Sparse Days 2018, 27 Sep.

Dissection solver for higher precision arithmetic by inner iterative refinement

Atsushi Suzuki[1]

[1]Cybermedia Center, Osaka University
atsushi.suzuki@cas.cmc.osaka-u.ac.jp

joint work with
François-Xavier Roux, ONERA, LJLL Sorbonne Université

Outline

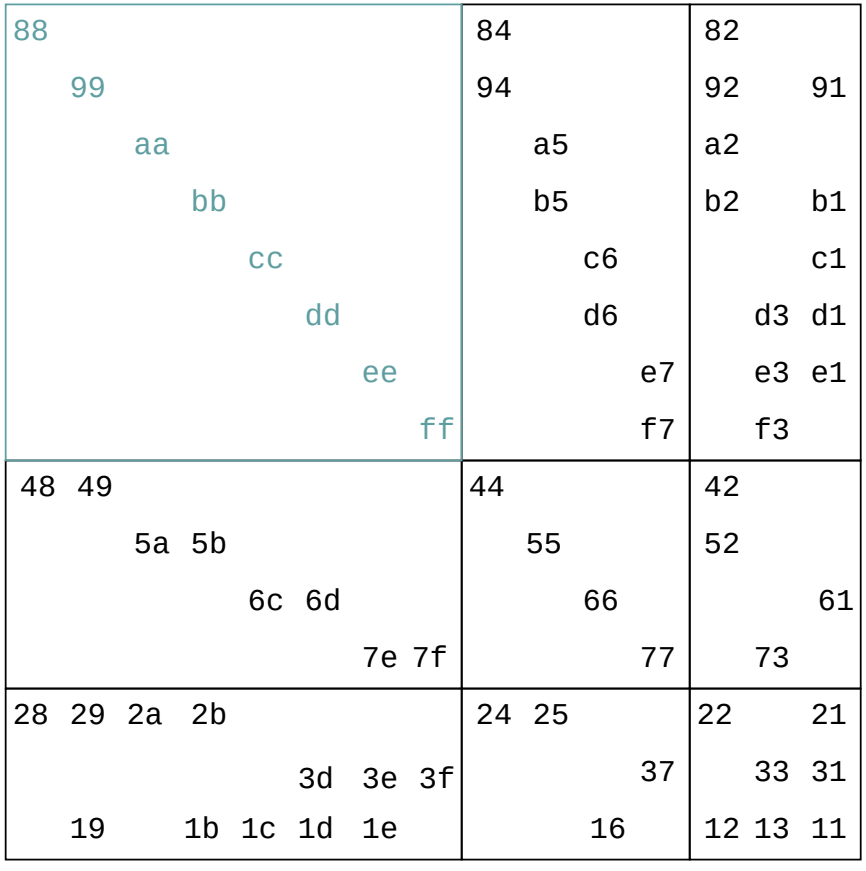
- overview of nested-dissection algorithm for sparse matrix
- LDU factorization with symmetric pivoting
- computation of a solution of singular linear system
- extension with 2x2 pivoting and unsymmetric pivoting
- kernel detection algorithm
- higher precision arithmetic with inner iterative refinement
- numerical results on matrices from a semi-conductor problem

recursive generation of Schur complement by nested-dissection

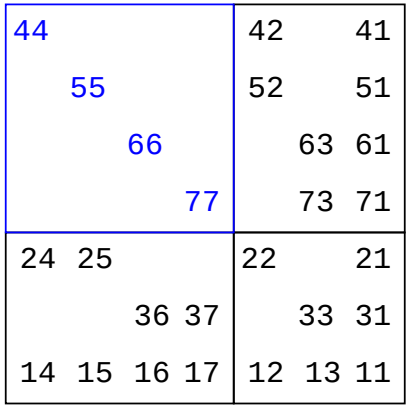
$$\begin{bmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S_{22} \end{bmatrix} \begin{bmatrix} I_1 & A_{11}^{-1}A_{12} \\ 0 & I_2 \end{bmatrix}$$

$S_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12} = A_{22} - (A_{21}U_{11}^{-1})D_{11}^{-1}L_{11}^{-1}A_{12}$: recursively computed

8 9 a b c d e f 4 5 6 7 2 3 1



Schur complement by sparse solver

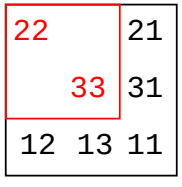


Schur complement by dense solver

Schur complement by dense solver

11

dense factorization



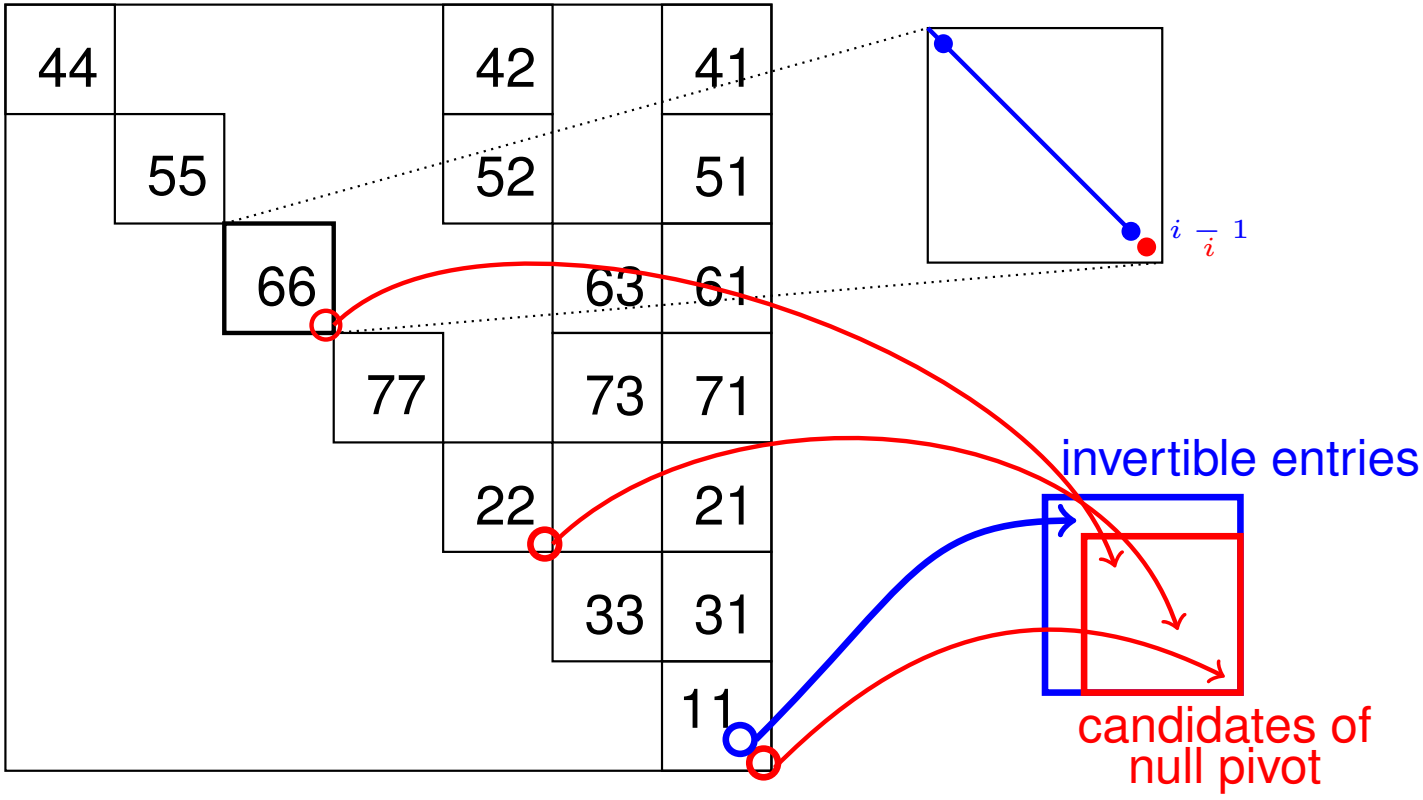
sparse part : completely in parallel

dense part : better use of BLAS 3; dgemm, dtrsm

Symmetric pivoting with postponing for block strategy : 1/2

- nested-dissection decomposition may produce singular sub-matrix for indefinite matrix

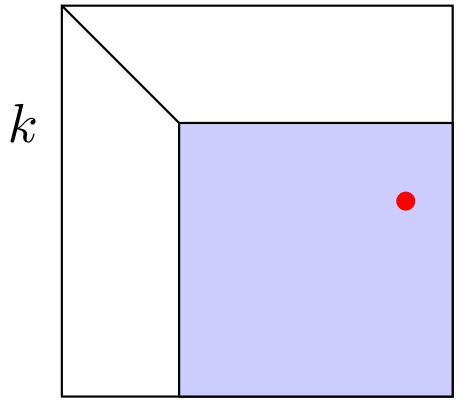
τ : given threshold for postponing, 10^{-2}
$|A(i,i)|/|A(i-1,i-1)| < \tau \Rightarrow \{A(k,j)\}_{i \leq k,j}$ are postponed



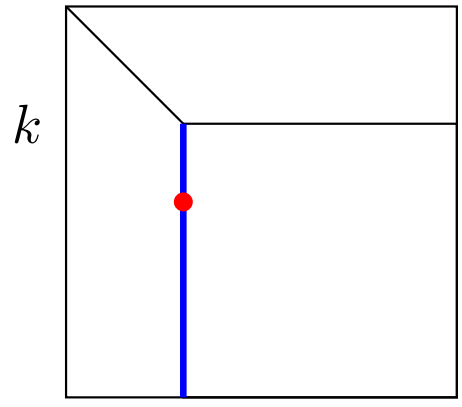
Schur complement matrix from postponed pivots is computed

Pivoting strategy

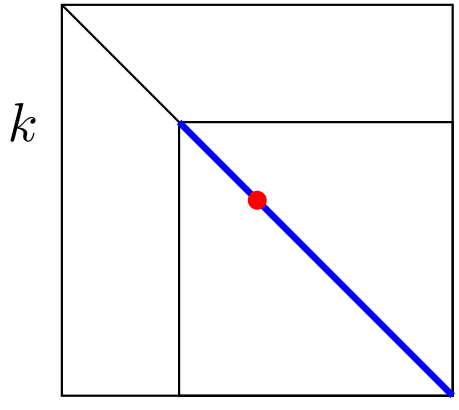
full pivoting : $A = \Pi_L^T L U \Pi_R$
find max$_{k<i,j\leq n}|A(i,j)|$



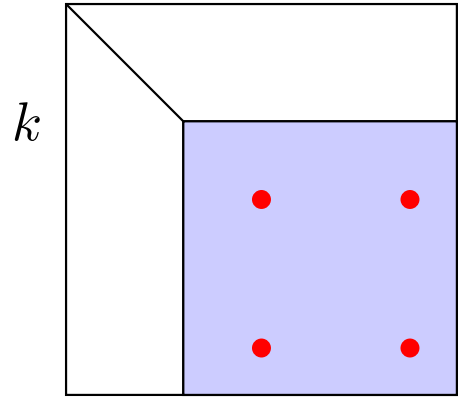
partial pivoting : $A = \Pi L U$
find max$_{k<i\leq n}|A(i,k)|$



symmetric pivoting : $A = \Pi^T L D U \Pi$
find max$_{k<i\leq n}|A(k,k)|$



2×2 pivoting : $A = \Pi^T L \tilde{D} U \Pi$
find max$_{k<i,j\leq n}$det$\begin{vmatrix} A(i,i) & A(i,j) \\ A(j,i) & A(j,j) \end{vmatrix}$



sym. pivoting is mathematically not always possible

Computation of a solution of singular linear system

- $A \in \mathbb{R}^{N\times N}$, $\quad k = \dim\mathrm{Ker}A$.
- index ordering $\{i_1, i_2, \cdots, i_N\}$ $\quad V_{N-k} = \mathrm{span}[\vec{e}_{i_1}, \vec{e}_{i_2}, \ldots, \vec{e}_{i_{N-k}}]$

assumption 1 : $V_{N-k} \cap \ker A = \{\vec{0}\} \quad \Rightarrow \quad \exists A_{11}^{-1}$
Solution in the space V_{N-k}

$$\vec{x} = A^{\dagger}\vec{f} \quad \Leftrightarrow \quad \vec{x} \in V_{N-k} \quad (A\vec{x} - \vec{f}, \vec{v}) = 0 \quad \forall \vec{v} \in V_{N-k}$$

$N_1 = \begin{bmatrix} A_{11}^{-1}A_{12} \\ -I_2 \end{bmatrix}$ $N_2 = \begin{bmatrix} A_{11}^{-T}A_{21}^{T} \\ -I_2 \end{bmatrix}$, $\ker A = \mathrm{span} N_1$, $\ker A^T = \mathrm{sapn} N_2$

assumption 2 :
$\vec{u} = A^{\dagger}\vec{f}$ is computable by LDU-factorization with symmetric pivot.

$\vec{f} \in \mathrm{Im}A$
$\Leftrightarrow \vec{f} \perp \mathrm{Ker}A^T \Leftrightarrow N_2^T\vec{f} = \vec{0} \Leftrightarrow A_{21}A_{11}^{-1}\vec{f}_1 - \vec{f}_2 = \vec{0} \Rightarrow \; AA^{\dagger}\vec{f} = \vec{f}$

$\forall \vec{\zeta} \in \mathbb{R}^k, \quad A(\vec{u} + N_1\vec{\zeta}) = \vec{f}$

- $\vec{u} + N_1\vec{\zeta} \in (\mathrm{Ker}A)^{\perp} \quad \Leftrightarrow \quad N_1^T(\vec{u} + N_1\vec{\zeta}) = \vec{0}$: algebraic inverse
- $\vec{u} + N_1\vec{\zeta} \in V_{N-k} \quad \Leftrightarrow \quad \vec{\zeta} = \vec{0}$: computational
- $\vec{u} + N_1\vec{\zeta} \in \mathrm{Im}A \quad \Leftrightarrow \quad N_2^T(\vec{u} + N_1\vec{\zeta}) = \vec{0} \Leftarrow \mathrm{Im}A \cap \mathrm{Ker}A = \{\vec{0}\}$

$\mathrm{Im}A \cap \mathrm{Ker}A = \{\vec{0}\} \Rightarrow N_2^T N_1$ is invertible
$N_1\vec{\eta} \in \mathrm{Im}A \cap \mathrm{Ker}A \Leftrightarrow N_2^T N_1\vec{\eta} = \vec{0}$, $\mathrm{Im}A \cap \mathrm{Ker}A = \{\vec{0}\} \Leftrightarrow (\, N_2^T N_1\vec{\eta} = \vec{0} \Rightarrow \vec{\eta} = \vec{0}\,)$

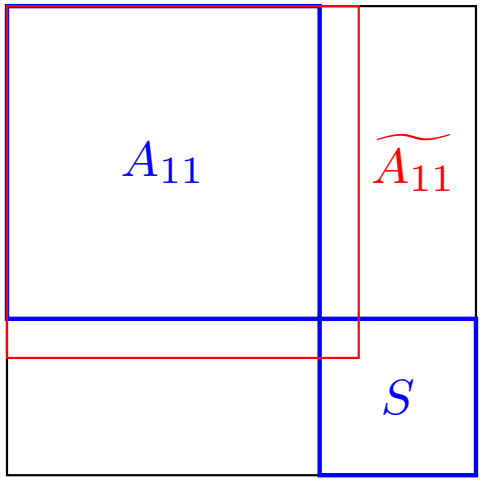
extension with 2x2 pivoting or unsymmetric pivoting

For matrix from PDE, most of part can be factorized with symmetric pivoting.

$\widetilde{A_{11}}$ has LDU-factorization with symmetric pivot

entries of S $\Leftarrow$ postponed pivots +

some invertible entries of $\widetilde{A_{11}}$



$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} I_{11} & A_{11}^{-1}A_{12} \\ 0 & I_2 \end{bmatrix}$$

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$$

- applying full pivot when S is (highly) unsymmetric, $\bar{S} = \Pi_L S \Pi_R$
- applying 2x2 pivot when S is symmetric + indefinite

with full pivoting Π_L / Π_R

$$\bar{S} = \Pi_L S \Pi_R = \begin{bmatrix} \bar{S}_{22} & \bar{S}_{23} \\ \bar{S}_{32} & \bar{S}_{33} \end{bmatrix} = \begin{bmatrix} \bar{S}_{22} & 0 \\ \bar{S}_{32} & E_{33} \end{bmatrix} \begin{bmatrix} I_2 & \bar{S}_{22}^{-1}\bar{S}_{23} \\ 0 & I_3 \end{bmatrix}$$

$$E_{33} = 0 \;\Rightarrow\; \mathrm{Ker}\bar{S} = \mathrm{span} \begin{bmatrix} \bar{S}_{22}^{-1}\bar{S}_{23} \\ -I_3 \end{bmatrix}, \quad \mathrm{Ker}S = \mathrm{span}\Pi_R \begin{bmatrix} \bar{S}_{22}^{-1}\bar{S}_{23} \\ -I_3 \end{bmatrix}$$

$$\mathrm{Ker}A = \begin{bmatrix} A_{11}^{-1}A_{12}^{(2)} & A_{11}^{-1}A_{12}^{(3)} \\ -I_2^{(2)} & 0 \\ 0 & -I_2^{(3)} \end{bmatrix} \Pi_R \begin{bmatrix} \bar{S}_{22}^{-1}\bar{S}_{23} \\ -I_3 \end{bmatrix}$$

novel kernel detection algorithm based on LDU

$A : N \times N$ unsymmetric, dimKer$A = k \geq 1$, dimIm$A \geq m$.
two parameters: l, n, which define size of factorization,

$$\begin{matrix} N-n \\ \updownarrow \\ n \\ \updownarrow \end{matrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S_{22} \end{bmatrix} \begin{bmatrix} I_1 & A_{11}^{-1}A_{12} \\ 0 & I_2 \end{bmatrix} \quad \widetilde{\mathrm{Im}_n} = \mathrm{span} \begin{bmatrix} \bar{A}_{11}^{-1}A_{12} \\ -I_2 \end{bmatrix}^{\perp}.$$

- projection : $P_n^{\perp} : \mathbb{R}^N \to \widetilde{\mathrm{Im}_n}$
- solution in subspace, $\bar{A}_{N-l}^{\dagger}\vec{b} = \begin{bmatrix} \bar{A}_{11}^{-1}\vec{b}_1 \\ \vec{0} \end{bmatrix}, b = \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \end{bmatrix} \quad \begin{matrix} \updownarrow N-l \\ \updownarrow \\ l \end{matrix}$

$\bar{A}_{11}^{-1}$: computed in quadruple-precision with perturbation to simulate double-precision round-off error.

kernel detection algorithm

DOI: 10.1002/nme.4729

n : candidate of dimension of the kernel
compute for $l = n-1, n, n+1$

$$\mathrm{err}_l^{(n)} := \max \left\{ \max_{\vec{x}=[\vec{0}\ \vec{x}_l]\neq\vec{0}} \frac{||P_n^{\perp}(\bar{A}_{N-l}^{\dagger}A\,\vec{x}-\vec{x})||}{||\vec{x}||}, \max_{\vec{x}=[\vec{x}_{N-l}\ \vec{0}]\neq\vec{0}} \frac{||\bar{A}_{N-l}^{\dagger}A\,\vec{x}-\vec{x}||}{||\vec{x}||} \right\}$$

$$\begin{array}{rcccccc} n=k+1 & \Leftrightarrow & \mathrm{err}_{n-1} \approx 0 & \wedge & \mathrm{err}_n \approx 0 & \wedge & \mathrm{err}_{n+1} \sim 1 \\ n=k & \Leftrightarrow & \mathrm{err}_{n-1} \gg 0 & \wedge & \mathrm{err}_n \approx 0 & \wedge & \mathrm{err}_{n+1} \sim 1 \\ n=k-1 & \Leftrightarrow & \mathrm{err}_{n-1} \gg 0 & \wedge & \mathrm{err}_n \gg 0 & \wedge & \mathrm{err}_{n+1} \sim 1 \end{array}$$

Theoretically $\neg A_{N-k+1}^{-1}$, but $\mathrm{err}_{k-1}^{(k)}$ is computable; $\sim ||\bar{A}_N^{-1}A_N - I_N||$.

higher precision arithmetic with inner iterative refinement

for high condition number matrix with coefficients given by "double"
A_{11} is moderate part $\Leftarrow$ postponing strategy with given threshold

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} I_{11} & A_{11}^{-1} A_{12} \\ 0 & I_2 \end{bmatrix}$$

$$S = A_{22} - A_{21} A_{11}^{-1} A_{12}$$

computation of S in higher precision is necessary

- iterative refinement to compute $X_{12} = A_{11}^{-1} A_{12}$
- quadruple precision arithmetic to compute $S = A_{22} - A_{21} X_{12}$.
- quadruple precision LDU-factorization for S with kernel detection

"quadruple" arithmetic is realized by "double-double" in QD library
iterative refinement
$A_{11} X_{12} = A_{12}$ in "double"
$R_{12}^{(1)} = A_{12} - A_{11} X_{12}$ in "quadruple"
loop $n = 1, 2, \cdots$
$A_{11} Y_{12}^{(n)} = R_{12}^{(n)}$ in "double"
$X_{12} = X_{12} + Y_{12}^{(n)}$ in "quadruple"
$R_{12}^{(n+1)} = A_{12} - A_{11} X_{12}$ in "quadruple"
SpMV by double-double is 10 times arithmetic intensive than double

Drift-Diffusion system at stationary state

- φ : electrostatic potential
- n : electron concentration
- p : hole concentration

$$\begin{aligned} \text{div}(\varepsilon E) &= q(p - n + C(x)) & E &= -\nabla\varphi \\ -\text{div} J_n &= 0 & J_n &= -q(\mu_n n \nabla\varphi - D_n \nabla n) \\ \text{div} J_p &= 0 & J_p &= -q(\mu_p p \nabla\varphi + D_p \nabla p) \end{aligned}$$

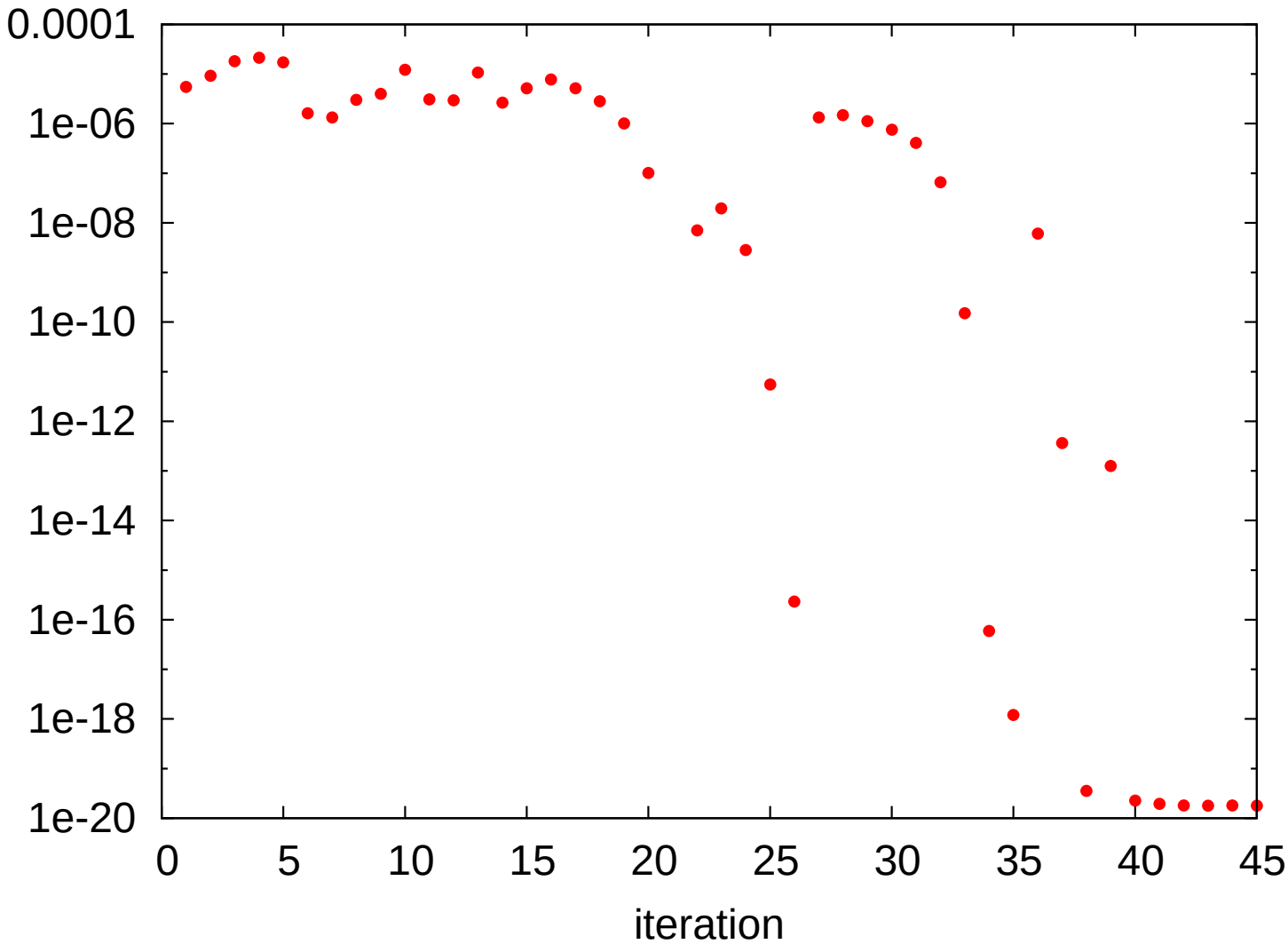
- q : positive electron charge
- ε : dielectric constant of the materials
- $D_n = \mu_n\theta$, $D_p = \mu_p\theta$: Einstein's relation

Maxwell-Boltzmann statistics : $p = N_i \exp(\frac{\varphi_p - \varphi}{\theta})$ leads to $J_p = -q\mu_p p \nabla\varphi_p$

- φ_p : quasi-Fermi level
- N_i : intrinsic concentration of the semiconductor
- $\theta = kT/q$: thermal voltage
- k : Boltzmann constant, T : lattice temperature

Finite Volume with Scharfetter-Gummel scheme + Newton iteration

Newton iteration of a semi-conductor problem



from 1st to 5th iterations and 10th, matrices are invertible, others are recognized as singular

matrices are provided by Toshiba Memory Corp.

results of kernel detection

$\widetilde{A} = W\,A\,W$: scaled with $[W]_{i\,i} = 1\sqrt{[A]_{i\,i}}$, $n = 568,455$, $nnz = 8,330,429$

matrix id	5th	6th	25th
dimKer$\widetilde{A}$	0	2	2
diagonal entries of S by QR	8.574766e+07	1.996129e+07	1.943695e+08
	8.902046e+04	9.001534e+04	4.834031e+02
	2.063527e-01	3.050581e-01	2.165032e-01
	1.799643e-01	2.632021e-01	2.034221e-01
	5.406636e-11	4.885182e-13	2.373664e-10
	1.286903e-13	2.328076e-15	2.063879e-16
$\|\|\widetilde{A}v_k\|\|$		4.376530e-16	1.102714e-16
		1.565334e-15	7.982637e-17
$\|\|\widetilde{A}^T w_k\|\|$		1.982592e-17	3.589934e-18
		1.983189e-17	3.589934e-18
(v_k, w_l)		4.482117e-12	2.802405e-13
		9.361029e-15	9.199130e-14
		2.251826e-13	2.810891e-13
		3.187650e-14	9.222188e-14
error	7.793333e-32	2.147198e-29	6.465044e-23
residual	5.901230e-32	5.658071e-32	1.605194e-30

- for 5th matrix, by assuming 2 dimensional kernel, $||\widetilde{A}v_1||$=1.286904e-13
- error and residual are computed by a manufactured solution

performance comparison

$n = 568,455$, $nnz = 8,330,429$, i7-6770HQ CPU @ 2.60GHz, g++-5, MKL BLAS

matrix id # postponed	1st w/o postponing	25th 6	25th 6
dimKer$\widetilde{A}$	0	2	0
diagonal entries of S by QR		1.943695e+08 4.834031e+02 2.165032e-01 2.034221e-01 2.373664e-10 2.063879e-16	1.799841e+08 4.161368e+02 2.975576e-01 2.440021e-01 2.158641e-10 1.726983e-16
$\|\|\widetilde{A}v_k\|\|$		1.102714e-16 7.982637e-17	2.063879e-16 1.188300e-20
$\|\|\widetilde{A}^T w_k\|\|$		3.589934e-18 3.589934e-18	
error residual	1.272821e-15 2.442552e-15	6.465044e-23 1.605194e-30	8.784892e-23 1.584077e-30
time for factorization(sec.)	double 22.580	quasi-quadruple 32.010	quadruple 5,620.7

- in quasi-quadruple computation, $A_{11}^{-1}A_{12}$ is computed by iterative refinement with 3 iterations
- in whole quadruple computation, S is generated by recursive computation following elimination tree

Summary

- Iterative refinement process for computing Schur complement matrix consisting postponed pivots can drastically reduce computational time without loosing accuracy
- Full pivoting procedure works as a preconditioner for kernel detection algorithm based on symmetric pivoting
- Complexity of quadruple precision arithmetic by double-double is 20 times large as double precision arithmetic. However, combination of inner iterative refinement and quadruple arithmetic for the last Schur complement requires only 50% increased elapsed time than double precision, which is numerically verified by matrices in a semi-conductor simulation

ongoing

- optimizing computation of iterative refinement by replacing multiple SpMV by SpMM

source code of Dissection is accessible within FreeFem++ repository
`https://github.com/FreeFem/FreeFem-sources/tree/master/download/dissection`

under GPL linking-exception / CeCILL-C licenses