# Computing the diagonal of the inverse of a sparse matrix

*Yousef Saad*

**Department of Computer Science and Engineering**

*University of Minnesota*

*"Sparse Days", Toulouse, June 15, 2010*

## *Motivation: DMFT*

'Dynamic Mean Field Theory' - quantum mechanical studies of highly correlated particles

➤ Equation to be solved (repeatedly) is Dyson's equation

$$G(\omega) = [(\omega + \mu)I - V - \Sigma(\omega) + T]^{-1}$$

- $\omega$ (frequency) and $\mu$ (chemical potential) are real
- $V$ = trap potential = real diagonal
- $\Sigma(\omega)$ == local self-energy - a complex diagonal
- $T$ is the hopping matrix (sparse real).

- Interested only in diagonal of $G(\omega)$ – in addition, equation must be solved self-consistently and ...
- ... must do this for many $\omega$'s
- Related approach: Non Equilibrium Green's Function (NEGF) approach used to model nanoscale transistors.
- Many new applications of diagonal of inverse [and related problems.]
- A few examples to follow

## *Introduction: A few examples*

*Problem 1:* Compute Tr[inv[A]] the trace of the inverse.

- Arises in cross validation :

$$\frac{\|(I - A(\theta))g\|_2}{\text{Tr}\,(I - A(\theta))} \quad \text{with} \quad A(\theta) \equiv I - D(D^T D + \theta L L^T)^{-1} D^T,$$

$D$ == blurring operator and $L$ is the regularization operator

- In [Huntchinson '90] Tr[Inv[A]] is stochastically estimated
- Many authors addressed this problem.

*Problem 2:* Compute Tr [ f (A)], $f$ a certain function

Arises in many applications in Physics. Example:

- Stochastic estimations of Tr ( f(A)) extensively used by quantum chemists to estimate Density of States, see

[Ref: H. Röder, R. N. Silver, D. A. Drabold, J. J. Dong, Phys. Rev. B. 55, 15382 (1997)]

*Problem 3:* Compute diag[inv(A)] the diagonal of the inverse

- Arises in Dynamic Mean Field Theory [DMFT, motivation for this work].

In DMFT, we seek the diagonal of a "Green's function" which solves (self-consistently) Dyson's equation. [see J. Freericks 2005]

- Related approach: Non Equilibrium Green's Function (NEGF) approach used to model nanoscale transistors.

- In uncertainty quantification, the diagonal of the inverse of a covariance matrix is needed [Bekas, Curioni, Fedulova '09]

*Problem 4:* Compute diag[ f (A)] ; $f$ = a certain function.

- Arises in any density matrix approach in quantum modeling - for example Density Functional Theory.

- Here, $f$ = Fermi-Dirac operator:

$$f(\epsilon) = \frac{1}{1 + \exp(\frac{\epsilon - \mu}{k_B T})}$$

Note: when $T \to 0$ then $f$ becomes a step function.

Note: if $f$ is approximated by a rational function then diag[f(A)] $\approx$ a lin. combinaiton of terms like diag[$(A - \sigma_i I)^{-1}$]

- Linear-Scaling methods based on approximating $f(H)$ and $\mathrm{Diag}(f(H))$ – avoid 'diagonalization' of $H$

# *Methods based on the sparse L U factorization*

- Basic reference:

K. Takahashi, J. Fagan, and M.-S. Chin, *Formation of a sparse bus impedance matrix and its application to short circuit study*, in Proc. of the Eighth Inst. PICA Conf., Minneapolis, MN, IEEE, Power Engineering Soc., 1973, pp. 63-69.

- Described in [Duff, Erisman, Reid, p. 273] -
- Algorithm used by Erisman and Tinney [Num. Math. 1975]

- Main idea. If $A = LDU$ and $B = A^{-1}$ then

$$B = U^{-1}D^{-1} + B(I - L); \quad B = D^{-1}L^{-1} + (I - U)B.$$

- Not all entries are needed to compute selected entries of $B$
- For example: Consider lower part, $i > j$; use first equation:

$$b_{ij} = (B(I - L))_{ij} = -\sum_{k>j} b_{ik} l_{kj}$$

- Need entries $b_{ik}$ of row $i$ where $L_{kj} \neq 0$, $k > j$.
- "Entries of $B$ belonging to the pattern of $(L, U)^T$ can be extracted without computing any other entries outside the pattern."

- More recently exploited in a different form in

L. Lin, C. Yang, J. Meza, J. Lu, L. Ying, W. E *SelInv – An algorithm for selected inversion of a sparse symmetric matrix*, Tech. Report, Princeton Univ.

- An algorithm based on a form of nested dissection is described in Li, Ahmed, Glimeck, Darve [2008]

- A close relative to this technique is represented in

L. Lin , J. Lu, L. Ying , R. Car , W. E *Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems* Comm. Math. Sci, 2009.

- Difficulty: 3-D problems.

## *Stochastic Estimator*

**Notation:**

- $A$ = original matrix, $B = A^{-1}$.
- $\delta(B) = \text{diag}(B)$ [matlab notation]
- $\mathcal{D}(B)$ = diagonal matrix with diagonal $\delta(B)$
- $\odot$ and $\oslash$: Elementwise multiplication and division of vectors
- $\{v_j\}$: Sequence of $s$ random vectors

---

**Result:**

$$\delta(B) \approx \left[\sum_{j=1}^{s} v_j \odot B v_j\right] \oslash \left[\sum_{j=1}^{s} v_j \odot v_j\right]$$

---

Refs: C. Bekas , E. Kokiopoulou & YS ('05), Recent: C. Bekas, A. Curioni, I. Fedulova '09.

➤ Let $V_s = [v_1, v_2, \ldots, v_s]$. Then, alternative expression:

$$\mathcal{D}(B) \approx \mathcal{D}(BV_sV_s^\top)\mathcal{D}^{-1}(V_sV_s^\top)$$

**Question:** When is this result exact?

**Main Proposition**

- Let $V_s \in \mathbb{R}^{n\times s}$ with rows $\{v_{j,:}\}$; and $B \in \mathbb{C}^{n\times n}$ with elements $\{b_{jk}\}$
- Assume that: $\langle v_{j,:}, v_{k,:}\rangle = 0, \forall j \neq k$, s.t. $b_{jk} \neq 0$

Then:

$$\mathcal{D}(B) = \mathcal{D}(BV_sV_s^\top)\mathcal{D}^{-1}(V_sV_s^\top)$$

➤ Approximation to $b_{ij}$ exact when rows $i$ and $j$ of $V_s$ are $\perp$

## *Ideas from information theory: Hadamard matrices*

- Consider the matrix $V$ – want the rows to be as 'orthogonal as possible among each other', i.e., want to minimize

$$E_{rms} = \frac{\|I - VV^T\|_F}{\sqrt{n(n-1)}} \quad \text{or} \quad E_{max} = \max_{i \neq j} |VV^T|_{ij}$$

- Problems that arise in coding: find code book [rows of $V$ = code words] to minimize 'cross-correlation amplitude'

- Welch bounds:

$$E_{rms} \geq \sqrt{\frac{n-s}{(n-1)s}} \qquad E_{max} \geq \sqrt{\frac{n-s}{(n-1)s}}$$

- Result: $\exists$ a sequence of $s$ vectors $v_k$ with binary entries which achieve the first Welch bound iff $s = 2$ or $s = 4k$.

- Hadamard matrices are a special class: $n \times n$ matrices with entries $\pm 1$ and such that $HH^\top = nI$.

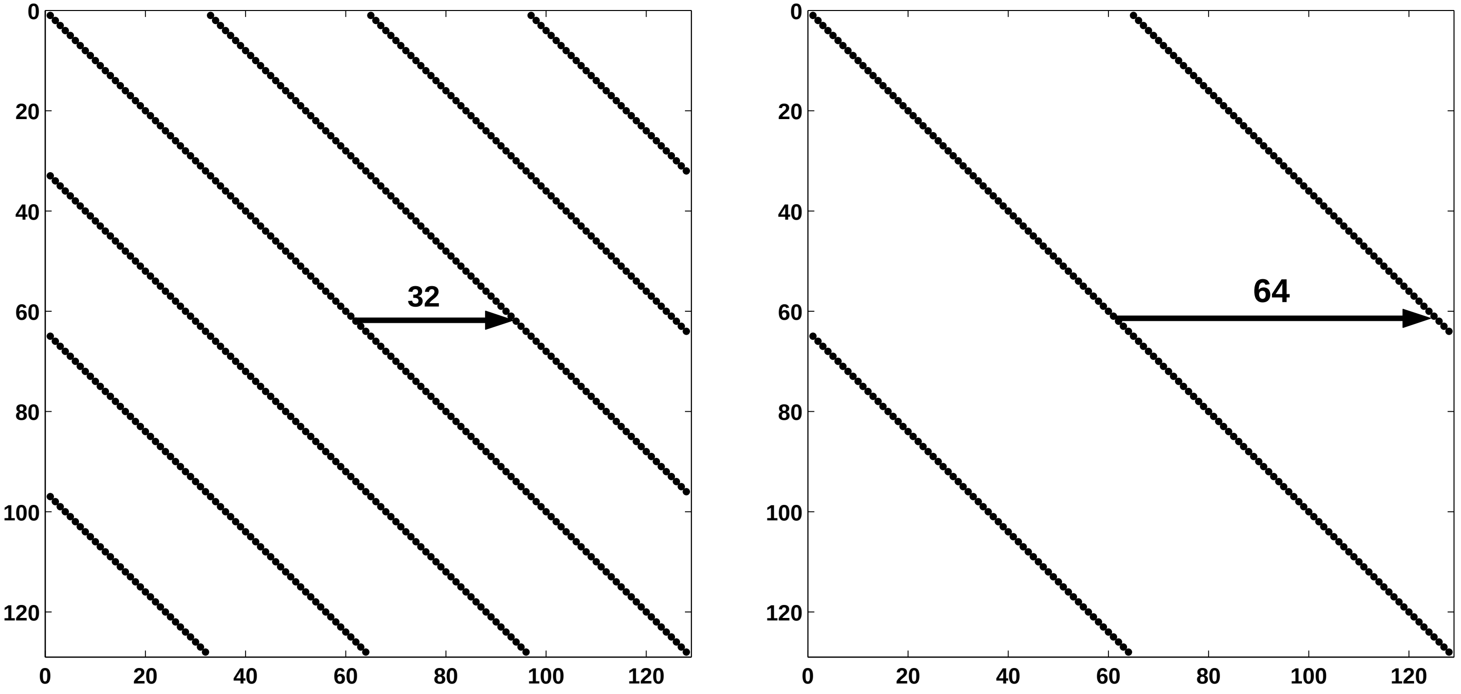
Examples :

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

- Achieve both Welch bounds
- Can build larger Hadamard matrices recursively:

  Given two Hadamard matrices $H_1$ and $H_2$, the Kronecker product $H_1 \otimes H_2$ is a Hadamard matrix.

- Too expensive to use the whole matrix of size $n$
- Can use $V_s$ = matrix of $s$ first columns of $H_n$




Pattern of $V_s V_s^\top$, for $s = 32$ and $s = 64$.

# *A Lanczos approach*

- Given a Hermitian matrix $A$ - generate Lanczos vectors via:

$$\beta_{i+1}q_{i+1} = Aq_i - \alpha_i q_i - \beta_i q_{i-1}$$

$\alpha_i, \beta_{i+1}$ selected s.t. $\|q_{i+1}\|_2 = 1$ and $q_{i+1} \perp q_i$, $q_{i+1} \perp q_{i-1}$

- Result:

$$AQ_m = Q_m T_m + \beta_{m+1} q_{m+1} e_m^\top,$$

- When $m = n$ then $A = Q_n T_n Q_n^\top$ and $A^{-1} = Q_n T_n^{-1} Q_n^\top$.
- For $m < n$ use the approximation: $A^{-1} \approx Q_m T_m^{-1} Q_m^\top \to$

$$\mathcal{D}(A^{-1}) \approx \mathcal{D}[Q_m T_m^{-1} Q_m^\top]$$

# ALGORITHM : 1 . *diagInv via Lanczos*

*For* $j = 1, 2, \cdots, Do$:

$\beta_{j+1} q_{j+1} = A q_j - \alpha_j q_j - \beta_j q_{j-1}$ *[Lanczos step]*

$p_j := q_j - \eta_j p_{j-1}$

$\delta_j := \alpha_j - \beta_j \eta_j$

$d_j := d_{j-1} + \frac{p_j \odot p_j}{\delta_j}$ *[Update of diag(inv(A))]*

$\eta_{j+1} := \frac{\beta_{j+1}}{\delta_j}$

*EndDo*

- $d_k$ (a vector) will converge to the diagonal of $A^{-1}$
- Limitation: Often requires all $n$ steps to converge
- One advantage: Lanczos is shift invariant – so can use this for many $\omega$'s
- Potential: Use as a direct method - exploiting sparsity

## *Using a sparse $V$: Probing*

**Goal:** Find $V_s$ such that (1) $s$ is small and (2) $V_s$ satisfies Proposition (rows $i$ & $j$ orthgonoal for any nonzero $b_{ij}$)

**Difficulty:** Can work only for sparse matrices but $B = A^{-1}$ is usually dense

- $B$ can sometimes be approximated by a sparse matrix.
- Consider for some $\epsilon$ :

$$(B_\epsilon)_{ij} = \begin{cases} b_{ij}, & |b_{ij}| > \epsilon \\ 0, & |b_{ij}| \le \epsilon \end{cases}$$

- $B_\epsilon$ will be sparse under certain conditions, e.g., when $A$ is diagonally dominant
- In what follows we assume $B_\epsilon$ is sparse and set $B := B_\epsilon$.
- Pattern will be required by standard probing methods.

# Generic Probing Algorithm

**ALGORITHM : 2. *Probing***

*Input:* $A$, $s$
*Output:* Matrix $\mathcal{D}(B)$
Determine $V_s := [v_1, v_2, \ldots, v_s]$
**for** $j \leftarrow 1$ **to** $s$
  Solve $Ax_j = v_j$
*end*
Construct $X_s := [x_1, x_2, \ldots, x_s]$
Compute $\mathcal{D}(B) := \mathcal{D}\left(X_s V_s^\top\right) \mathcal{D}^{-1}(V_s V_s^\top)$

➤ Note: rows of $V_s$ are typically scaled to have unit 2-norm =1., so $\mathcal{D}^{-1}(V_s V_s^\top) = I$.

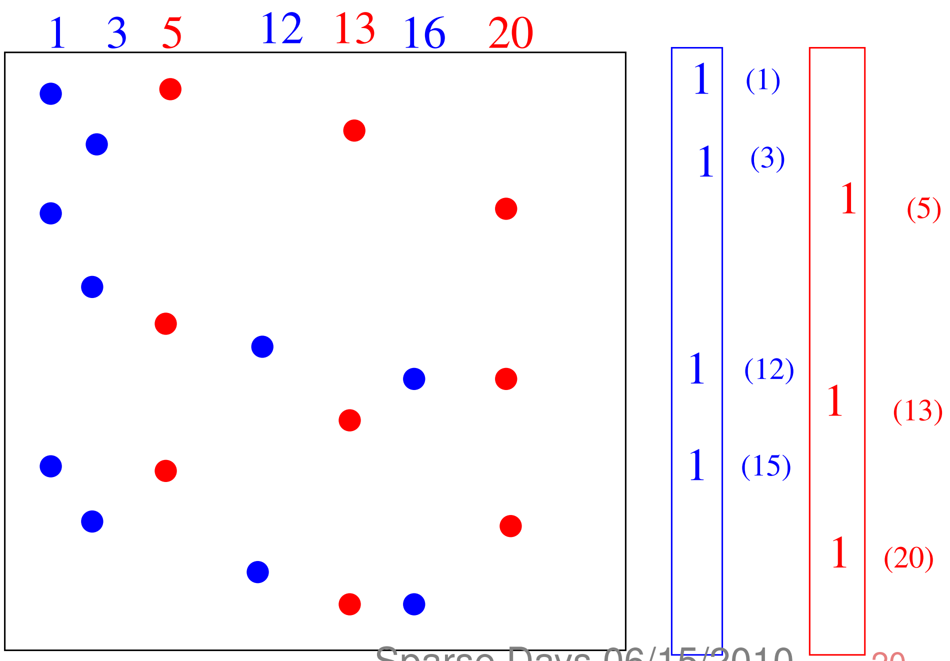
## Standard probing (e.g. to compute a Jacobian)

- Several names for same method: "probing"; "CPR", "Sparse Jacobian estimators",..

Basis of the method: can compute Jacobian if a coloring of the columns is known so that no two columns of the same color overlap.

All entries of same color can be computed with one matvec.

*Example:* For all blue entries multiply $B$ by the blue vector on right.



## *What about Diag(inv(A))?*

- Define $v_i$ - probing vector associated with color $i$:

$$[v_i]_k = \begin{cases} 1 & \text{if } color(k) == i \\ 0 & \text{otherwise} \end{cases}$$

- Standard probing satisfies requirement of Proposition but...
- ... this coloring is not what is needed! [It is an overkill]

**Alternative:**

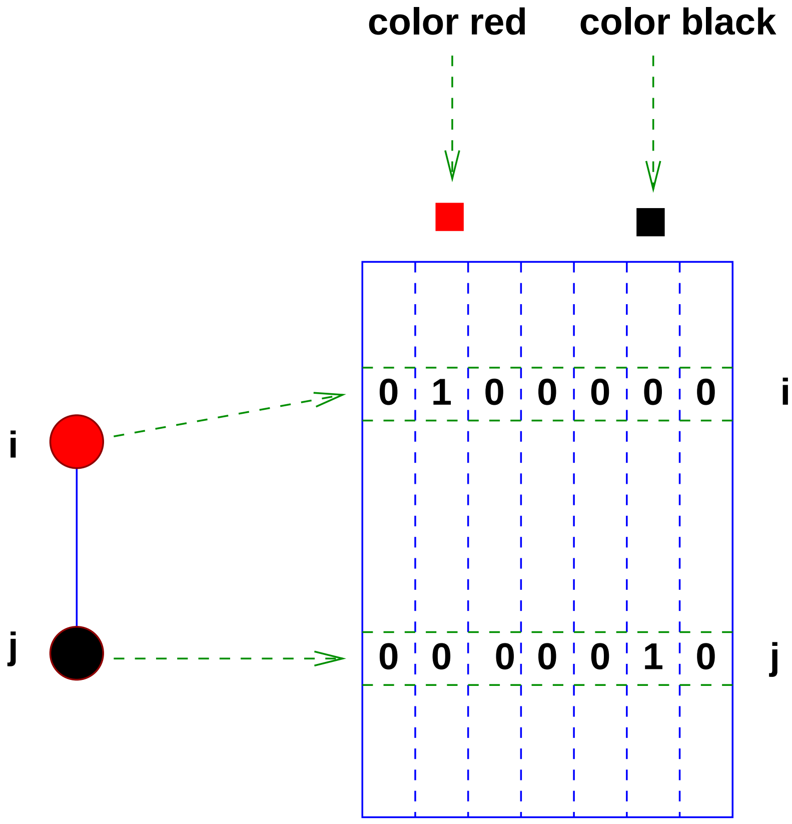
- Color the graph of $B$ in the standard graph coloring algorithm [Adjacency graph, not graph of column-overlaps]

**Result:** Graph coloring yields a valid set of probing vectors for $\mathcal{D}(B)$.

**Proof:**

➤ Column $v_c$: one for each node $i$ whose color is $c$, zero elsewhere.

➤ Row $i$ of $V_s$: has a '1' in column $c$, where $c = color(i)$, zero elsewhere.

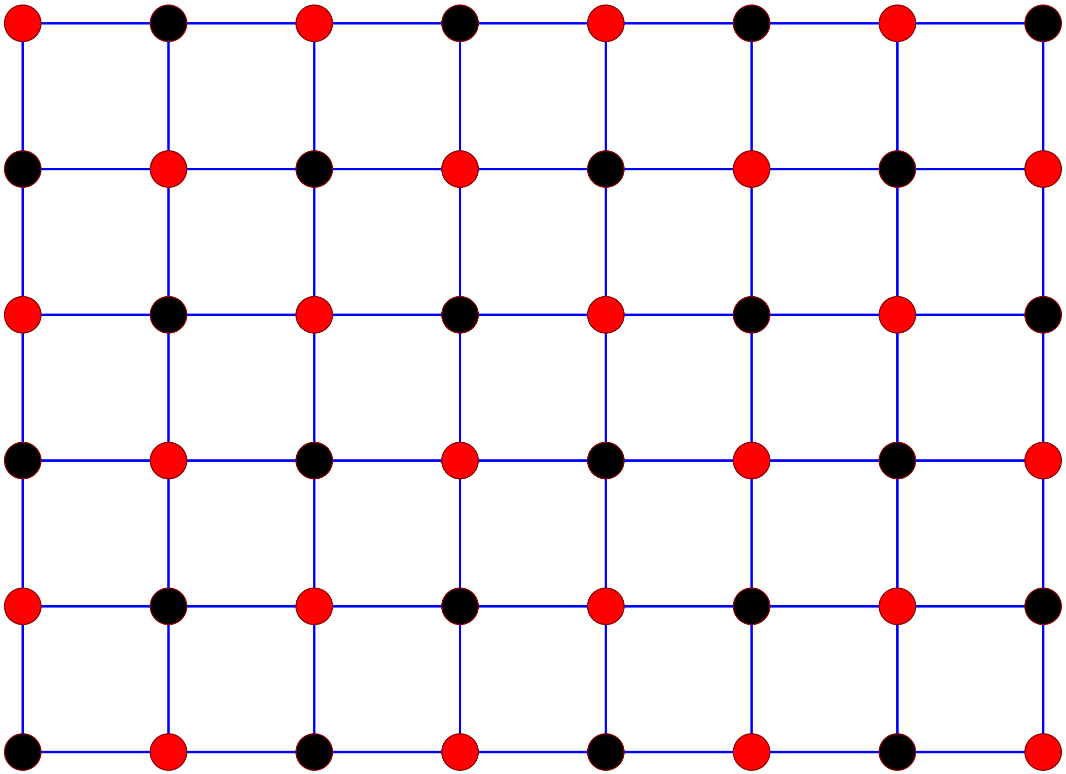



➤ If $b_{ij} \neq 0$ then in matrix $V_s$:

- $i$-th row has a '1' in column $color(i)$, '0' elsewhere.
- $j$-th row has a '1' in column $color(j)$, '0' elsewhere.

➤ The 2 rows are orthogonal.

# Example:



- Two colors required for this graph $\longrightarrow$ two probing vectors
- Standard method: 6 colors [graph of $B^TB$]

## *Next Issue: Guessing the pattern of B*

- Recall that we are dealing with $B := B_\epsilon$ ['pruned' $B$]
- Assume $A$ diagonally dominant
- Write $A = D - E$, with $D = \mathcal{D}(A)$. Then :

$$A = D(I - F) \quad \text{with} \quad F \equiv D^{-1}E \quad \rightarrow$$

$$A^{-1} \approx \underbrace{(I + F + F^2 + \cdots + F^k)D^{-1}}_{B^{(k)}}$$

- When $A$ is D.D. $\|F^k\|$ decreases rapidly.
- Can approximate pattern of $B$ by that of $B^{(k)}$ for some $k$.
- Interpretation in terms of paths of length $k$ in graph of $A$.

*Q: How to select $k$?*

*A: Inspect $A^{-1}e_j$* for some $j$

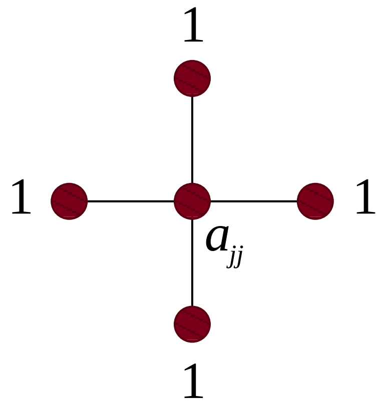
- Values of solution outside pattern of $(A^k e_j)$ should be small.
- If during calculations we get larger than expected errors – then redo with larger $k$, more colors, etc..
- Can we salvage what was done? Question still open.

# *Preliminary experiments*

## *Problem Setup*

- **DMFT**: Calculate the imaginary time Green's function
- **DMFT Parameters**: Set of physical parameters is provided
- **DMFT loop**: At most 10 outer iterations, each consisting of 62 inner iterations

- **Each inner iteration**: Find $\mathcal{D}(B)$
- **Each inner iteration**: Find $\mathcal{D}(B)$
- **Matrix**: Based on a five-point stencil with $a_{jj} = \mu + i\omega - V - s(j)$




## *Probing Setup*

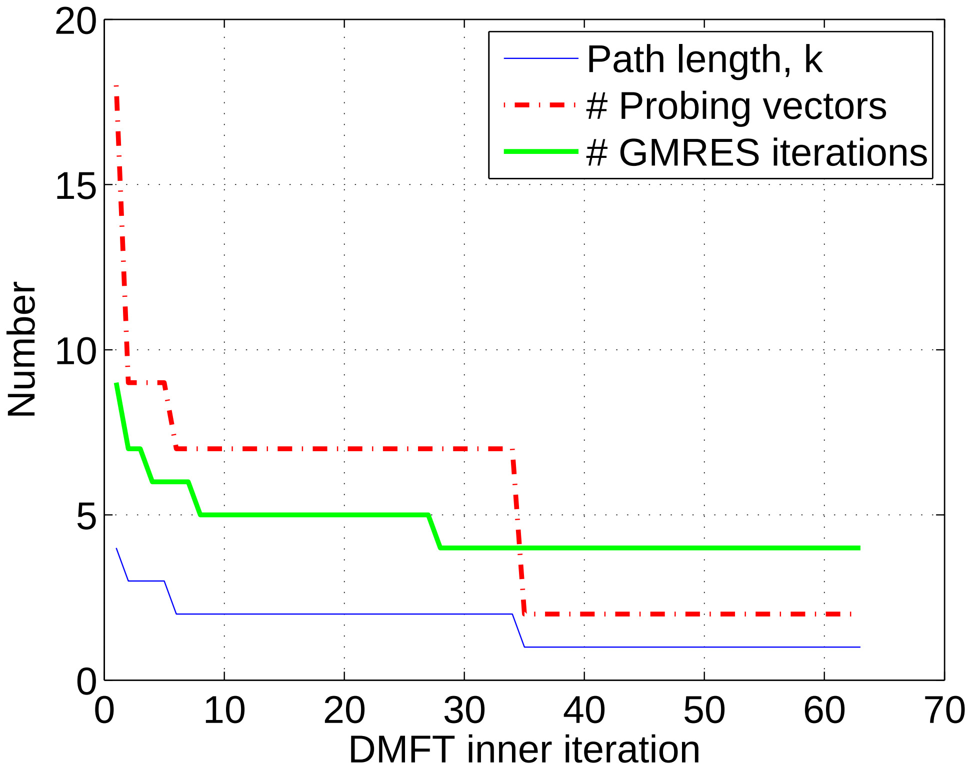
- **Probing tolerance**: $\epsilon = 10^{-10}$
- **GMRES tolerance**: $\delta = 10^{-12}$

## *Results*

CPU times (sec) for one inner iteration of DMFT.

| $n \longrightarrow$ | $21^2$ | $41^2$ | $61^2$ | $81^2$ |
|---|---|---|---|---|
| **LAPACK** | 0.5 | 26 | 282 | $> 1000$ |
| **Lanczos** | 0.2 | 9.9 | 115 | 838 |
| **Probing** | 0.02 | 0.19 | 0.79 | 2.0 |

A few statistics for case $n = 81$



## *Challenge: The indefinite case*

- The DMFT code deals with a separate case which uses a "real axis" sampling..
- Matrix $A$ is no longer diagonally dominant – Far from it.
- This is a much more challenging case.
- One option: solve $Ax_j = e_j$ FOR ALL $j$'s - with the ARMS solver using ddPQ ordering + exploit multiple right-hand sides
- More appealing: DD-type approaches

# *Divided & Conquer approach*

Let $A$ == a 5-point matrix (2-D problem) split roughly in two:

$$A = \left(\begin{array}{cccc|cccc} A_1 & -I & & & & & & \\ -I & A_2 & -I & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -I & A_k & -I & & & \\ \hline & & & -I & A_{k+1} & -I & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & -I & A_{n_y-1} & -I \\ & & & & & & -I & A_{n_y} \end{array}\right)$$

where $\{A_j\}$ = tridiag. Write:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & \\ & A_{22} \end{pmatrix} + \begin{pmatrix} & A_{12} \\ A_{21} & \end{pmatrix},$$

with $A_{11} \in \mathbb{C}^{m \times m}$ and $A_{22} \in \mathbb{C}^{(n-m)\times(n-m)}$,

- Observation:

$$A = \begin{pmatrix} A_{11} + E_1 E_1^T & \\ & A_{22} + E_2 E_2^T \end{pmatrix} - \begin{pmatrix} E_1 E_1^T & E_1 E_2^T \\ E_2 E_1^T & E_2 E_2^T \end{pmatrix}.$$

where $E_1$, $E_2$ are (relatively) small rank matrices:

$$E_1 := \begin{pmatrix} \\ I \end{pmatrix} \in \mathbb{C}^{m \times n_x}, \quad E_2 := \begin{pmatrix} I \\ \\ \end{pmatrix} \in \mathbb{C}^{(n-m) \times n_x},$$

Of the form

$$A = C - EE^T, \quad C := \begin{pmatrix} C_1 & \\ & C_2 \end{pmatrix} \quad E := \begin{pmatrix} E_1 \\ E_2 \end{pmatrix}$$

- Idea: Use Sherman-Morrisson formula.

$$A^{-1} = C^{-1} + UG^{-1}U^T, \quad \text{with:}$$

$$U = C^{-1}E \in \mathbb{C}^{n \times n_x} \quad G = I_{n_x} - E^T U \in \mathbb{C}^{n_x \times n_x},$$

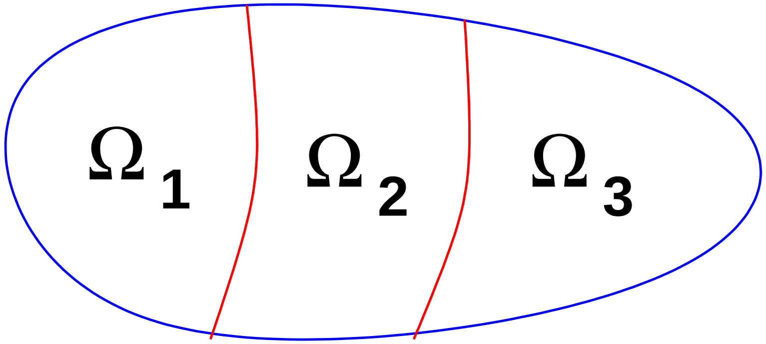
$\mathcal{D}(A^{-1})$ can be found from

$$\mathcal{D}(A^{-1}) = \underbrace{\begin{pmatrix} \mathcal{D}(C_1^{-1}) & \\ & \mathcal{D}(C_2^{-1}) \end{pmatrix}}_{recursion} + \mathcal{D}(UG^{-1}U^T).$$

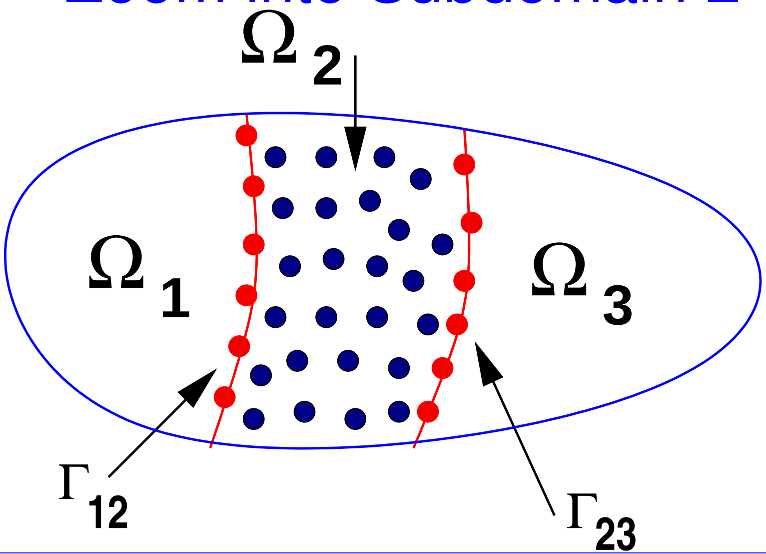
- $U$ : solve $CU = E$, or $\begin{cases} C_1 U_1 = E_1, \\ C_2 U_2 = E_2 \end{cases}$ Solve iteratively
- $G$: $G = I_{n_x} - E^T U = I_{n_x} - E_1^T U_1 - E_2^T U_2$

# *Domain Decomposition approach*

Domain decomposition with $p = 3$ subdomains



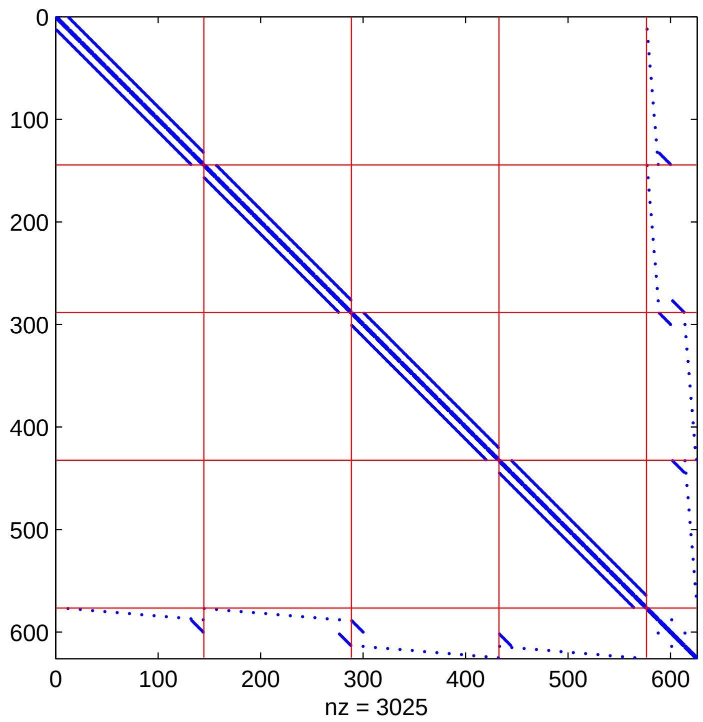

Zoom into Subdomain 2




Under usual ordering [interior points then interface points]:

$$A = \begin{pmatrix} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_p & F_p \\ F_1^T & F_2^T & \cdots & F_p^T & C \end{pmatrix} \equiv \begin{pmatrix} B & F \\ F^T & C \end{pmatrix},$$

Example of matrix $A$ based on a DDM ordering with $p = 4$ subdomains. ($n = 25^2$)




Inverse of $A$ [Assuming both $B$ and $S$ nonsingular]

$$A^{-1} = \begin{pmatrix} B^{-1} + B^{-1} F S^{-1} F^T B^{-1} & -B^{-1} F S^{-1} \\ -S^{-1} F^T B^{-1} & S^{-1} \end{pmatrix}$$

$$S = C - F^T B^{-1} F,$$

$$\mathcal{D}(A^{-1}) = \begin{pmatrix} \mathcal{D}(B^{-1}) + \mathcal{D}(B^{-1}FS^{-1}F^TB^{-1}) & \\ & \mathcal{D}(S^{-1}) \end{pmatrix}$$

- Note: each diagonal block decouples from others:

Inverse of $A$ in $i$-th block (domain)

$$(A^{-1})_{ii} = \mathcal{D}(B_i^{-1}) + \mathcal{D}(H_i S^{-1} H_i^T)$$
$$H_i = B_i^{-1} F_i$$

- Note: only nonzero columns of $F_i$ are those related to interface vertices.

- Approach similar to Divide and Conquer but not recursive..

# DMFT experiment

Times (in seconds) for direct inversion (INV), divide-and-conquer (D&C), and domain decomposition (DD) methods.

- $p = 4$ subd. for DD
- Various sizes - 2-D problems
- Times: seconds in matlab

| $\sqrt{n}$ | INV | D&C | DD |
|---|---|---|---|
| 21 | .3 | .1 | .1 |
| 51 | 12 | 1.4 | .7 |
| 81 | 88 | 7.1 | 3.2 |

- NOTE: work still in progress

# *Conclusion*

- Diag(inv(A)) problem: easy for Diag. Dominant case. Very challenging in (highly) indefinite case.
- Dom. Dec. methods can be a bridge between the two cases
- Approach [specifically for DMFT problem] :
  - Use direct methods in strongly Diag. Dom. case
  - Use DD-type methods in nearly Diag. Dom. case
  - Use direct methods in all other cases [until we find better means :-) ]