UNIVERSITY OF MINNESOTA TWIN CITIES

Sparse Matrix Methods and Applications

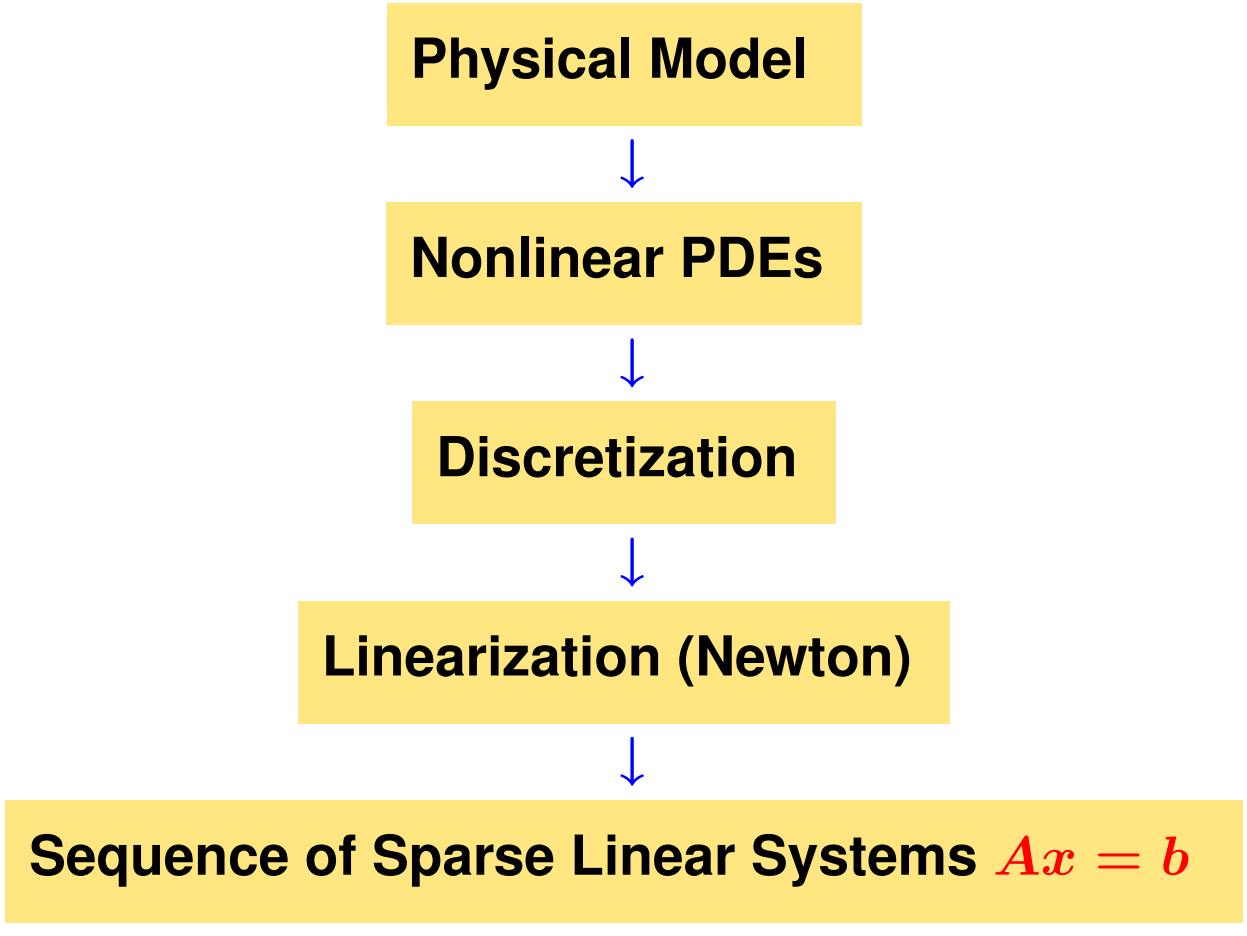
Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

Wimereux, April 1st, 2008

Typical Problem:



What are sparse matrices?

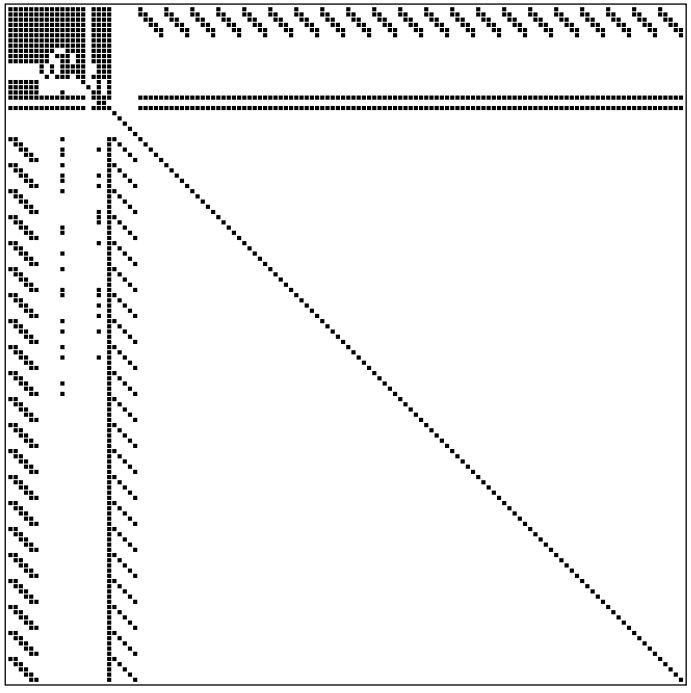
Common definition: "..matrices that allow special techniques to take advantage of the large number of zero elements and the structure."

A few applications of sparse matrices : Structural Engineering, Reservoir simulation, Electrical Networks, optimization problems, ...

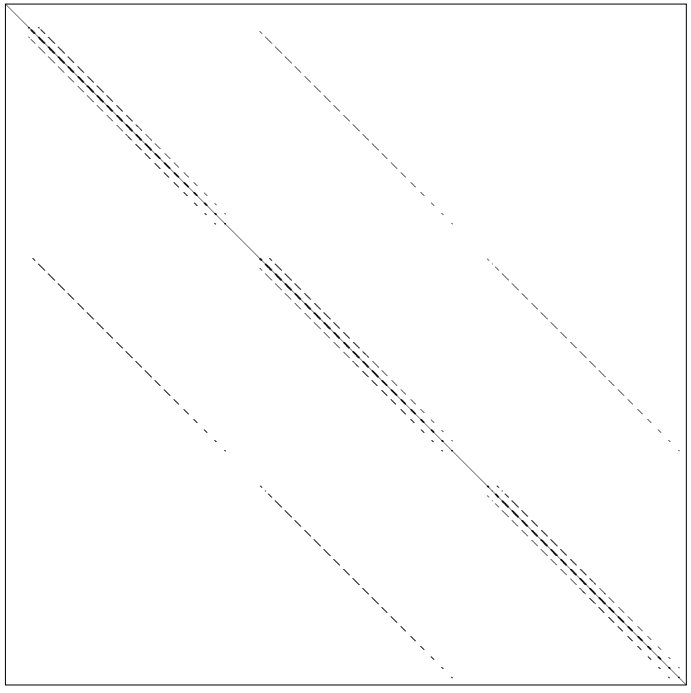
Goals: Much less storage and work than dense computations.

Observation: A^{-1} is usually dense, but L and U in the LU factorization may be reasonably sparse (if a good technique is used).

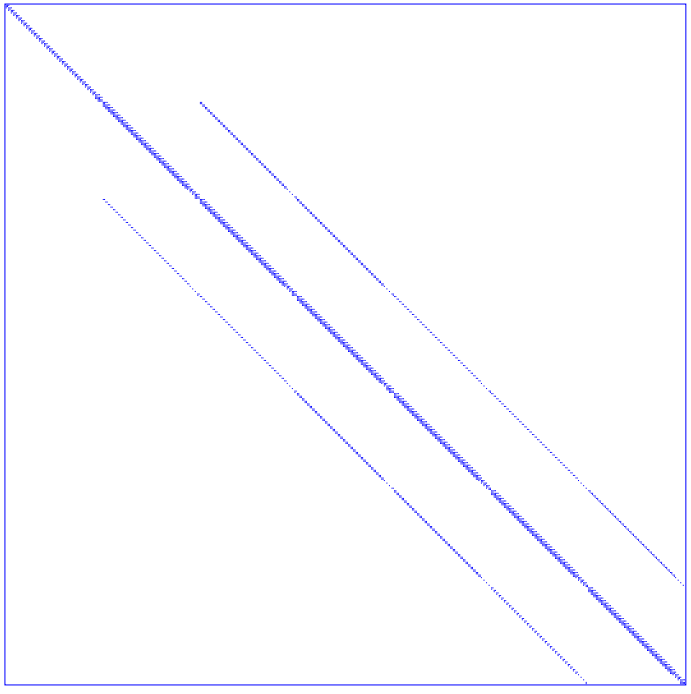
Nonzero patterns of a few sparse matrices



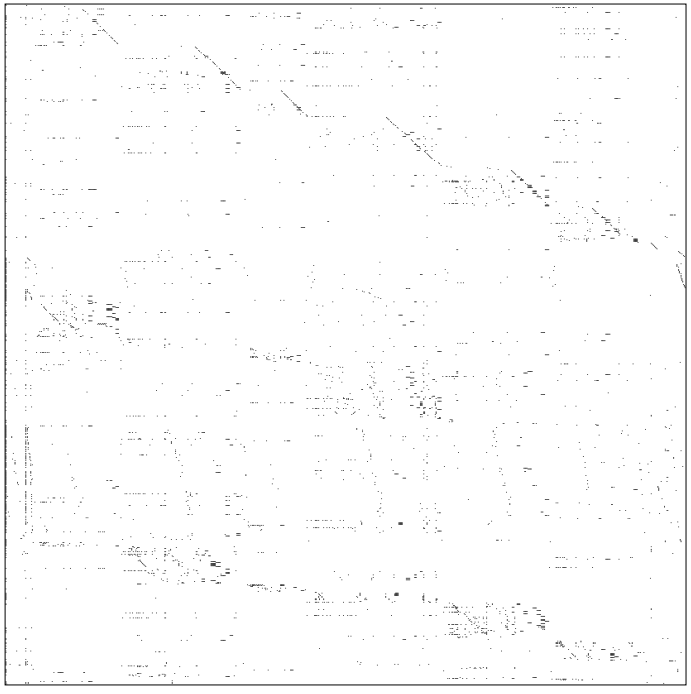
ARC130: Unsymmetric matrix from laser problem. a.r.curtis, oct 1974



SHERMAN5: fully implicit black oil simulator 16 by 23 by 3 grid, 3 unk



PORES3: Unsymmetric MATRIX FROM PORES



BP_1000: UNSYMMETRIC BASIS FROM LP PROBLEM BP

➤ **Two types of matrices:** structured (e.g. Sherman5) and unstructured (e.g. BP_1000)

➤ **Main goal of Sparse Matrix Techniques:** To perform standard matrix computations economically i.e., without storing the zeros of the matrix.

➤ **Example:** To add two square dense matrices of size n requires $O(n^2)$ operations. To add two sparse matrices A and B requires $O(nnz(A) + nnz(B))$ where $nnz(X) =$ number of nonzero elements of a matrix X.

➤ For typical Finite Element /Finite difference matrices, number of nonzero elements is $O(n)$.

PROJECTION METHODS FOR LINEAR SYSTEMS

Projection Methods

Initial Problem: $Ax = b$ **A is large and sparse**

Given two subspaces K and L of $\mathbb{R}^N$ define the *approximate problem:*

Find $\tilde{x} \in K$ such that $b - A\tilde{x} \perp L$

- **Result: a small linear system ('projected problems')**
- **With a nonzero initial guess x_0, the approximate problem is**

Find $\tilde{x} \in x_0 + K$ **such that** $b - A\tilde{x} \perp L$

Write $\tilde{x} = x_0 + \delta$ and $r_0 = b - Ax_0$. Leads to a system for δ:

Find $\delta \in K$ such that $r_0 - A\delta \perp L$

Matrix representation:

Let

- $V = [v_1, \ldots, v_m]$ **a basis of** K **&**
- $W = [w_1, \ldots, w_m]$ **a basis of** L

Then letting x **be the approximate solution** $\tilde{x} = x_0 + \delta \equiv x_0 + Vy$ **where** y **is a vector of** $\mathbb{R}^m$**, the Petrov-Galerkin condition yields,**

$$W^T(r_0 - AVy) = 0$$

and therefore

$$\tilde{x} = x_0 + V[W^T AV]^{-1} W^T r_0$$

Remark: In practice $W^T AV$ **is known from algorithm and has a simple structure [tridiagonal, Hessenberg,..]**

Prototype Projection Method

Until Convergence Do:

1. Select a pair of subspaces K, and L;
2. Choose bases $V = [v_1, \ldots, v_m]$ for K and $W = [w_1, \ldots, w_m]$ for L.
3. Compute

$$r \leftarrow b - Ax,$$

$$y \leftarrow (W^T A V)^{-1} W^T r,$$

$$x \leftarrow x + Vy.$$

Two important particular cases.

1. $L = AK$. then $\|b - A\tilde{x}\|_2 = \min_{z \in K} \|b - Az\|_2$

   $\longrightarrow$ class of minimal residual methods: CR, GCR, ORTHOMIN, GMRES, CGNR, ...

2. $L = K$ $\longrightarrow$ class of Galerkin or orthogonal projection methods.

   When A is SPD then

$$\|x^* - \tilde{x}\|_A = \min_{z \in K} \|x^* - z\|_A.$$

One-dimensional projection processes

$$K = span\{d\}$$
and
$$L = span\{e\}$$

Then $\tilde{x} \leftarrow x + \alpha d$ and Petrov-Galerkin condition $r - A\delta \perp e$ yields

$$\boxed{\alpha = \frac{(r,e)}{(Ad,e)}}$$

Three popular choices:

(I) Steepest descent.

(II) Minimal residual iteration.

(III) Residual norm steepest descent . [Same as (I) for $A^T Ax = A^T b$]

(I) **Steepest descent.** A is SPD. Take at each step $d = r$ and $e = r$.

Iteration:

$$\begin{aligned} r &\leftarrow b - Ax, \\ \alpha &\leftarrow (r, r)/(Ar, r) \\ x &\leftarrow x + \alpha r \end{aligned}$$

- Each step minimizes

$$f(x) = \|x - x^*\|_A^2 = (A(x - x^*), (x - x^*))$$

in direction $-\nabla f$. Convergence guaranteed if A is SPD.

(II) Minimal residual iteration. A positive definite ($A + A^T$ is SPD).

Take at each step $d = r$ and $e = Ar$.

Iteration:

$$
\begin{aligned}
&r \leftarrow b - Ax, \\
&\alpha \leftarrow (Ar, r)/(Ar, Ar) \\
&x \leftarrow x + \alpha r
\end{aligned}
$$

- Each step minimizes $f(x) = \|b - Ax\|_2^2$ in direction r.
- Converges under the condition that $A + A^T$ is SPD.

Krylov Subspace Methods

Principle: Projection methods on Krylov subspaces:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$$

- probably the most important class of iterative methods.
- many variants exist depending on the subspace L.

Simple properties of K_m. Let $\mu =$ deg. of minimal polynomial of v

- $K_m = \{p(A)v | p =$ polynomial of degree $\leq m-1\}$
- $K_m = K_\mu$ for all $m \geq \mu$. Moreover, K_μ is invariant under A.
- $dim(K_m) = m$ iff $\mu \geq m$.

Arnoldi's Algorithm

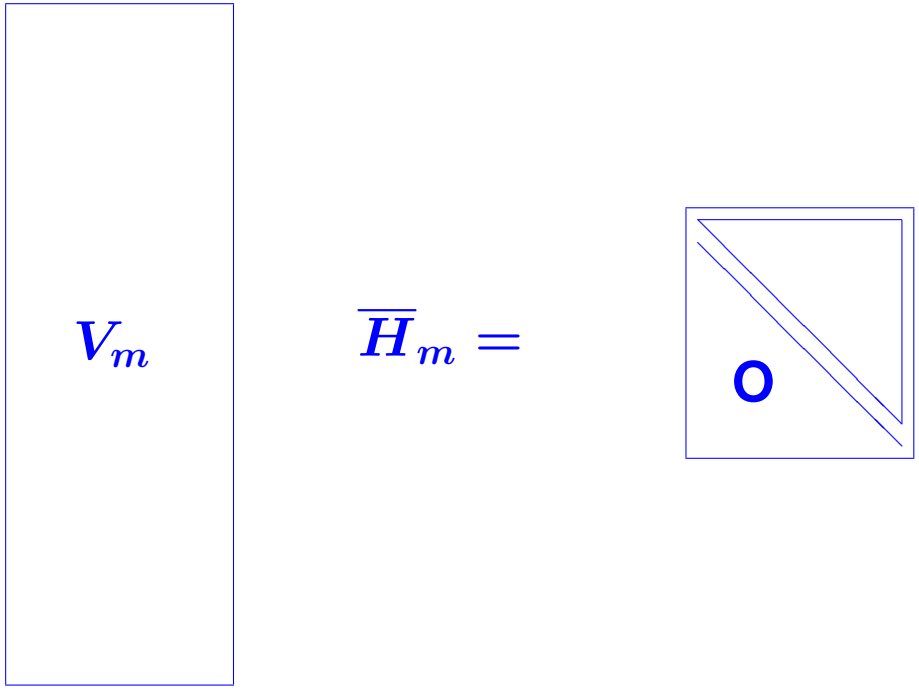
- **Goal: to compute an orthogonal basis of K_m.**
- **Input: Initial vector v_1, with $\|v_1\|_2 = 1$ and m.**

For $j = 1, ..., m$ do

- **Compute $w := Av_j$**
- **for $i = 1, \ldots, j$, do** $\begin{cases} h_{i,j} := (w, v_i) \\ w := w - h_{i,j} v_i \end{cases}$
- **$h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$**

Result of orthogonalization process

1. $V_m = [v_1, v_2, ..., v_m]$ **orthonormal basis of** K_m**.**
2. $AV_m = V_{m+1}\overline{H}_m$
3. $V_m^T A V_m = H_m \equiv \overline{H}_m -$ **last row.**



Arnoldi's Method ($L_m = K_m$)

From Petrov-Galerkin condition when $L_m = K_m$, we get

$$x_m = x_0 + V_m H_m^{-1} V_m^T r_0$$

If, in addition we choose $v_1 = r_0/\|r_0\|_2 \equiv r_0/\beta$ in Arnoldi's algorithm, then

$$\boxed{x_m = x_0 + \beta V_m H_m^{-1} e_1}$$

Several algorithms mathematically equivalent to this approach:

* FOM [YS, 1981] (above formulation)

* Young and Jea's ORTHORES [1982].

* Axelsson's projection method [1981].

Minimal residual methods ($L_m = AK_m$)

When $L_m = AK_m$, we let $W_m \equiv AV_m$ and obtain:

$$\begin{aligned} x_m &= x_0 + V_m[W_m^T A V_m]^{-1} W_m^T r_0 = \\ &= x_0 + V_m[(AV_m)^T A V_m]^{-1}(AV_m)^T r_0. \end{aligned}$$

Use again $v_1 := r_0/(\beta := \|r_0\|_2)$ and: $AV_m = V_{m+1}\bar{H}_m$

$$x_m = x_0 + V_m[\bar{H}_m^T \bar{H}_m]^{-1}\bar{H}_m^T \beta e_1 = x_0 + V_m y_m$$

where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ over $y \in \mathbb{R}^m$. Therefore, (Generalized Minimal Residual method (GMRES) [Saad-Schultz, 1983]):

$$x_m = x_0 + V_m y_m \quad \textbf{where} \quad y_m : \min_y \|\beta e_1 - \bar{H}_m y\|_2$$

Equivalent methods:

- **Axelsson's CGLS**
- **Orthomin (1980)**
- **Orthodir**
- **GCR**

EIGENVALUE PROBLEMS

Origins of Eigenvalue Problems

- **Structural Engineering [$Ku = \lambda Mu$]**
- **Electronic structure calculations [Shrödinger equation..]**
- **Stability analysis [e.g., electrical networks, mechanical system,..]**
- **Bifurcation analysis [e.g., in fluid flow]**

➤ **Large sparse eigenvalue problems are among the most demanding calculations (in terms of CPU time) in scientific computing.**

New applications in information technology

- **Search engines (google) rank web-sites in order to improve searches**
- **The google toolbar on some browsers (http://toolbar.google.com) - gives a measure of relevance of a page.**
- **The problem can be formulated as a Markov chain – Seek the dominant eigenvector**
- **Algorithm used: power method**
- **For details see:**

`http://www.iprcom.com/papers/pagerank/index.html`

The Problem

We consider the eigenvalue problem

$$Ax = \lambda x \text{ or } Ax = \lambda Bx$$

Typically: B is symmetric (semi) positive definite, A is symmetric or nonsymmetric

Requirements vary:

- Compute a few λ_i 's with smallest or largest real parts;
- Compute all λ_i's in a certain region of $\mathbb{C}$;
- Compute a few of the dominant eigenvalues;
- Compute all λ_i's.

Types of problems

* **Standard Hermitian (or symmetric real)** $\boxed{Ax = \lambda x}$, $A^H = A$

* **Standard non-Hermitian** $\boxed{Ax = \lambda x}$, $A^H \neq A$

* **Generalized**

$$Ax = \lambda Bx$$

Several distinct sub-cases (B SPD, B SSPD, B singular with large null space, both A and B singular, etc..)

* **Quadratic**

$$(A + \lambda B + \lambda^2 C)x = 0$$

* **Nonlinear**

$$A(\lambda)x = 0$$

A few popular solution Methods

- **Subspace Iteration [Now less popular – sometimes used for validation]**
- **Arnoldi's method (or Lanczos) with polynomial acceleration [Stiefel '58, Rutishauser '62, YS '84,'85, Sorensen '89,...]**
- **Shift-and-invert and other preconditioners. [Use Arnoldi or Lanczos for $(A - \sigma I)^{-1}$.]**
- **Davidson's method and variants, Generalized Davidosn's method [Morgan and Scott, 89], Jacobi-Davidsion**
- **Emerning method: Automatic Multilevel Substructuring (AMLS).**

Projection Methods for Eigenvalue Problems

General formulation:

Projection method onto K orthogonal to L

- **Given: Two subspaces K and L of same dimension.**
- **Find: $\tilde{\lambda}, \tilde{u}$ such that**

$$\tilde{\lambda} \in \mathbb{C}, \tilde{u} \in K; \quad (\tilde{\lambda} I - A)\tilde{u} \perp L$$

Two types of methods:

Orthogonal projection methods: situation when $L = K$.

Oblique projection methods: When $L \neq K$.

Rayleigh-Ritz projection

Given: a subspace X known to contain good approximations to eigenvectors of A.

Question: How to extract good approximations to eigenvalues/ eigenvectors from this subspace?

Answer: Rayleigh Ritz process.

Let $Q = [q_1, \ldots, q_m]$ an orthonormal basis of X. Then write an approximation in the form $\tilde{u} = Qy$ and obtain y by writing

$$Q^H(A - \tilde{\lambda}I)\tilde{u} = 0$$

- $Q^H A Q y = \tilde{\lambda} y$

Subspace Iteration

- **Original idea: projection technique onto a subspace if the form**

$$Y = A^k X$$

- **In practice: Replace A^k by suitable polynomial [Chebyshev]**

Advantages:
- **Easy to implement (in symmetric case);**
- **Easy to analyze;**

Disadvantage: Slow.

- **Often used with polynomial acceleration: $A^k X$ replaced by $C_k(A)X$.**

Typically C_k = Chebyshev polynomial.

KRYLOV SUBSPACE METHODS

Krylov Subspace Methods

Principle: Projection methods on Krylov subspaces, i.e., on

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$$

- **probably the most important class of projection methods [for linear systems and for eigenvalue problems]**
- **many variants exist depending on the subspace L.**

Properties of K_m. Let $\mu =$ deg. of minimal polynom. of v. Then,

- $K_m = \{p(A)v | p =$ **polynomial of degree** $\leq m-1\}$
- $K_m = K_\mu$ **for all** $m \geq \mu$**. Moreover,** K_μ **is invariant under** A**.**
- $dim(K_m) = m$ **iff** $\mu \geq m$**.**

Arnoldi's Algorithm

- **Goal: to compute an orthogonal basis of K_m.**
- **Input: Initial vector v_1, with $\|v_1\|_2 = 1$ and m.**

ALGORITHM : 1. ***Arnoldi's procedure***

For $j = 1, ..., m$ ***do***

Compute $w := Av_j$

For $i = 1, \ldots, j$, ***do*** $\begin{cases} h_{i,j} := (w, v_i) \\ w := w - h_{i,j} v_i \end{cases}$

$h_{j+1,j} = \|w\|_2$. $v_{j+1} = w / h_{j+1,j}$

End

Result of Arnoldi's algorithm

Let

$$\overline{H}_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{pmatrix} \qquad H_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{pmatrix}$$

1. $V_m = [v_1, v_2, ..., v_m]$ **orthonormal basis of** K_m**.**
2. $AV_m = V_{m+1}\overline{H}_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$
3. $V_m^T A V_m = H_m \equiv \overline{H}_m -$ **last row.**

Appliaction to eigenvalue problems

- **Write approximate eigenvector as $\tilde{u} = V_m y$ + Galerkin condition**

$$(A - \tilde{\lambda} I) V_m y \;\perp\; \mathcal{K}_m \rightarrow V_m^H (A - \tilde{\lambda} I) V_m y = 0$$

- **Approximate eigenvalues are eigenvalues of H_m**

$$\boldsymbol{H_m y_j = \tilde{\lambda}_j y_j}$$

Associated approximate eigenvectors are

$$\boldsymbol{\tilde{u}_j = V_m y_j}$$

Typically a few of the outermost eigenvalues will converge first.

Restarted Arnoldi

In practice: Memory requirement of algorithm implies restarting is necessary

ALGORITHM : 2. **Restarted Arnoldi (computes rightmost eigenpair)**

1. Start: *Choose an initial vector v_1 and a dimension m.*
2. Iterate: *Perform m steps of Arnoldi's algorithm.*
3. Restart: *Compute the approximate eigenvector $u_1^{(m)}$*
4. *associated with the rightmost eigenvalue $\lambda_1^{(m)}$.*
5. *If satisfied stop, else set $v_1 \equiv u_1^{(m)}$ and goto 2.*

Example:

Small Markov Chain matrix [Mark(10) , dimension = 55]. Restarted Arnoldi procedure for computing the eigenvector associated with the eigenvalue with algebraically largest real part. We use $m = 10$.

m	$\Re(\lambda)$	$\Im(\lambda)$	Res. Norm
10	0.9987435899D+00	0.0	0.246D-01
20	0.9999523324D+00	0.0	0.144D-02
30	0.1000000368D+01	0.0	0.221D-04
40	0.1000000025D+01	0.0	0.508D-06
50	0.9999999996D+00	0.0	0.138D-07

Hermitian case: The Lanczos Algorithm

- **The Hessenberg matrix becomes tridiagonal :**

$$A = A^H \quad \text{and} \quad V_m^H A V_m = H_m \quad \rightarrow H_m = H_m^H$$

- **We can write**

$$H_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \beta_4 & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & & \beta_m & \alpha_m \end{pmatrix} \tag{1}$$

- **Consequence: three term recurrence**

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

1. **Choose an initial vector v_1 of norm unity. Set $\beta_1 \equiv 0, v_0 \equiv 0$**
2. **For $j = 1, 2, \ldots, m$ Do:**
3. $w_j := Av_j - \beta_j v_{j-1}$
4. $\alpha_j := (w_j, v_j)$
5. $w_j := w_j - \alpha_j v_j$
6. $\beta_{j+1} := \|w_j\|_2$. **If $\beta_{j+1} = 0$ then Stop**
7. $v_{j+1} := w_j / \beta_{j+1}$
8. **EndDo**

Hermitian matrix + Arnoldi $\longrightarrow$ Hermitian Lanczos

- **In theory v_i's defined by 3-term recurrence are orthogonal.**
- **However: in practice severe loss of orthogonality;**

Observation [Paige, 1981]: Loss of orthogonality starts suddenly, when the first eigenpair has converged. It is a sign of loss of linear indedependence of the computed eigenvectors. When orthogonality is lost, then several the copies of the same eigenvalue start appearing.

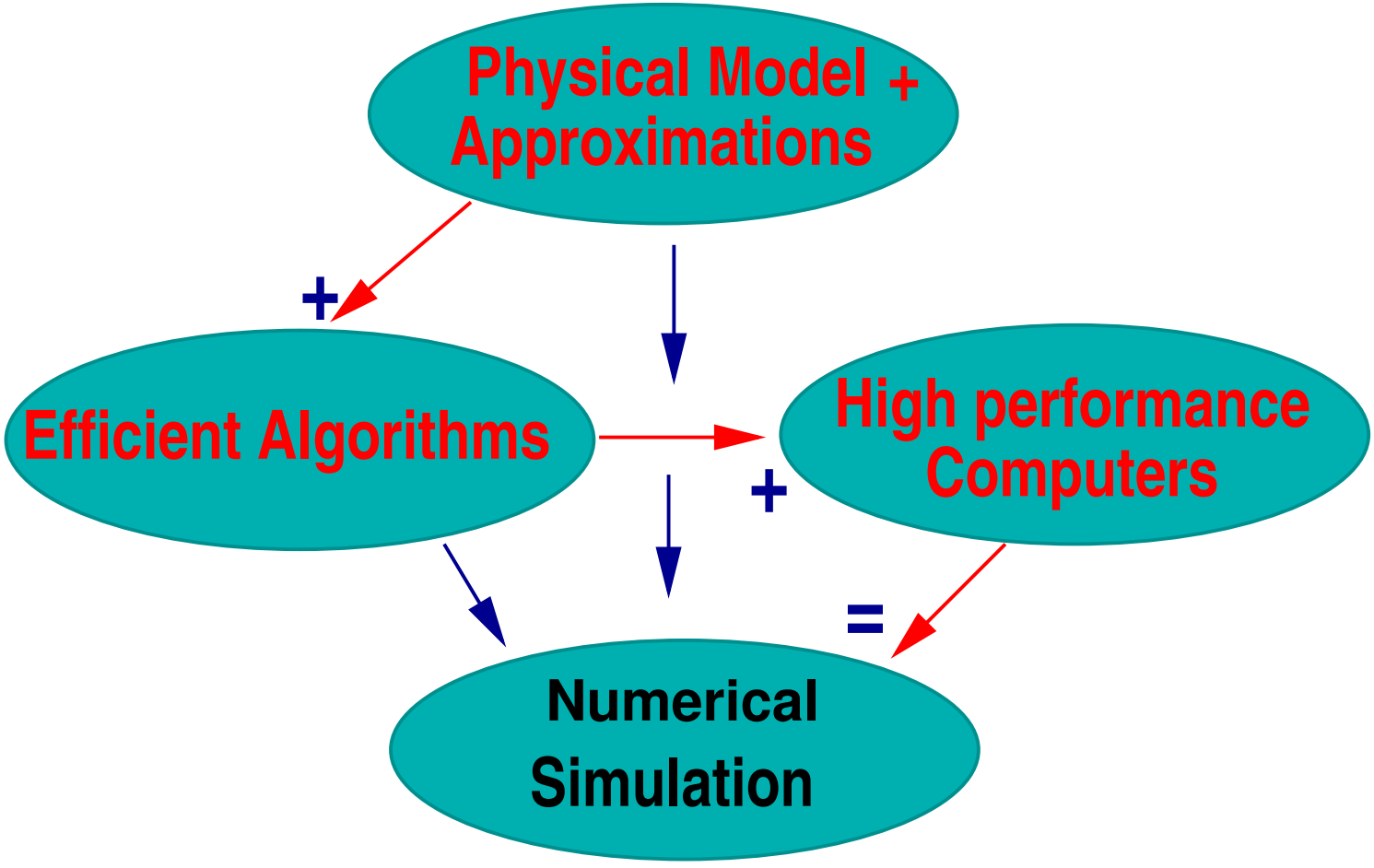
Reorthogonalization

- Full reorthogonalization – reorthogonalize v_{j+1} against all previous v_i's every time.
- Partial reorthogonalization – reorthogonalize v_{j+1} against all previous v_i's only when needed [Parlett & Simon]
- Selective reorthogonalization – reorthogonalize v_{j+1} against computed eigenvectors [Parlett & Scott]
- No reorthogonalization – Do not reorthogonalize - but take measures to deal with 'spurious' eigenvalues. [Cullum & Willoughby]

APPLICATIONS: ELECTRONIC STRUCTURE

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.
- 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

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

- **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 $\rho(r)$.**

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

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

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

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

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

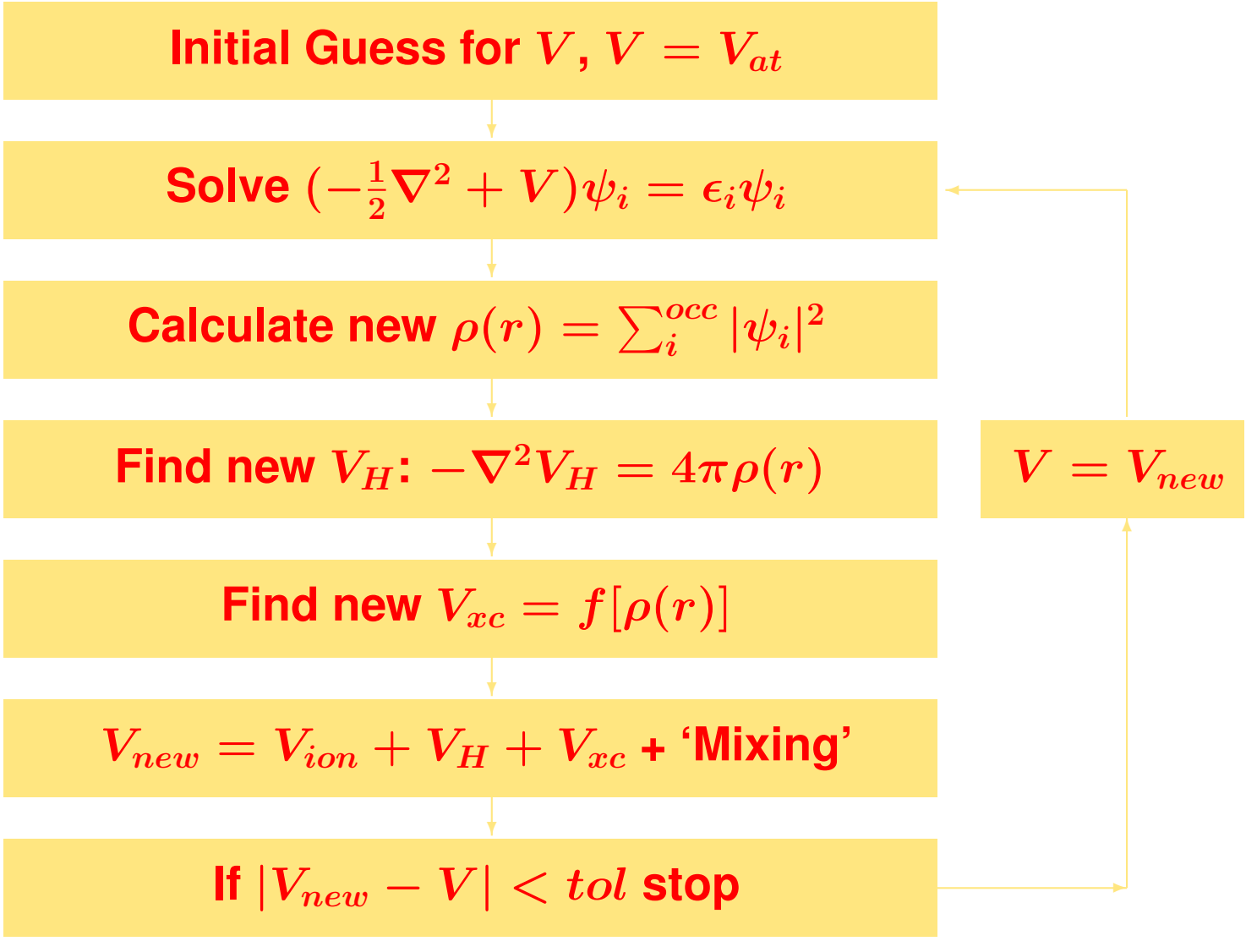
- Nonlinear eigenvalue problem: Linearize + iterate to self-consistence
- Nonlinear optimization: minimize energy [again linearize + achieve self-consistency]

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.**

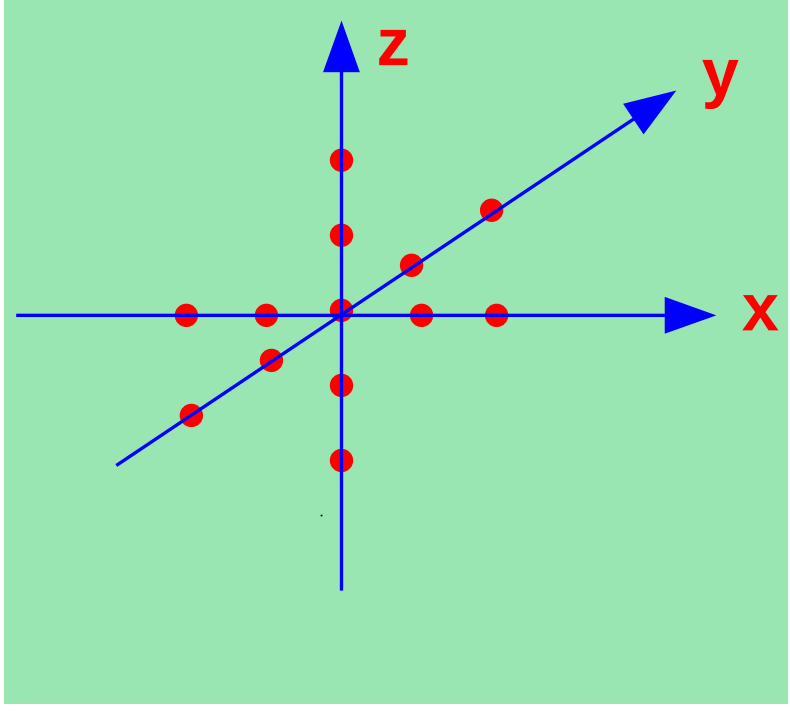
Characteristic : Large number of eigenvalues /-vectors to compute [occupied states]. For example Hamiltonian matrix size can be $N = 1,000,000$ and the number of eigenvectors about 1,000.

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

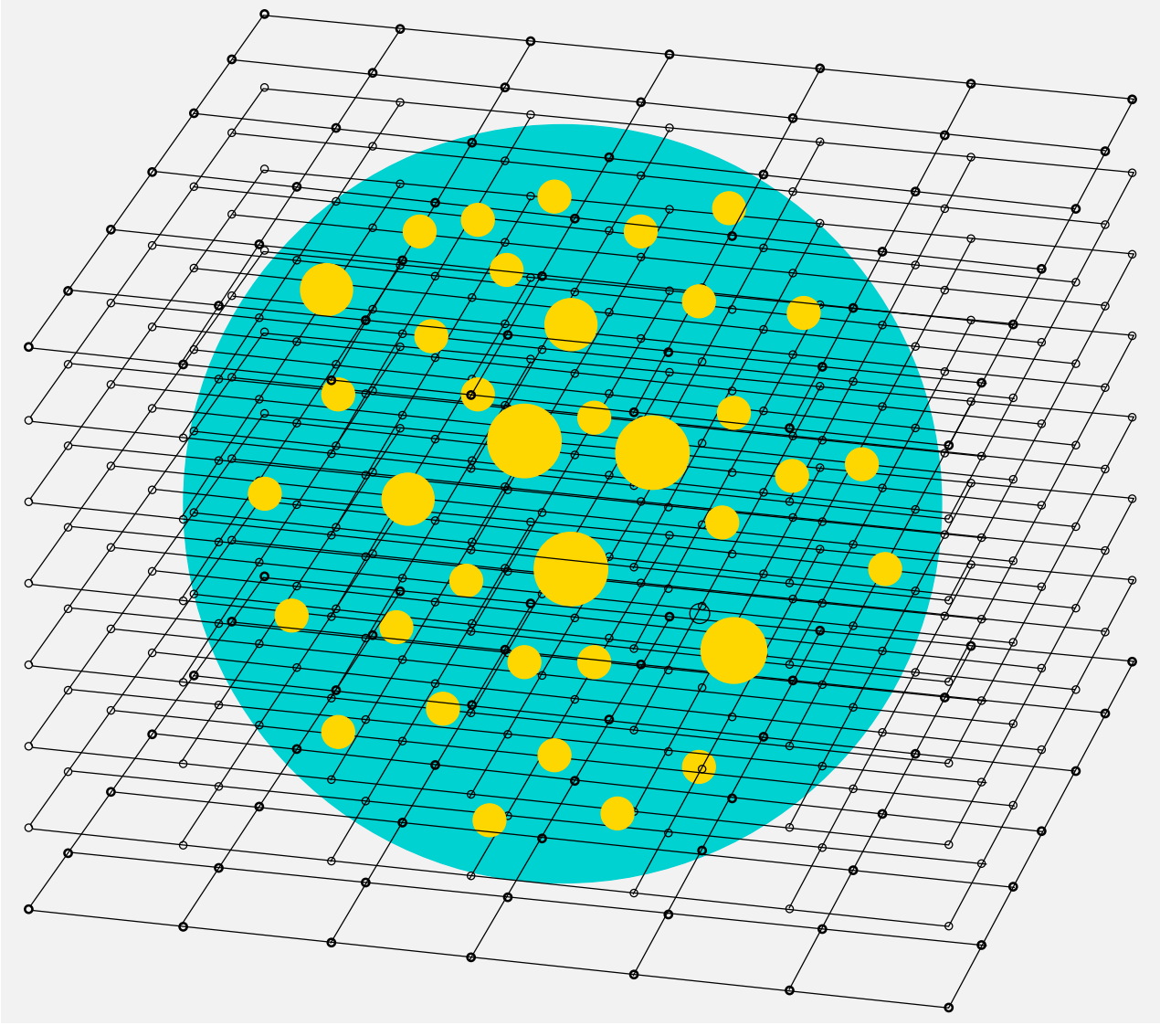
Real-space Finite Difference Methods

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

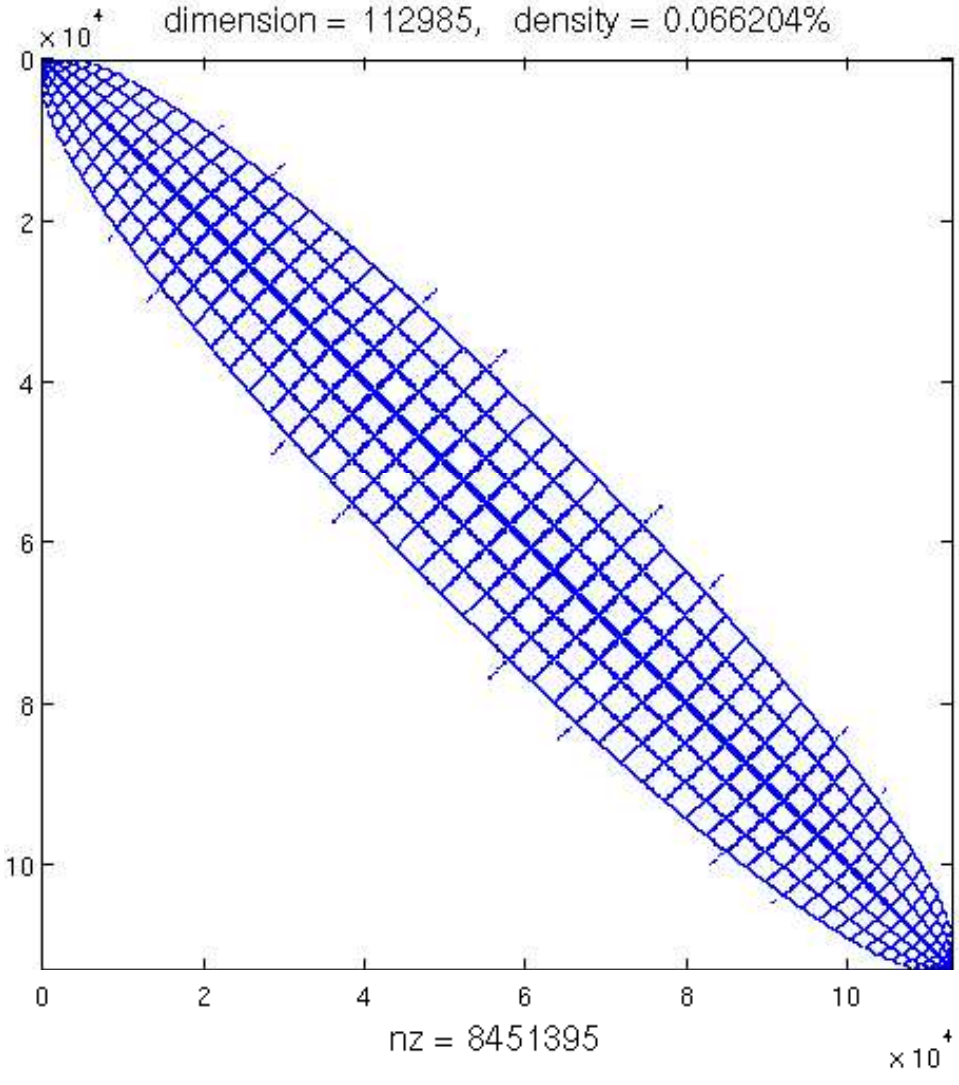
Order 4 Finite Difference Approximation:



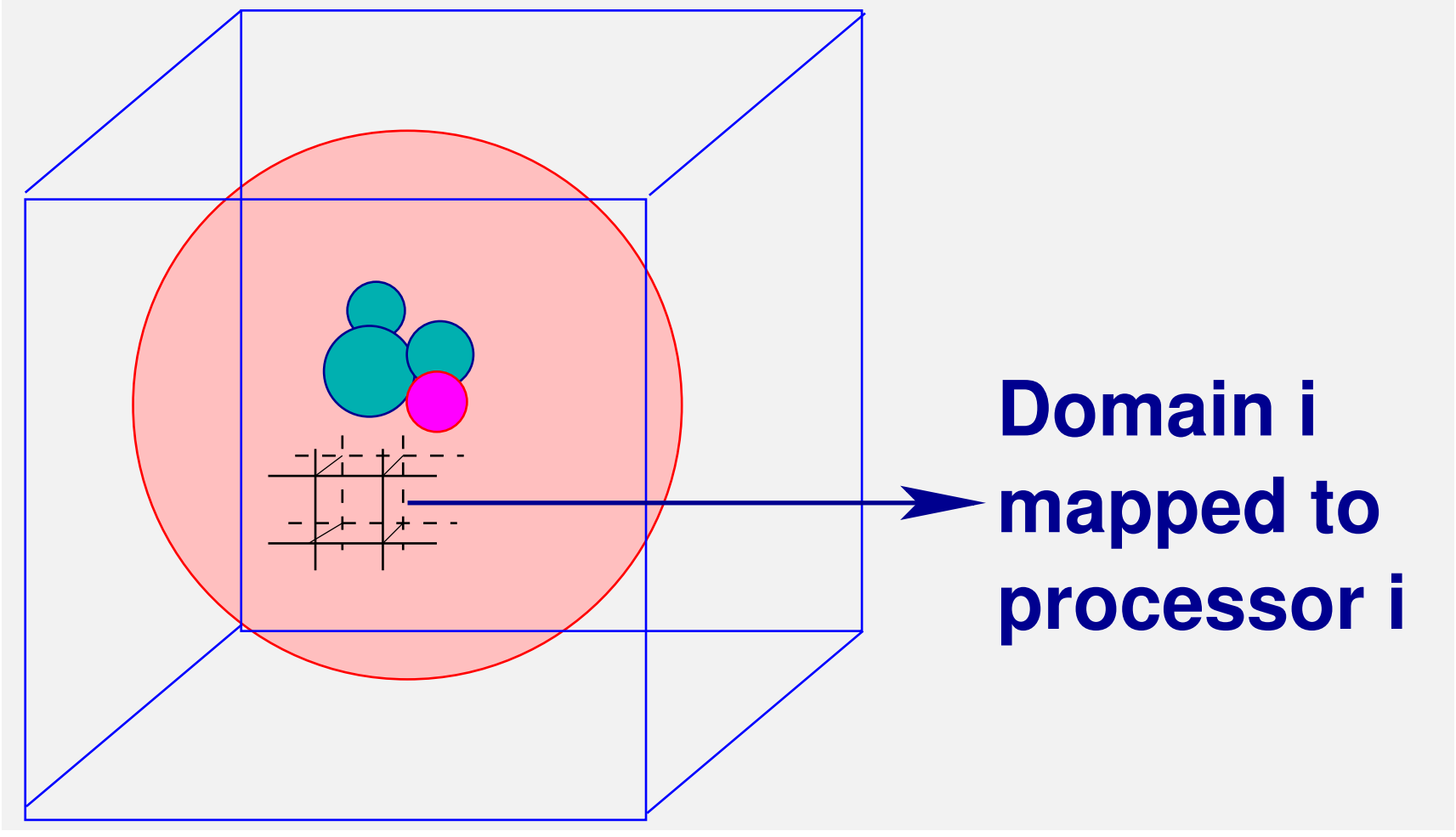
The physical domain



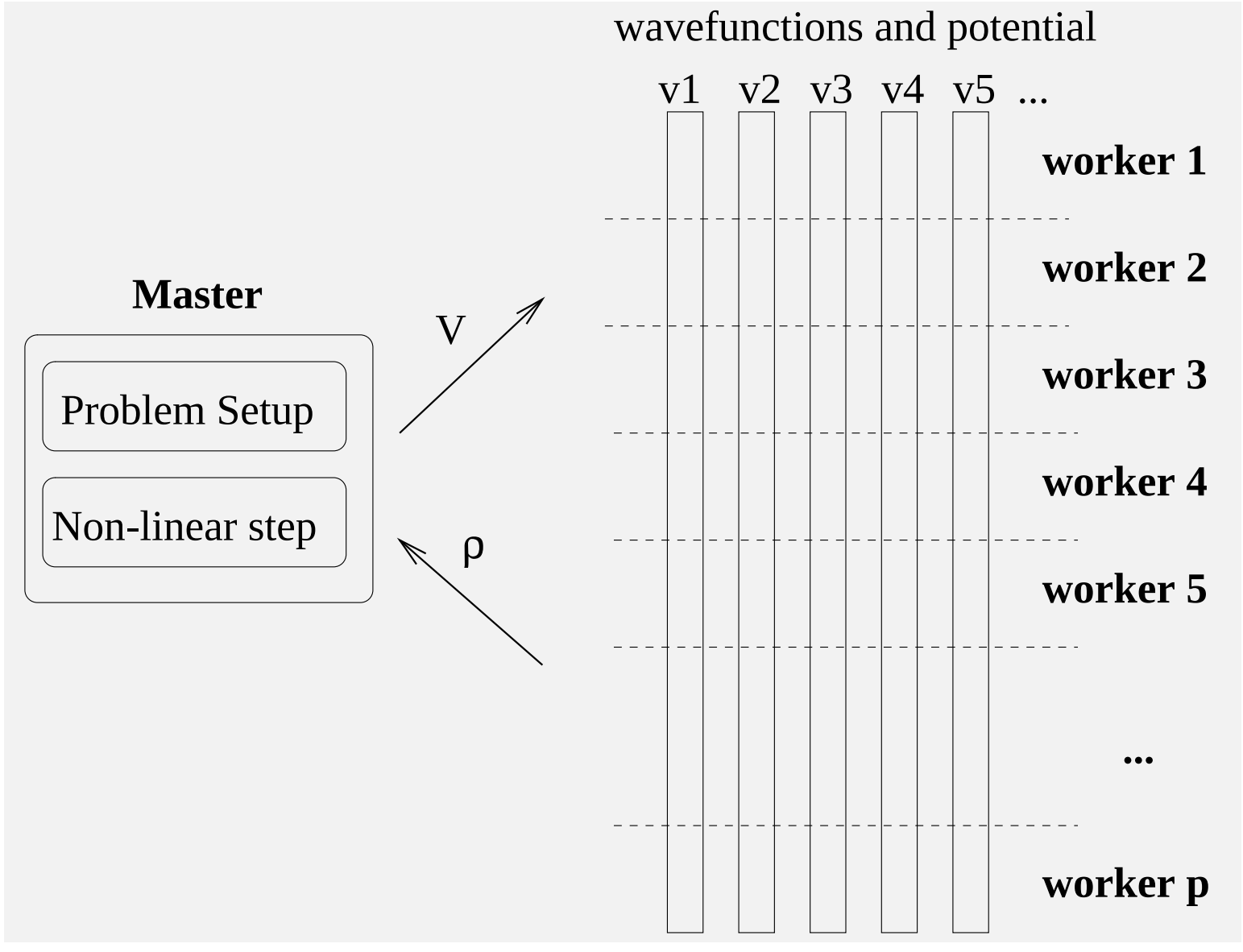
Pattern of resulting matrix for Ge99H100:



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

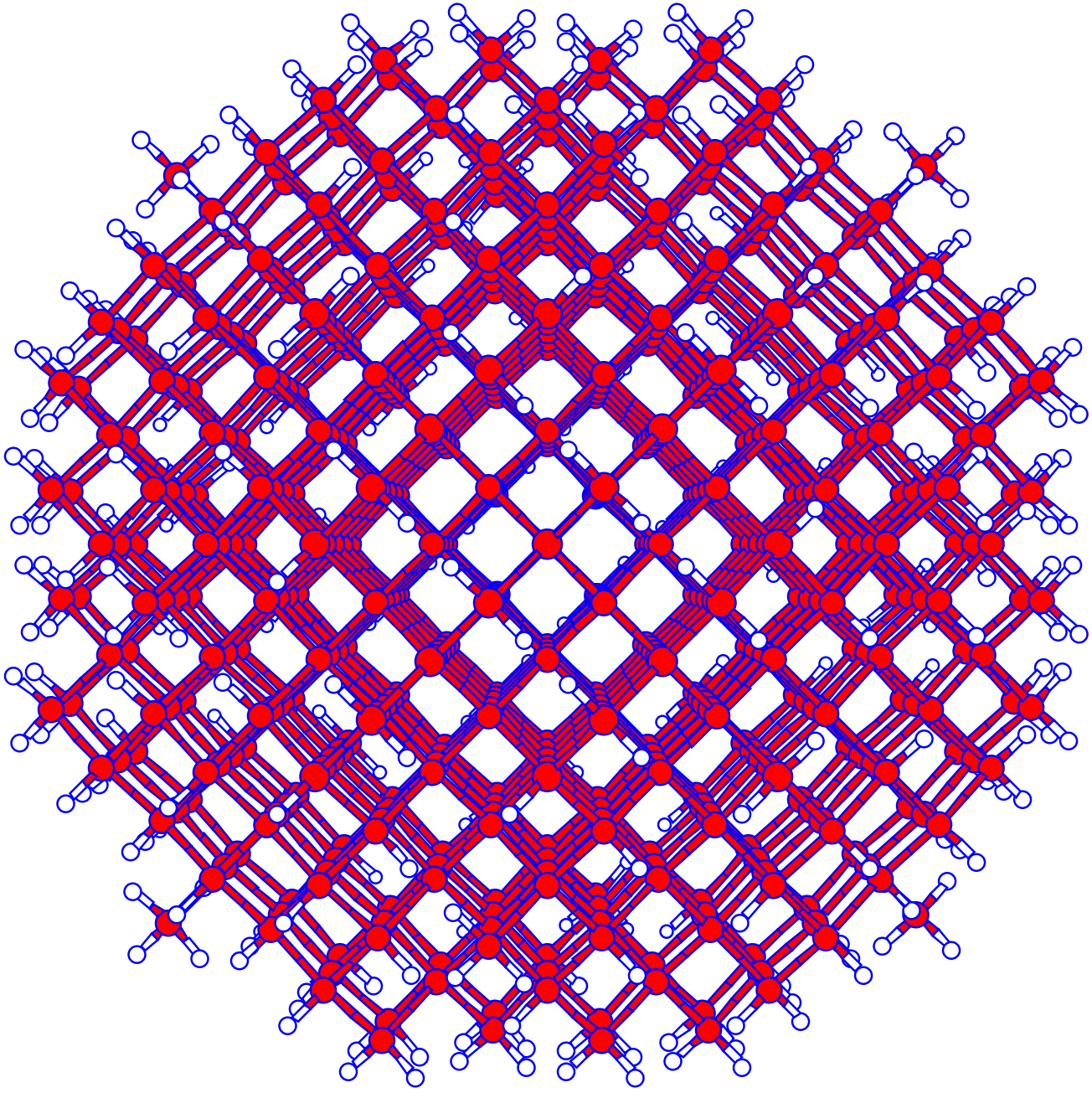
➤ **Small silicon clusters ($\approx 20 - 100 Angst$. Involve up to a few hundreds atoms)**

- **$Si_{525}H_{276}$ 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**

$Si_{525}H_{276}$



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:

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

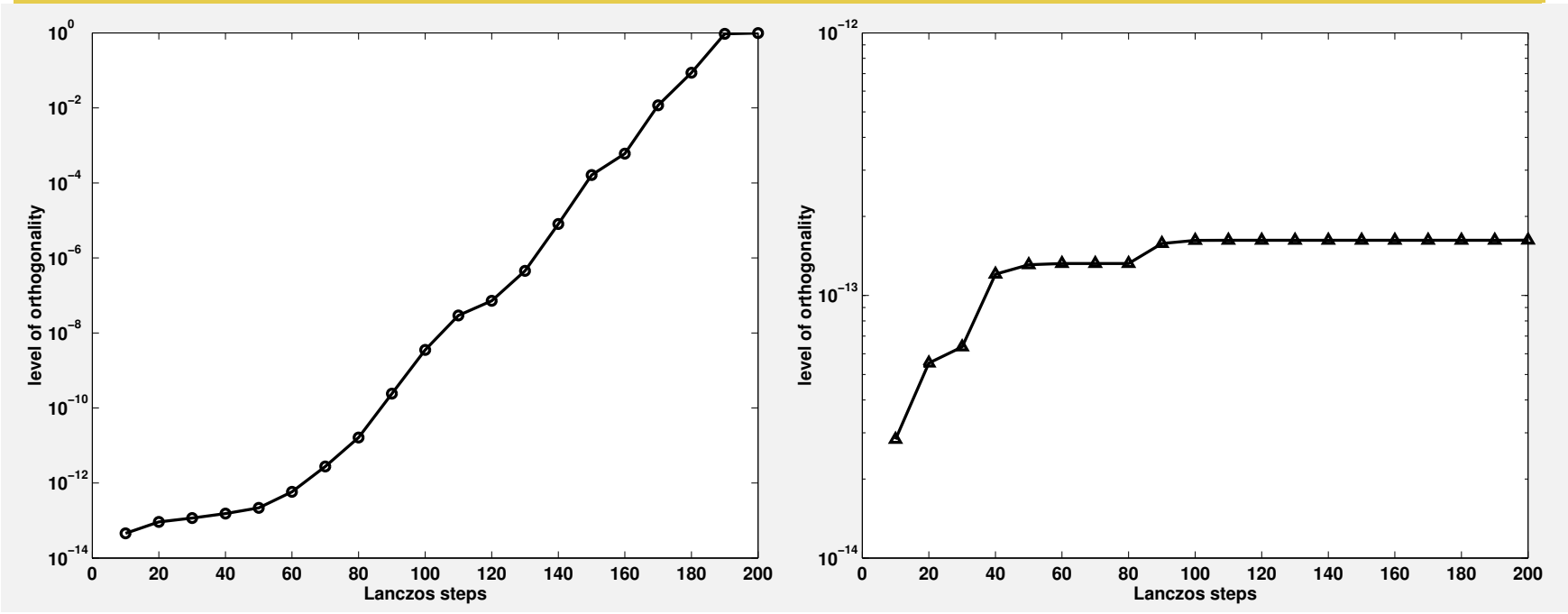
- Scalars β_{j+1}, α_j selected so that $v_{j+1} \perp v_j$, $v_{j+1} \perp v_{j-1}$, and $\|v_{j+1}\|_2 = 1$.

- In theory this is enough to guarantee that $\{v_j\}$ is orthonormal.

+ we have:

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

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

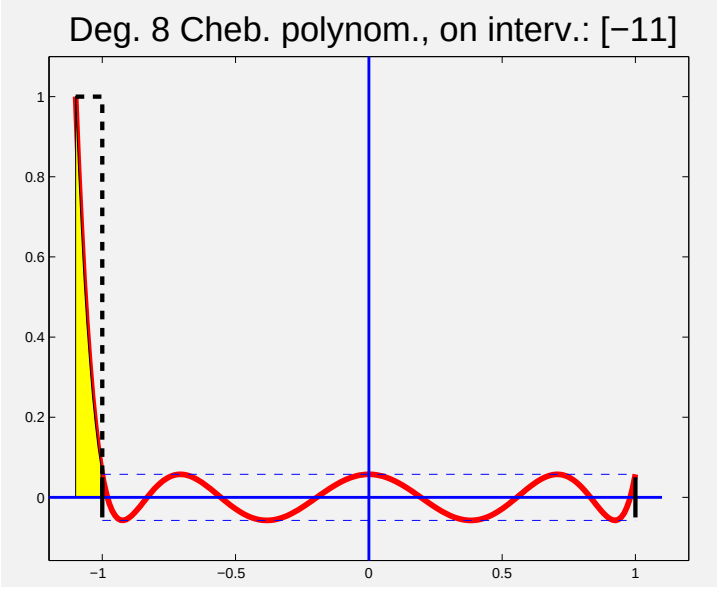
$$\hat{v}_i = P_k(A)v_i$$

➤ **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



Tests: – 3 examples only shown

➤ 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.

Larger tests

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

$Si_{9041}H_{1860}$

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_{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.

DATA MINING

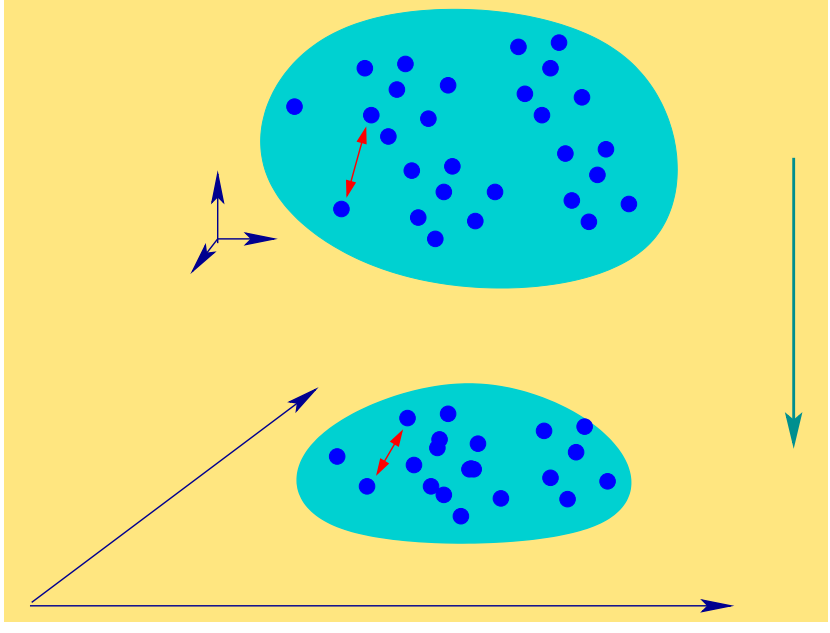
Dimension reduction

➤ **Dimensionality Reduction (DR) techniques pervasive to many applications**

➤ **Techniques depend on desirable features or application: Preserve angles? Preserve a graph? Maximize variance? ..**

➤ **Mapping can be explicit or implicit**

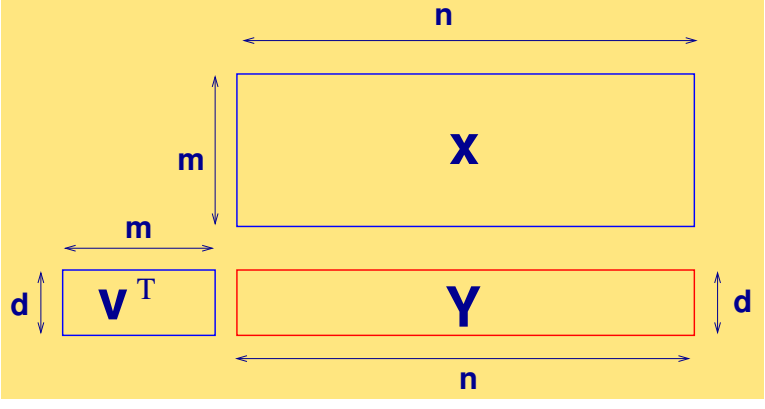
➤ **Important class of methods: linear projections.**



Linear Dimensionality Reduction

Given: a data set $X = [x_1, x_2, \ldots, x_n]$, and d the dimension of the desired reduced space Y.

Want: a linear transformation from X to Y



$$\boldsymbol{X} \in \mathbb{R}^{m \times n}$$

$$\boldsymbol{V} \in \mathbb{R}^{m \times d}$$

$$\boxed{\boldsymbol{Y} = \boldsymbol{V}^\top \boldsymbol{X}}$$

$$\rightarrow \quad \boldsymbol{Y} \in \mathbb{R}^{d \times n}$$

- m-dimens. objects (x_i) 'flattened' to d-dimens. space (y_i)

Constraint: The y_i's must satisfy certain properties

- Optimization problem

Linear Dimensionality Reduction: PCA

PCA: the projected data must have maximum variance

- Leads to maximizing

$$\text{Tr}\left[V^\top \bar{X}\bar{X}^\top V\right]$$

over all orthogonal $m \times d$ matrices V – where $\bar{X} = X(I - \frac{1}{n}ee^T)$ - origin-recentered version of X

- Solution $V = \{$ dominant eigenvectors $\}$ of the covariance matrix
- Also V == set of left singular vectors of $\bar{X}$

PCA and reduced rank matrix vector product

Often it is required to approximate the original data (matrix) by a low rank matrix before attempting to solve the original problem.

➤ **In Latent Semantic Indexing (LSI), the "query" is performed in the dominant singular space of A**

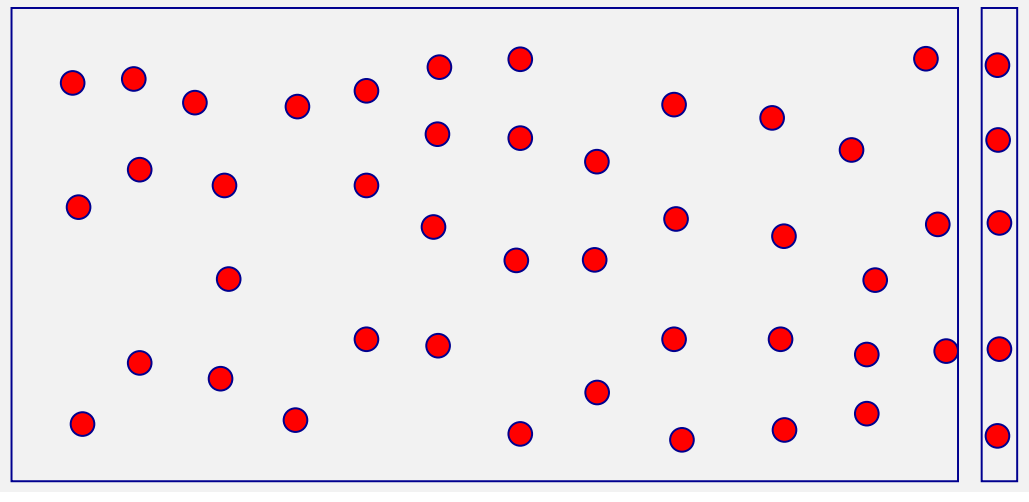
➤ **Methods utilizing Principal Component Analysis, e.g. Face Recognition.**

Technique: **Replace A (or $A^\dagger$) by a lower rank approximation A_k before solving original problem.**

➤ **This approximation captures main features of the data & gets rid of noise and redundancy**

Information Retrieval: Vector Space Model

➤ **Given a collection of documents (columns of a matrix A) and a query vector q.**



➤ **Collection represented by an $m \times n$ term by document matrix A**

➤ $\boxed{a_{ij} =}$ **some scaled frequency of term i in document j**

➤ **A query q is a set of terms (a 'pseudo-document') – represented similarly to a column**

Vector Space Model - continued

- Problem: find columns of A (documents) that best match q

Vector Space model: similarity metric = cosine of angle between a column and q

$$\frac{c^T q}{\|c\|_2 \|q\|_2}$$

- To rank all documents we need to compute similarity vector

$$s = A^T q$$

- Many problems with literal matching: *polysemy*, *synonymy*, ...
- Need to extract intrinsic information – or underlying "semantic" information –

Use of the SVD

- Solution (LSI): replace matrix A by a low rank approximation using the Singular Value Decomposition (SVD)

$$A = U\Sigma V^T \quad \rightarrow \quad A_k = U_k \Sigma_k V_k^T$$

- U_k : term space, V_k: document space.

New similarity vector:

$$s_k = A_k^T q = V_k \Sigma_k U_k^T q$$

Issues:

- How to select k?
- Problem with computational cost (memory + computation)
- Problem with updates

IR: Use of the Lanczos algorithm

➤ Lanczos is good at catching large (and small) eigenvalues: can compute singular vectors with Lanczos, & use them in LSI

➤ Can do better: Use the Lanczos vectors directly for the projection..

➤ First advocated by: K. Blom and A. Ruhe [SIMAX, vol. 26, 2005]. Use Lanczos bidiagonalization.

➤ Use a similar approach – But directly with AA^T or A^TA.

IR: Use of the Lanczos algorithm (1)

- Let $A \in \mathbb{R}^{m \times n}$. Apply the Lanczos procedure to $M = AA^T$.

Result:

$$Q_k^T A A^T Q_k = T_k$$

with Q_k orthogonal, T_k tridiagonal.

- Define $s_i \equiv$ orth. projection of Ab on subspace span$\{Q_i\}$

$$s_i := Q_i Q_i^T Ab.$$

- s_i can be easily updated from s_{i-1}:

$$s_i = s_{i-1} + q_i q_i^T Ab.$$

IR: Use of the Lanczos algorithm (2)

- If $n < m$ it may be more economial to apply Lanczos to $M = A^T A$ which is $n \times n$. Result:

$$\bar{Q}_k^T A^T A \bar{Q}_k = \bar{T}_k$$

- Define:

$$t_i := A \bar{Q}_i \bar{Q}_i^T b,$$

- Project b first before applying A to result.

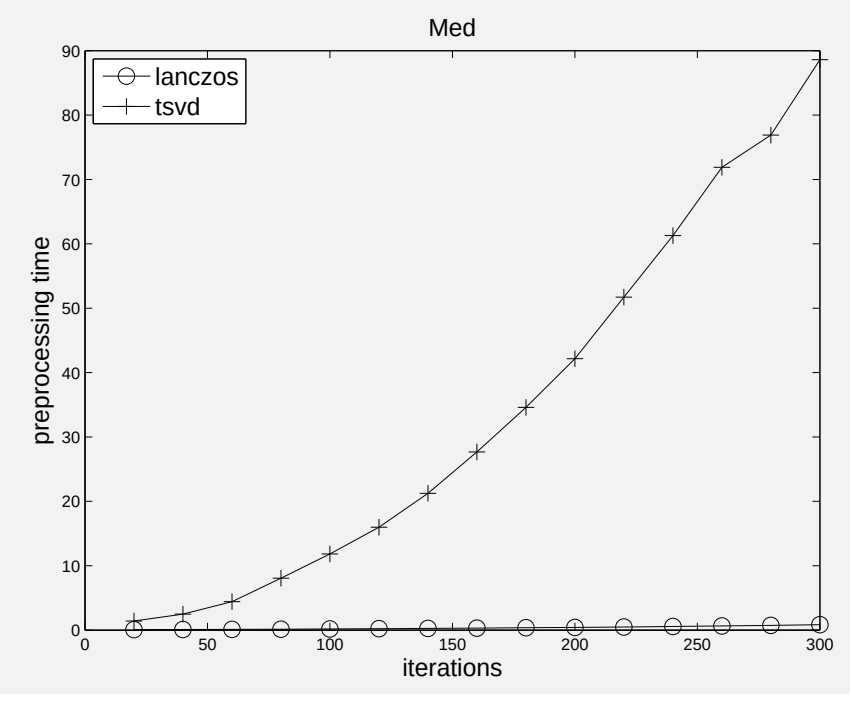
Tests: IR

Information retrieval datasets

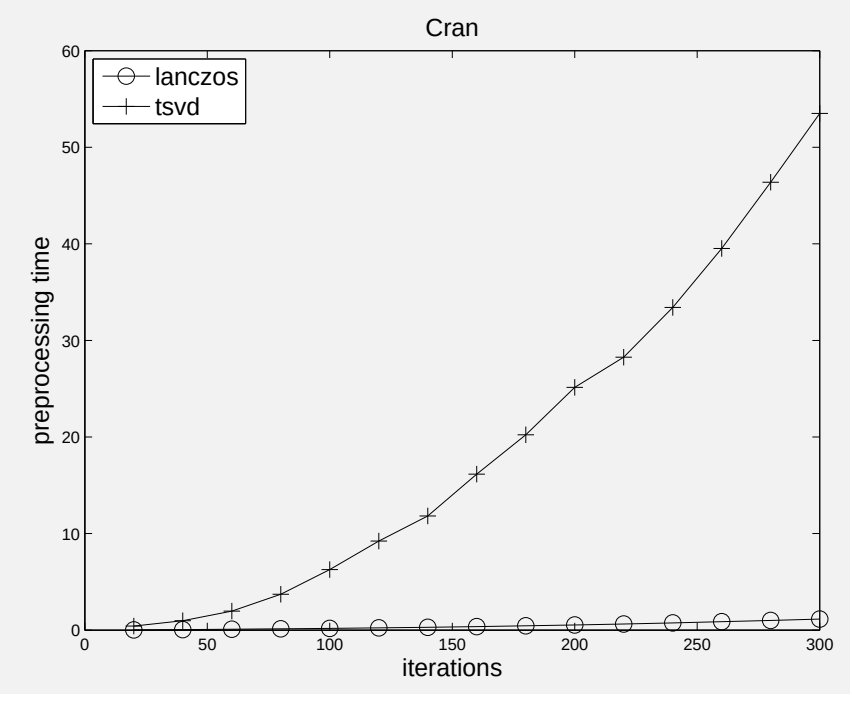
	# Terms	# Docs	# queries	sparsity
MED	7,014	1,033	30	0.735
CRAN	3,763	1,398	225	1.412

Preprocessing times

Med dataset.

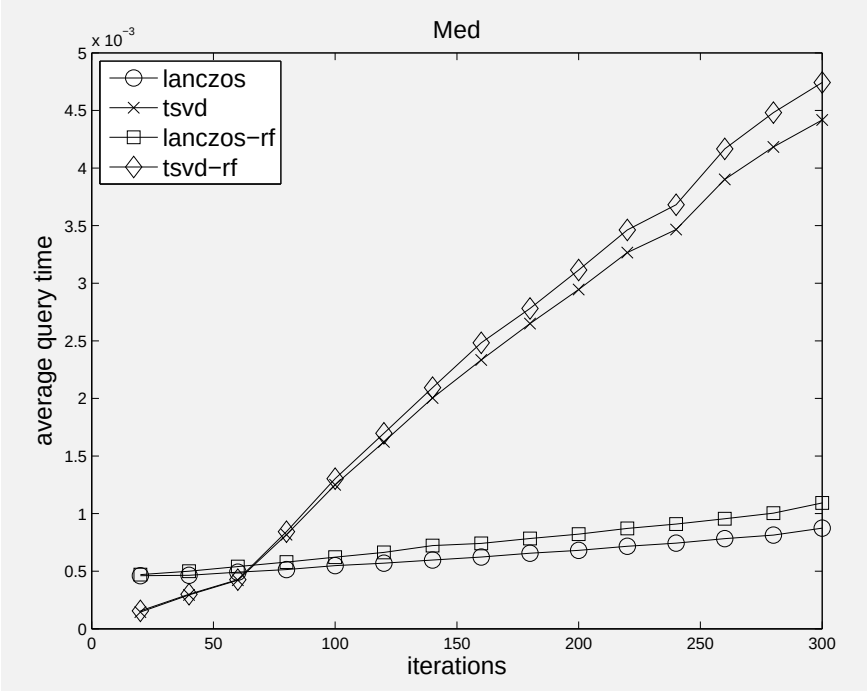


Cran dataset.

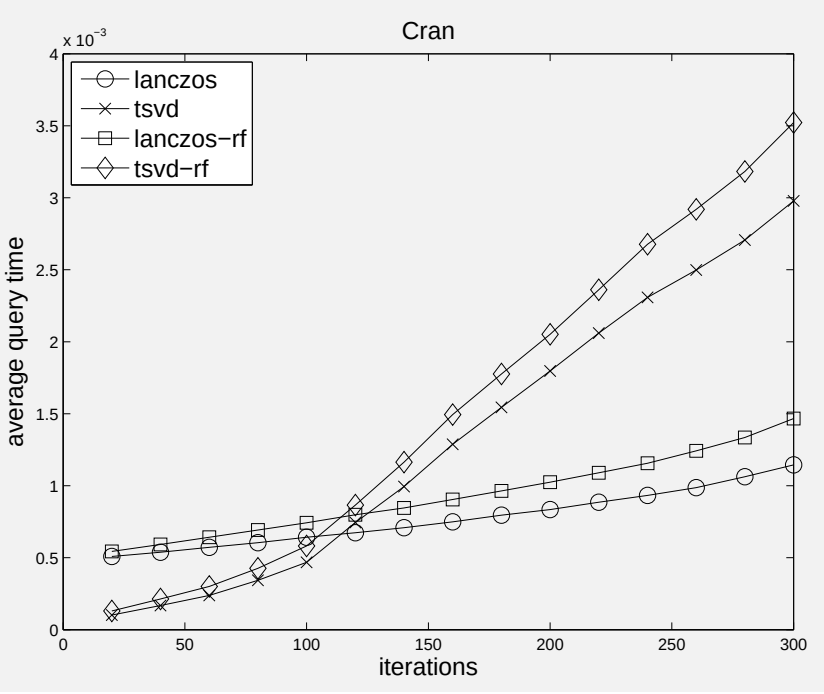


Average query times

Med dataset

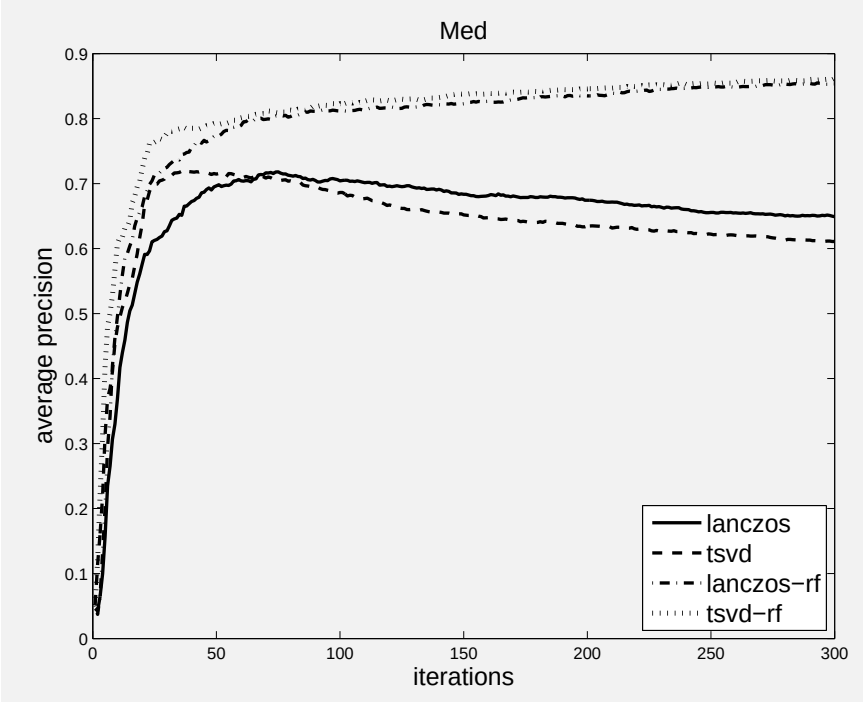


Cran dataset.

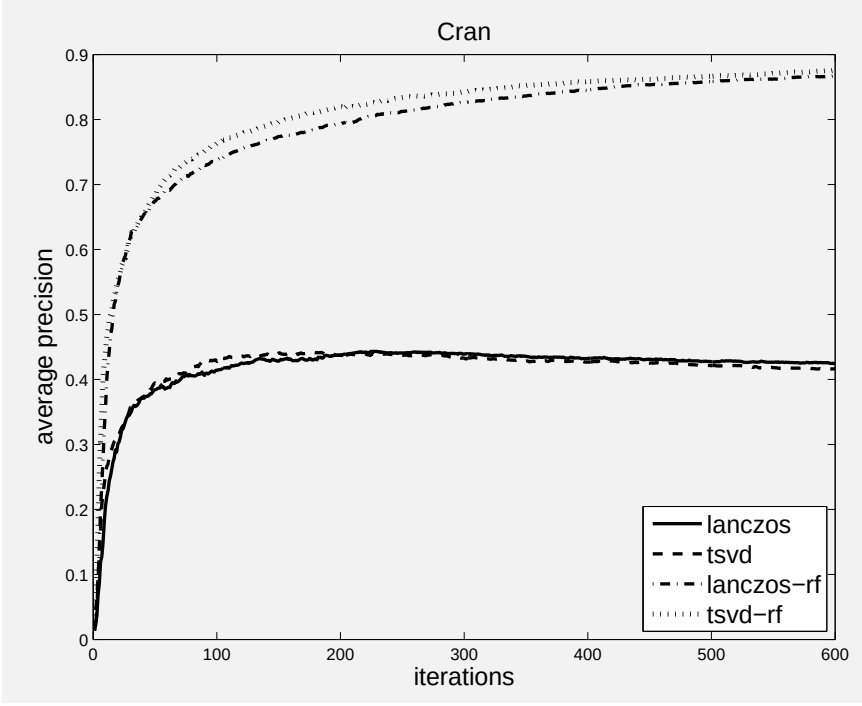


Average retrieval precision

Med dataset



Cran dataset



Retrieval precision comparisons

Conclusion

- Many interesting sparse matrix problems in various applications
- Still much to do in "solvers" [parallel implementations, robust techniques, ...]
- In data mining: quite a bit of data and software freely available..

My web-page:

`http://www.cs.umn.edu/~saad`