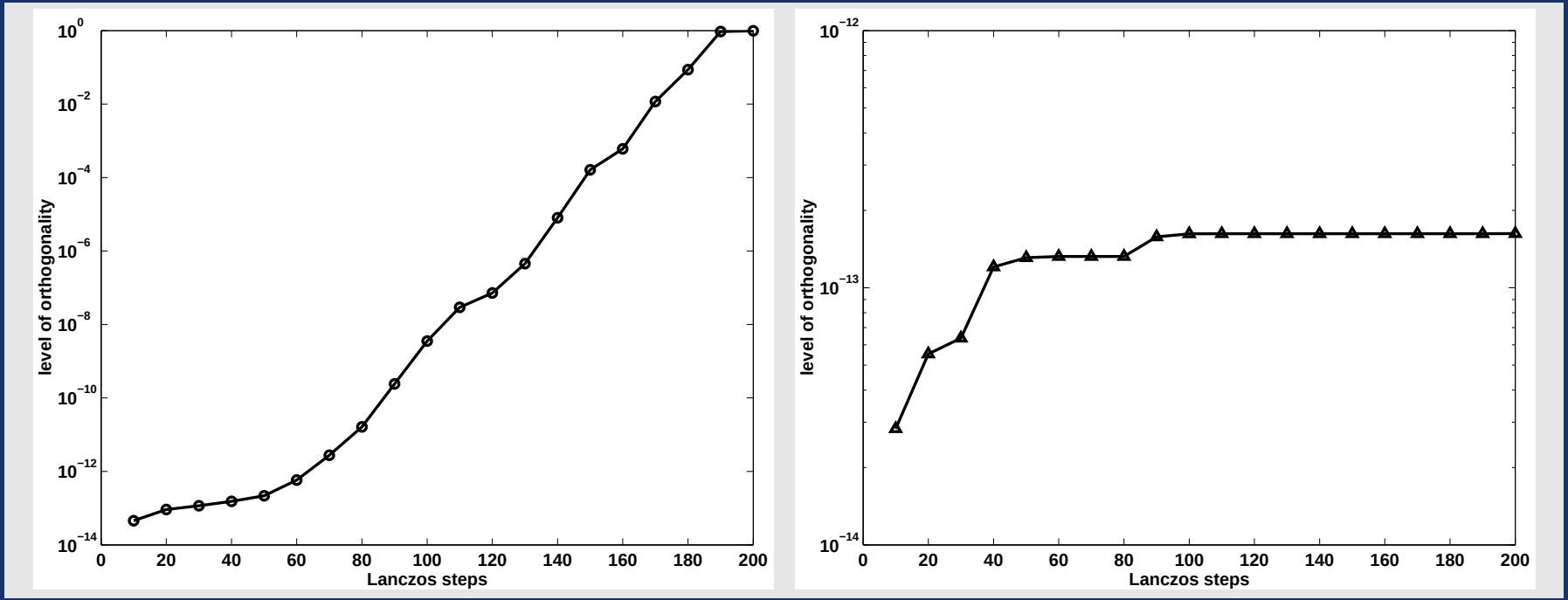
Ingredients: (1) test of loss of orthogonality (recurrence relation) and (2) stopping criterion based on charge density instead of eigenvectors.

- Recall that 3-term recurrence in theory suffices to generate orthonormal basis of Krylov subspace (Lanczos algorithm). But: severe loss of orthogonality as soon as 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 reorthogonalization 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 Subspace iteration

- Main ingredient: Chebyshev filtering

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

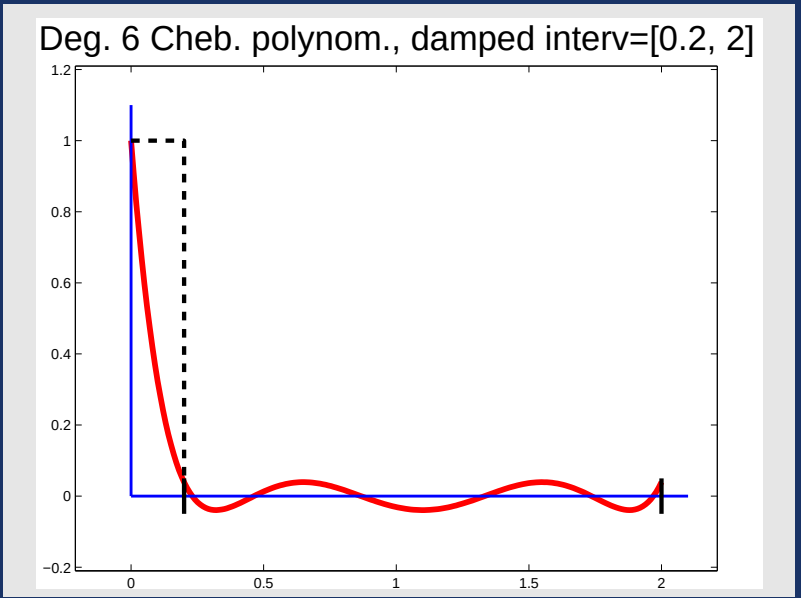
$$\hat{v}_i = P_k(A)v_i$$

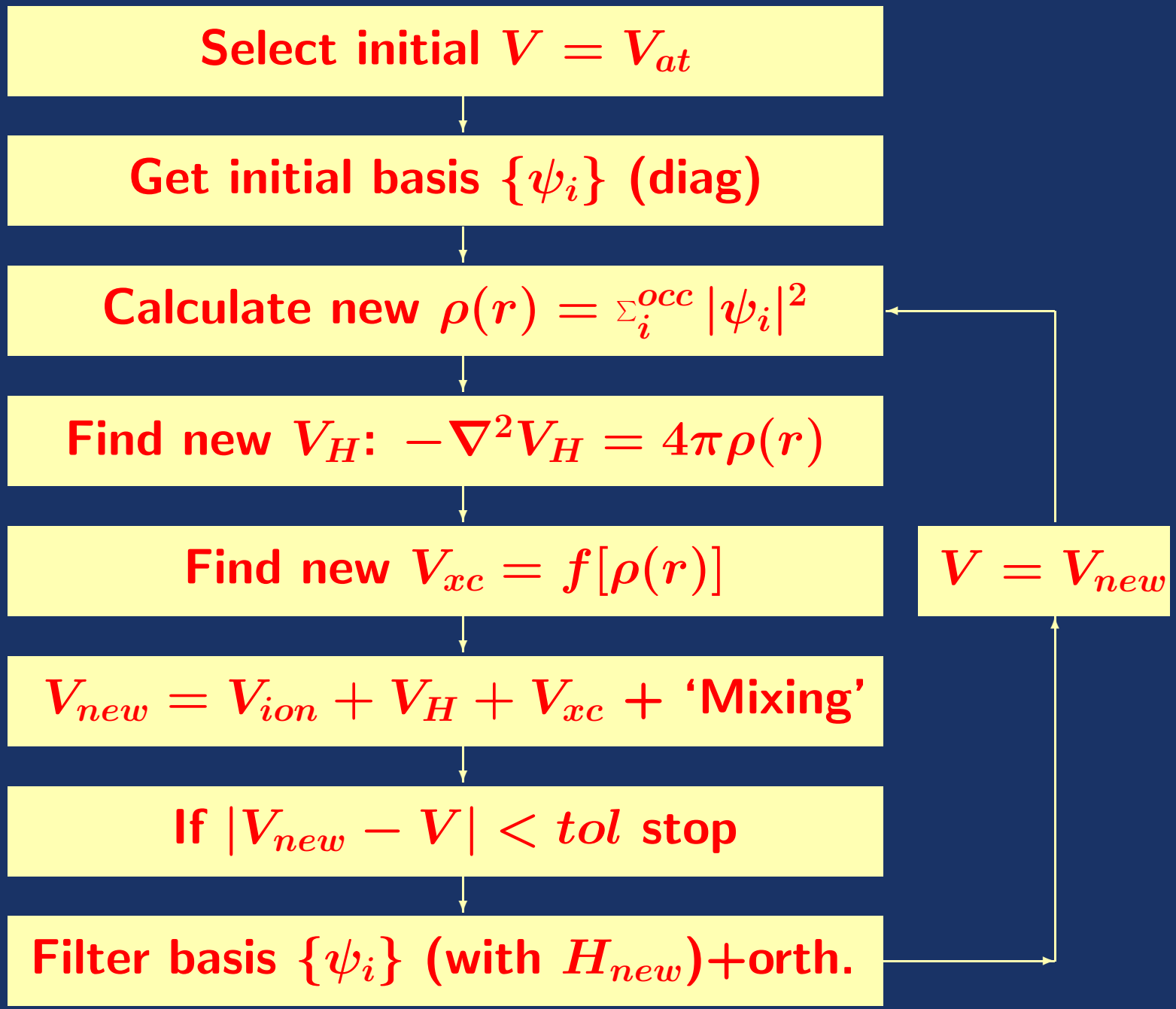
- p_k = Polynomial of low degree: enhances desired eigen-components

The filtering step is not used to compute eigenvectors accurately

SCF & diagonization loops merged

Important: convergence still good and robust




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, **Self-Consistent-Field Calculations with Chebyshev Filtered Subspace Iteration,** *Minnesota Supercomputer Institute, tech. report., Oct. 2005.*

[See http://www.cs.umn.edu/∼saad]

Chebyshev Subspace iteration - experiments

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

$Si_{525}H_{276}$, $n_H = 292,584$, $n_{state} = 1194$ 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

$Si_{65}Ge_{65}H_{98}$, $n_H = 185,368$, $n_{state} = 313$ Polynomial degree used is 8. Total energies same to within 9 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. $n_H = 874,976$, $n_{state} = 520 \times 2$, Total energies same to within ~ 5 digits.

Some recent results

▶▶ Recent tests with *Large* systems (never attempted before). Some are still running as we speak

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]

$Si_{6047}H_{1308}$

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

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

$Si_{9041}H_{1860}$

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

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_{360}

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
2160×2	12989799	146	-795.22329	16.55 hrs.	140.68 hrs.

Fe_{360}, #PEs = 24. $n_H = 3262312$; $m = 20$ for Chebyshev-Davidson; $m = 17$ for CheFSI.

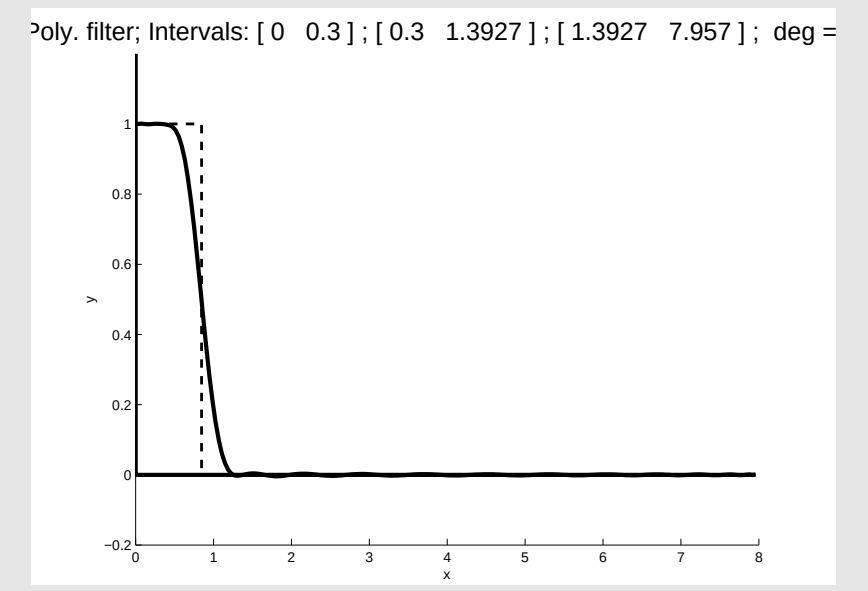
Fe_{388}

n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
2328×2	18232215	187	-795.24763	16.22(*) hrs.	247.05 hrs.

Fe_{388} #PE= 24. $n_H = 3332856$. $m = 20$ for Chebyshev-Davidson; $m = 18$ for CheFSI.

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!