FreeFEM days 2020, 11 Dec.

Quadruple arithmetic computation for FreeFEM with application to a semi-conductor problem

Atsushi Suzuki[1]

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

outline

- a semi-conductor problem by mixed formulation
- N-P-N device problem results in very high condition number
- element stiffness matrix by quadruple precision
- solution of linear system in quadruple accuracy precision by Dissection sparse matrix solver
- numerical example of floating phenomena of N-P-N device
- ongoing project to implement quadruple accuracy arithmetic into FreeFEM

drift-diffusion equation with Slotboom variables : 1/3

unknowns:
φ : electro-static potential
n : electron concentration
p : hole concentration
nondimensionalized drift-diffusion system : De Mari scaling

$$\begin{aligned} -\mathrm{div}(\lambda^2 \nabla \varphi) &= p - n + C(x) \\ -\mathrm{div} J_n &= 0 \qquad J_n = \nabla n - n \nabla \varphi \\ \mathrm{div} J_p &= 0 \qquad J_p = -(\nabla p + p \nabla \varphi) \end{aligned}$$

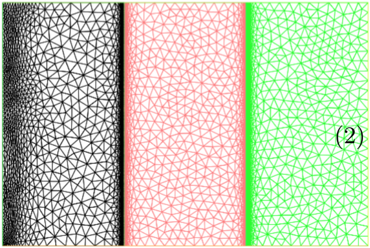
Slotboom variables η and ξ. $\quad n = e^{\varphi}\eta$, $p = e^{-\varphi}\xi$

$$\begin{aligned} -\mathrm{div}(\lambda^2 \nabla \varphi) &= e^{-\varphi}\xi - e^{\varphi}\eta + C(x) \\ -\mathrm{div} J_n &= 0 \qquad J_n = e^{\varphi}\nabla(e^{-\varphi} n) = e^{\varphi}\nabla \eta \qquad e^{-\varphi} J_n = \nabla \eta \\ \mathrm{div} J_p &= 0 \qquad J_p = -e^{-\varphi}\nabla(e^{\varphi} p) = -e^{-\varphi}\nabla \xi \qquad e^{\varphi} J_p = -\nabla \xi \end{aligned}$$

problem of an N-P-N device of semi-conductor

unknowns:
φ : electro-static potential
n : electron concentration
p : hole concentration

(3)

(4)  (2)

(1)

N region : $x < 0.1$ or $x > 0.2$,
$\tilde{C}(x,y) = n_d = 10^{20}$
P region : others
$\tilde{C}(x,y) = -n_a = -\beta \times 10^{17}$
$\tilde{n}\,\tilde{p} = n_i^2,\ n_i = 1.08 \times 10^{10}$
charge neutrality
$\tilde{p} - \tilde{n} + \tilde{C}(x,y) = 0$ on (2), (4).

thermal voltage
$V_{th} = K_B T/q = 0.026$

bias
$\tilde{\varphi}_{app} = 0.01, \quad \tilde{\varphi}_{app}/V_{th} \simeq 0.38$

dimensionless unknowns : φ, n, p
thermal equilibrium $n\,p = 1$

(2) $\quad \varphi = \sinh^{-1}(\frac{n_d}{2n_i}) \qquad n_0 = \sqrt{1 + \frac{1}{4}(\frac{n_d}{n_i})^2} + \frac{n_d}{2n_i},\ n = n_0,\ p = \frac{1}{n_0}$

(4) $\quad \varphi = \sinh^{-1}(\frac{n_d}{2n_i}) + \frac{\tilde{\varphi}_{app}}{\tilde{V}_{th}} \qquad n = n_0 e^{-\tilde{\varphi}_{app}/\tilde{V}_{th}},\ p = e^{\tilde{\varphi}_{app}/\tilde{V}_{th}}/n_0$

$(1),(3)$ $\quad \partial_\nu \varphi = 0 \qquad \partial_\nu n = 0,\ \partial_\nu p = 0$

drift-diffusion equation with Slotboom variables : 2/3

weak-formulation

$$V(g_\varphi) := \{\psi \in H^1(\Omega)\,;\, \psi = g_\varphi \text{ on } \Gamma_D\}$$
$$\Sigma = \{v \in H(\text{div})\,;\, v\cdot\nu = 0 \text{ on } \Gamma_N\}$$

nonlinear problem to find $(\varphi, J_n, \eta, J_p, \xi)$

$$\lambda^2 \int_\Omega \nabla\varphi\cdot\nabla\psi = \int_\Omega (e^{-\varphi}\xi - e^{\varphi}\eta + C)\psi \qquad \forall \psi \in V(0)$$
$$\int_\Omega e^{-\varphi} J_n\cdot v + \int_\Omega \eta\nabla\cdot v - \int_\Omega \nabla\cdot J_n\, q = \int_{\Gamma_D} \eta\, v\cdot\nu \qquad \forall (v,q) \in \Sigma\times L^2(\Omega)$$
$$\int_\Omega e^{\varphi} J_p\cdot v - \int_\Omega \xi\,\nabla\cdot v + \int_\Omega \nabla\cdot J_p\, q = -\int_{\Gamma_D} \xi\, v\cdot\nu \qquad \forall (v,q) \in \Sigma\times L^2(\Omega)$$

Fréchet derivative for $(\delta\varphi, \delta J_n, \delta\eta, \delta J_p, \delta\xi)$ with fixed $(\varphi, J_n, \eta, J_p, \xi)$

$$\lambda^2 \int_\Omega \nabla\delta\varphi\cdot\nabla\psi + \int_\Omega (e^{-\varphi}\delta\varphi\xi - e^{-\varphi}\delta\xi + e^{\varphi}\delta\varphi\eta + e^{\varphi}\delta\eta)\psi$$
$$\int_\Omega \left\{e^{-\varphi}(-\delta\varphi)J_n\cdot v + e^{-\varphi}\delta J_n\cdot v\right\} + \int_\Omega \delta\eta\nabla\cdot v - \int_\Omega \nabla\cdot\delta J_n\, q$$
$$\int_\Omega \left\{e^{\varphi}\delta\varphi J_p\cdot v + e^{\varphi}\delta J_p\cdot v\right\} - \int_\Omega \delta\xi\,\nabla\cdot v + \int_\Omega \nabla\cdot\delta J_p\, q$$

drift-diffusion equation with Slotboom variables : 3/3

Newton iteration: $dF(x^k)[\delta x] = -F(x^k)\,, \quad x^{k+1} = x^k + \delta x$

Jacobian matrix :

$$
\begin{bmatrix}
A_\varphi(\varphi^k, \eta^k, \xi^k) & 0 & C_\varphi(\varphi^k) & 0 & -C_\varphi(-\varphi^k) \\
-D_{J_n}(\varphi^k, J_n^k) & M(-\varphi^k) & B^T & & \\
 & -B & 0 & & \\
D_{J_p}(\varphi^k, J_p^k) & & & M(\varphi^k) & -B^T \\
 & & & B & 0
\end{bmatrix}
\begin{bmatrix}
\delta\varphi \\ \delta J_n \\ \delta\eta \\ \delta J_p \\ \delta\xi
\end{bmatrix}
$$

each block is defined by following bilinear form

$$A_\varphi(\varphi^k, \eta^k, \xi^k)\delta\varphi \;\leftrightarrow\; \lambda^2 \int_\Omega \nabla\delta\varphi \cdot \nabla\psi + \int_\Omega (e^{-\varphi^k}\delta\varphi\,\xi^k + e^{\varphi^k}\delta\varphi\,\eta^k)\psi$$

$$C_\varphi(\varphi^k)\delta\eta \;\leftrightarrow\; \int_\Omega e^{\varphi^k}\delta\eta\psi, \qquad D_{J_n}(\varphi^k, J_n^k)\delta\varphi \;\leftrightarrow\; \int_\Omega e^{-\varphi^k}\delta\varphi\, J_n^k \cdot v$$

$$M(-\varphi^k)\delta J_n \;\leftrightarrow\; \int_\Omega e^{-\varphi}\delta J_n \cdot v$$

$$B^T\delta\eta \;\leftrightarrow\; \int_\Omega \delta\eta\nabla \cdot v$$

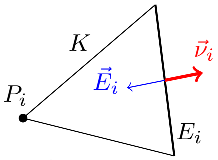
FreeFEM script for linearlized mixed form for hole unknowns

$$\int_\Omega e^{\varphi^k}\delta J_p\cdot v - \int_\Omega \delta\xi\,\nabla\cdot v + \int_\Omega \nabla\cdot\delta J_p\, q = \int_\Omega e^{\varphi^k}\delta\varphi J_p^k\cdot v + \cdots$$

```
fespace Vh(Th, RT0); fespace Qh(Th, P1); fespace Xh(Th, P1);
Vh [up1, up2], [v1, v2], [up1k, up2k];   Qh pp, q;
Xh phik, phi;
varf massexp([up1, up2], [v1, v2]) =
  int2d(Th)(exp(phik)*(up1*v1+up2*v2))
  + on(neumann, up1=0.0, up2=0.0);
varf divup([up1, up2], q)=
  int2d(Th)(q*(dx(up1)+dy(up2)));
varf RHSp([up1, up2], [v1, v2]) =
int2d(Th)(exp(phik)*phi*(up1k*v1+up2k*v2))
  + on(neumann, up1=0.0, up2=0.0);
matrix M=massexp(Vh, Vh);
matrix B=divup(Vh, Qh);
matrix A=[[M, B], [B', 0]]; //'
set(A, solver=sparsesolver, tolpivot=1.0e-2);
real[int] rhsJp = RHSp(0, Vh);
real[int] zero(Xh.ndof); zero =0.0;
real[int] rhs = [rhsJp, zero];
real[int] sol=A^-1 * rhs;
```

- double precision arithmetic is not enough for matrix generation and linear solver

Raviart-Thomas finite element



$RT0(K) = (P0(K))^2 + \vec{x}P0(K) \subset (P1(K))^2$.

- K : triangle element
- $\{E_i\}$: edges of K
- $\vec{\nu}_i$: outer normal of K on E_i
- $\vec{E}_i$: normal to edge E_i given by whole triangulation

$$\vec{v} \in RT0(K) \;\Rightarrow\; \vec{v}|_{E_i} \cdot n_i \in P0(E_i), \quad \mathrm{div}\vec{v} \in P0(K)$$

finite element basis

$$\vec{\Psi}_i(\vec{x}) = \sigma_i \frac{|E_i|}{2|K|}(\vec{x} - \vec{P}_i) \quad \sigma_i = \vec{E}_i \cdot \vec{\nu}_i, \quad P_i : \text{ node of } K$$

$$\int_K e^{\varphi_h} \vec{\Psi}_j \cdot \vec{\Psi}_i \;\leftarrow\; \int_K e^{\varphi_1\lambda_1 + \varphi_2\lambda_2 + \varphi_3\lambda_3} \lambda_k \lambda_l \quad \text{by exact quadrature}$$

$$\int_K e^{\varphi_h} \vec{\Psi}_j \cdot \vec{\Psi}_i \;\simeq\; \frac{1}{|K|} \int_K e^{\sum_k \varphi_k \lambda_k} \int_K \vec{\Psi}_j \cdot \vec{\Psi}_i \quad : \text{exponential fitting}$$

$$\int_K e^{\varphi_h} \vec{\Psi}_j \cdot \vec{\Psi}_i \;\simeq\; |K| \sum_k \omega_k e^{\sum_k \varphi_k(x_k)\lambda_k} \vec{\Psi}_j(x_k) \cdot \vec{\Psi}_i(x_k) \; : \text{numerical quadrature}$$

$\{\lambda_1, \lambda_2, \lambda_3\}$: barycentric coordinates of K
$\{\omega_k, x_k\}$ weight and point of numerical quadrature

exact integration of shape functions with exponential weight

N-relative exponential functions

$$\mathsf{exp1}(x) := \frac{e^x - 1}{x} \qquad \mathsf{exp1}^{(1)}(x) = \mathsf{exp1}(x) - \frac{1}{2}\mathsf{exp2}(x)$$

$$\mathsf{exp2}(x) := 2!\frac{e^x - 1 - x}{x^2} \qquad \mathsf{exp2}^{(1)}(x) = \mathsf{exp2}(x) - \frac{2}{3}\mathsf{exp3}(x)$$

$$\mathsf{exp3}(x) := 3!\frac{e^x - 1 - x - \frac{x^2}{2}}{x^3} \qquad \mathsf{exp3}^{(1)}(x) = \mathsf{exp3}(x) - \frac{3}{4}\mathsf{exp4}(x)$$

$$\begin{aligned}
\int_{\hat{K}} e^{\varphi_1\lambda_1+\varphi_2\lambda_2+\varphi_3\lambda_3}\lambda_1^2\lambda_2 &= \frac{e^{\varphi_1}}{3(\varphi_2-\varphi_3)}\left\{\mathsf{exp3}^{(1)}(\varphi_2-\varphi_1) - \right. \\
&\qquad \left. \frac{1}{\varphi_2-\varphi_3}\left(\mathsf{exp3}(\varphi_2-\varphi_1) - \mathsf{exp3}(\varphi_3-\varphi_1)\right)\right\} \\
&= \frac{e^{\varphi_1}}{3}\left\{\frac{1}{2!}\mathsf{exp3}^{(2)}(\varphi_2-\varphi_1) + \frac{1}{3!}\mathsf{exp3}^{(3)}(\varphi_2-\varphi_1)(\varphi_3-\varphi_2)\right. \\
&\qquad \left. + \frac{1}{4!}\mathsf{exp3}^{(4)}(\varphi_2-\varphi_1)(\varphi_3-\varphi_2)^2 + \cdots\right\} \\
&= \frac{e^{\varphi_1}}{3}\left\{\frac{1}{20} + \frac{3\widetilde{\varphi_2}+\widetilde{\varphi_3}}{130} + \frac{6\widetilde{\varphi_2}^2+4\widetilde{\varphi_2}\,\widetilde{\varphi_3}+\widetilde{\varphi_3}^2}{840} + \right. \\
&\qquad \left. \frac{10\widetilde{\varphi_2}^3+10\widetilde{\varphi_2}^2\,\widetilde{\varphi_3}+5\widetilde{\varphi_2}\,\widetilde{\varphi_3}^2+\widetilde{\varphi_3}^3}{6720} + \cdots\right\}
\end{aligned}$$

$\widetilde{\varphi_2} = \varphi_2 - \varphi_1, \widetilde{\varphi_3} = \varphi_3 - \varphi_2$

numerical quadrature with higher accuracy up to 15th order

$$\int_K f(x) = \sum_k \omega_k f(x_k) \quad \{\omega_k, x_k\}$$: weight and point of numerical quadrature

Symmetric integration points and weight in a triangle are obtained by numerical optimization

L. Zhang, T. Cui, “A set of symmetric quadrature rules on triangles and tetrahedra”, Journal of Computational Mathematics, 27(1) pp.89-96, 2009.

http://lsec.cc.ac.cn/phg/download/

- Newton iteration by `bc` command with 74 digits, which can cover octuple precision, starting from double precision data

```
// dim: 2, points: 7, equations: 5, unknowns: 5, symmetry: true
// Newton step 0, error = 4.5945e-35, residual = 10.0000
// Newton step 1, error = 8.0661e-70, residual = 1.9444e-36
// Newton step 2, error = 4.9378e-139, residual = 1.0929e-70
// Newton step 3, error = 1.2610e-157, residual = 1.2352e-139
SymStar2D_P5.init(5, 7);
SymStar2D_P5.wp21(
".12593918054482715259568394550018133365763923191225700764451035519490543076",
".10128650732345633880098736191512382805557515689087662730535363018507223372");
SymStar2D_P5.wp21(
".13239415278850618073764938783315199967569410142107632568882297813842790258",
".47014206410511508977044120951344760051585341453769480126607494124349919485");
SymStar2D_P5.wp3(
".22500000000000000000000000000000000000000000000000000000000000000000000000",
".33333333333333333333333333333333333333333333333333333333333333333333333333");
```

double-double data for quadruple accuracy

double-double is less accurate than IEEE754 quadruple floating point

	15	112	quadruple

	11	52	double		11	52	double

but achieves faster computation using hardware for double preicision

- qd library by Y. Hida, X. S.Li and D. H. Bailey

in FreeFEM

```
fespace Vh(Th, RT0); fespace Qh(Th, P1);
Vh [up1, up2], [v1, v2], [up1k, up2k];
varf massexp([up1, up2], [v1, v2]) =
  int2d(Th)(exp(phik)*(up1*v1+up2*v2))
  + on(neumann, up1=0.0, up2=0.0);
complex[int] phiq(Xh.ndof), xq(Vh.ndof+Qh.ndof);
matrix<complex> Mq=massexp(Vh, Vh);
expmassq(Th, Mq, phiq) // dynamic loading library
complex[in] kernelsq(1), kernelstq(1); // storing kernels
DissectionSolveq(Aq,xq,nkernel,kernelsq,kerneltsq,tgv,tolpiv);
```

- array `complex[int]` keeps lower and higher values as real and imaginary components
- `matrix<complex>` keeps double-double data with the same sparse nonzero pattern

Dynamic loading library for double-double data

```
AnyType expmatrix_Op<Complex>::operator() (Stack stack) const
{
  const Mesh *pTh = GetAny<const Mesh *>((*xth)(stack));
  Matrice_Creuse<Complex> *sparsemat =
      GetAny<Matrice_Creuse<Complex> *>((*xsparsemat)(stack));
  KN<Complex> *phic = GetAny<KN<Complex> *>((*xphi)(stack));
  const Mesh &Th = *pTh;
  MatriceMorse<Complex> *AA = sparsemat->pHM();
  int *irow,  *jcol;  Complex *aval;
  AA->setfortran(false);
  AA->CSR(irow, jcol, aval);                           // access to CRS
  quadruple *avalq = (quadruple *)aval;                // casting
  for (int i=0; i < AA->size(); i++) avalq[i]=qaudruple(0.0);
  quadruple *phiq = (quadruple *)&(*phic)[0];          // casting
  expmatrixcalc(pTh, irow, jcol, avalq, phiq);
  return 1L;
}
template<typename T>
void expmatrixcalc(const Mesh *pTh, int *irow, int *jcol,
                   T *aval, T *phi)
{
  const Mesh &Th = *pTh;
  FESpace *pVh = new FESpace(*pTh, RTLagrange); // for matrix
  FESpace *pXh = new FESpace(*pTh, P1Lagrange); // for phi
  FESpace &Vh = *pVh; FESpace &Xh = *pXh;
  for (int k = 0; k < Th.nt; k++) { // element mass matrix
    for (int i = 0; i < 3; i++) {
      px[i] = T(Th(Th(kkk,i)).x); py[i] = T(Th(Th(kkk,i)).y);
      int ii = Vh(k, i); // DOF of vector RT0 element
      for (int ll = irow[ii]; ll < irow[ii + 1]; ll++) {
        if (jcol[ll] == // access CSR entry
```

ongoing project to integrate quadruple precision into FreeFEM

introducing quadruple data type realized by double-double

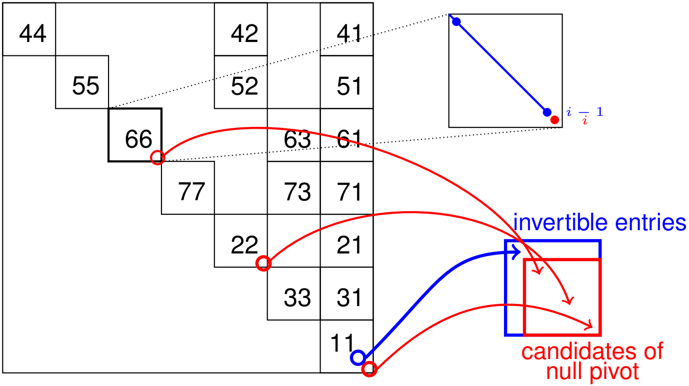
- the first version can work only with quadruple instead of double by replacing C++ object `real` by `quadruple` (80% done)
- mixed usage of matrices by double and quadruple data in one script will be in the future

```
fespace Vh(Th, RT0); fespace Qh(Th, P1); fespace Xh(Th, P1);
Vh<quadruple> [up1, up2], [v1, v2], [up1k, up2k];
Qh<quadruple> pp, q;
Xh<quadruple> phik, phi;
varf massexp([up1, up2], [v1, v2]) =
  int2d(Th, qft=qf9pT)(exp(phik)*(up1*v1+up2*v2))
  + on(neumann, up1=0.0, up2=0.0);
// ...
varf RHSp([up1, up2], [v1, v2]) =
  int2d(Th)(exp(phik)*phi*(up1k*v1+up2k*v2));
  + on(neumann, up1=0.0, up2=0.0);
matrix<quadruple> M=massexp(Vh, Vh);
matrix<quadruple> A=[[M, B], [B', 0]]; //'
set(A, solver=sparsesolver, tolpivot=1.0e-2);
quadruple[int] rhsJp = RHSp(0, Vh);
quadruple[int] zero(Xh.ndof); zero = 0.0;
quadruple[int] rhs = [rhsJp, zero];
quadruple[int] sol=A^-1 * rhs;
```

Dissection sparse direct solver written by C++ template

- nested-dissection ordering by `SCOTCH` or `METIS`

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

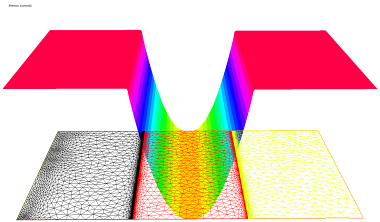


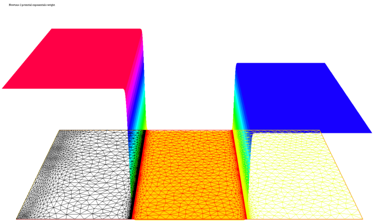
- Schur complement from postponed pivots will be examined by kernel detection algorithm
- C++ template implementation allows quadruple / octuple arithemtic

optimized BLAS 3; dgemm, dtrsm to get performance

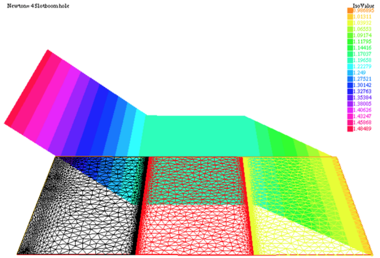
pseudo kernel of N-P-N problem (floating phenomena)

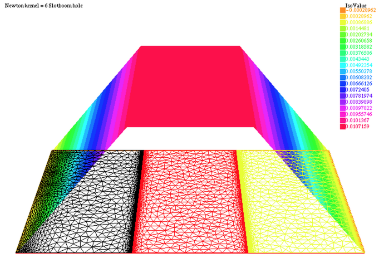
with bias $\varphi_{app} = 0.3846$,   at 6-th Newton iteration





(left) : thermal equilibrium $-18.13 \leq \phi \leq 24.45$, (right) : e^{φ} : range$\sim 3.1 \times 10^{18}$





(left) : solution of ξ,

(right) : pseudo kernel vector of ξ

detected pseudo kernel of N-P-N problem by Dissection

stiffness matrix (90,631 DOF) by quadruple precision using QD library
postponed pivot = 86 during global symm. factorization with $\tau = 10^{-2}$
postponed pivot = 1 by sym. pivoting during re-factorization of 86×86

diagonal entries of inflated 6×6 matrix (6=1+4+1) by QR factorization

1	1.3325798227230853655416520747392e+00
2	6.4240269936482403641445882556516e-01
3	3.6261212845073655501562402844582e-01
4	2.1213856832382004516781285182289e-01
5	2.4163468634185244678350896403378e-17
6	1.2770719422459101423055812467233e-31

matrix residual : $\beta_p = ||\widetilde{A_p^{-1}} A_p - I_p||_\infty$ $\widetilde{A_p^{-1}}$: inverse with pertubation

1	4.9303806576313199521478151448458e-32
2	9.8607613152626399042956302896916e-32
3	1.4791141972893959856443445434537e-31
4	8.7496110059860563499793450832972e-32
5	9.3937963997437069736249355582165e-16
6	3.6258408001413823623948870895643e-01

error of kernel vectors with supposed dimension of inflated matrix

$k=2$	$k=3$
2.6846625560958350e-17	2.8823202208084498e-01
1.7688955507704848e-17	2.6846625560958350e-17
	1.8991343789490032e-01

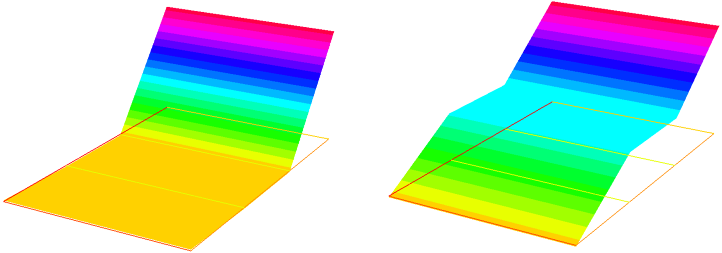
Newton iteration for an N-P-N device problem

Jacobian matrix of Newton iteration is quasi-singular due to floating phenomena

- assuming the Jacobian matrix is invertible
  $\Rightarrow$ condition number is large $\simeq 10^{16}$
- assuming the Jacobian matrix is singular
  $\Rightarrow$ kernel vector of the Jacobian matrix is computed by an extra minimization problem with Hessian only for kernel vector

solution of hole density in Slotboom variable

na $= 6 \times 10^{17}$, $\varphi_{\text{app}} = 0.3846$

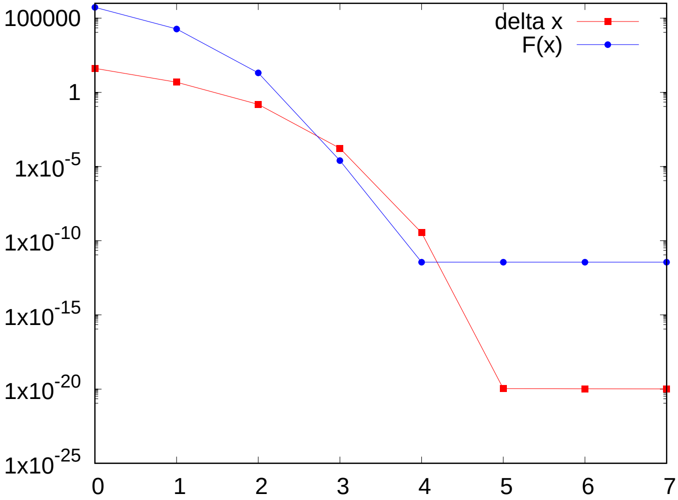


singular matrix　　　　+ adjusting / invertible

Numerical results of N-P-N device problem

Newton iteration by assuming matrix is singular + adjusting (Hessian for kernel vector in double precision)

na$= 6 \times 10^{17}$ $\varphi_{\text{app}} = 0.3846$



Summary

- Computation of element stiffness matrix by quadruple precision is mandatory for N-P-N device problem
- Dissection solver can factorize sparse matrix given by quadruple precision
- Pseudo kernel of matrix in N-P-N device semi-conductor is well detected by Dissection
- numerical quadrature table is prepared up to octuple precision
- complex data type is used to store double-double data with computation of FE matrix by dynamic loading library
- Dissection can get solution in quadruple from matrix given by double

ongoing

- all arithmetic by quadruple precision instead of double in FreeFEM is available soon
- fespace, array, and matrices by double and quadruple need to coexist in FreeFEM script

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

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