Verified error bounds for sparse systems

Black box Matlab/Octave/INTLAB algorithms

Siegfried M. Rump, Hamburg/Tokio

- Algorithm I
- Algorithm II

  } s.p.d., symm./Hermitian, general
- Least squares/underdetermined
- Nonlinear systems
- Interval data
- from INTLAB Version 14

S.M. Rump: Verified error bounds for sparse systems Part I:
The splitting of a matrix into two factors, 49 pages
Verified error bounds for sparse systems Part II:
Inertia-based bounds, least squares and nonlinear systems, 48 pages

to appear in SIMAX (see my homepage)

Symmetric/Hermitian (positive definite) A

Let $B := A - sI$ for $0 < s \in \mathbb{R}$ and let $B \approx \tilde{G}\tilde{G}^T$

Define $E := \tilde{G}\tilde{G}^T - B$ and assume $\|E\|_2 \le e$

$$(*) \quad \sigma_{\min}(A) = \sigma_{\min}(\tilde{G}\tilde{G}^T + sI - E) \ge \sigma_{\min}(\tilde{G}\tilde{G}^T + sI) - \|E\|_2 \ge s - e$$

Thus $\alpha := s - e$ is a lower bound for $\sigma_{\min}(A)$. Hence

$$\|A^{-1}b - \tilde{x}\|_\infty \le \|A^{-1}b - \tilde{x}\|_2 = \|A^{-1}(b - A\tilde{x})\|_2 \le \alpha^{-1}\|b - A\tilde{x}\|_2$$

That might be applied to A^TA, but $\mathrm{cond}(A^TA) = \mathrm{cond}(A)^2$.

Challenge: Linear systems $Ax = b$ with general sparse A

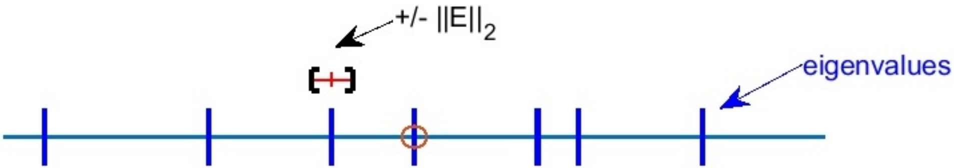
Verification for symmetric/Hermitian matrices [R., 1993, 1995]

Theorem 1 *Let symmetric $A \in \mathbb{R}^{n\times n}$, $0 < \tilde\lambda \in \mathbb{R}$ and $\tilde L_1, \tilde D_1, \tilde L_2, \tilde D_2 \in \mathbb{R}^{n\times n}$ be given. If the inertia of $\tilde D_1$ and $\tilde D_2$ are equal, then for any matrix norm*

$$\sigma_{\min}(A) > \tilde\lambda - \max\{\|A - \tilde\lambda I - \tilde L_1 \tilde D_1 \tilde L_1^T\|, \|A + \tilde\lambda I - \tilde L_2 \tilde D_2 \tilde L_2^T\|\}. \quad (1)$$

If all eigenvalues of $\tilde D_1$ are positive, then

$$\sigma_{\min}(A) > \tilde\lambda - \|A - \tilde\lambda I - \tilde L_1 \tilde D_1 \tilde L_1^T\|. \quad (2)$$



Not pursued because lack of robustness of LDL^T-decomposition.

Nowadays robust LDL^T-decompositions are available

Good scaling and equilibration routines are available, in particular

Iain S. Duff: Ma57 - a code for the solution of sparse symmetric definite and indefinite systems, ACM TOMS, 30(2):118-144, 2004.

That makes the old methods attractive, observed in

T. Terao, K. Ozaki: Method for verifying solutions of sparse linear systems with general coefficients, Appl. Math. Comp. 490, 2025.

A drawback of $B - LDL^T$ is that the product of 3 matrices is needed.

That triggered a revival with several improvements of my old methods

Factor $A \approx F_1 F_2$ such that $A - F_1 F_2$ requires only one dot product

We present two main algorithms and treat three cases separately:

I. Symmetric/Hermitian positive definite

II. general symmetric/Hermitian

III. general real/complex

Until further notice discussion of Algorithm I

Many details to be considered

Back
Close

Lower bound for $\sigma_{\min}(A)$ for symmetric (s.p.d.) A

Recall $\sigma_k(A) = \lambda_k(A) > 0$

$$s \lesssim \sigma_{\min}(A)$$
$$B \approx \tilde{G}\tilde{G}^T$$
$$E := \tilde{G}\tilde{G}^T - B \text{ and } \|E\|_2 \le e$$

$$\Rightarrow \quad \sigma_{\min}(A) \ge s - e$$

Needs I) approximation s of $\sigma_{\min}(A)$ and

II) upper bound for $\|E\|_2$

I) Approximation of $\sigma_{\min}(A)$ for symmetric A

i) `svds(A,1,'smallestnz')` fast, but often inaccurate

ii) `s = abs(eigs(A,1,'smallestabs'))`

- Robust and accurate, but sometimes slow
- iterative based on decomposition of A

In most cases a factor of A is already computed, therefore

iii) Few inverse power iterations based on factor of A

II) Upper bound for $\|E\|_2$

Common bound $\|E\|_2 \le \sqrt{\|E\|_1 \|E\|_\infty}$

Better use Collatz-Wieland bound for the Perron root

$$0 \le A \in M_n(\mathbb{R}),\ 0 < x \in \mathbb{R}^n : \quad \varrho(A) \le \max_k \frac{(Ax)_k}{x_k}$$

For general E

$$\|E\|_2 \le \||E|\|_2 = \sigma_{\max}(|E|) = \sqrt{\lambda_{\max}(|E|^T|E|)} \le \max_k \frac{(|E|^T(|E|x))_k}{x_k}$$

For symmetric/Hermitian E

$$\|E\|_2 \le \max_k \frac{(|E|x)_k}{x_k} \qquad \text{both for } any\ 0 < x \in \mathbb{F}^n$$

Few power iterations on x yield a very good bound on $\||E|\|_2$

Since E is an inclusion of a matrix residual, usually $\mathrm{mag}(E) \approx |E|$

Lower bound for $\sigma_{\min}(A)$ for symmetric (s.p.d.) A (cont'd)

Recall $\sigma_k(A) = \lambda_k(A) > 0$

$$s \lesssim \sigma_{\min}(A)$$
$$B \approx \tilde{G}\tilde{G}^T$$
$$E := \tilde{G}\tilde{G}^T - B \text{ and } \|E\|_2 \le e$$

$\Rightarrow \quad \sigma_{\min}(A) \ge s - e$

- approximation s of $\sigma_{\min}(A)$ $\checkmark$
- upper bound for $\|E\|_2$ $\checkmark$

$\rightarrow$ III) Need to compute E such that $|\tilde{G}\tilde{G}^T - B| \le E$

III) A priori bounds on $\|\tilde{G}\tilde{G}^T - B\|$ for $\tilde{G}\tilde{G}^T \approx B$

Lemma [Demmel, 1989] Let the floating-point Cholesky decomposition of B run to completion with Cholesky factor $\tilde{G}$. Then

$$|E_{ij}| \le \frac{(n+1)\mathbf{u}}{1-(n+1)\mathbf{u}}(\tilde{G}_{ii}\tilde{G}_{jj})^{1/2}$$

for $E := \tilde{G}\tilde{G}^T - B$.

Lemma [R./Ogita, 2006] With the assumptions as before

$$\|E\|_2 \le \sum_{k=1}^{n} \varphi_{k+1} B_{kk} \text{ with } \varphi_k := \frac{\gamma_k}{1-\gamma_k} \text{ and } \gamma_k := \frac{k\mathbf{u}}{1-k\mathbf{u}}$$

Rigorous estimate based on *floating-point Cholesky decomposition*

A priori bounds on $\|\tilde{G}\tilde{G}^T - B\|$ for sparse B with $\tilde{G}\tilde{G}^T \approx B$ (cont'd)

Lemma [R., 2025] Let the floating-point Cholesky decomposition of B run to completion with Cholesky factor $\tilde{G}$.

$E := \tilde{G}\tilde{G}^T - B$

μ_k number of nonzero elements in $\tilde{G}_{*k}$

$\Phi_{ij} := \min(\mu_i, \mu_j) + 1$

D diagonal with $D_{kk} := \left(\dfrac{B_{kk}}{1 - \Phi_{kk}\mathbf{u}}\right)^{1/2}$

Then $\|E\|_2 \le \mathbf{u}\|D\Phi D\|_2 \le \mathbf{u}\max_k \dfrac{(D\Phi Dx)_k}{x_k}$ for every $0 < x \in \mathbb{R}^n$

Φ is full, so how to compute $\Phi(Dx)$?

A priori bounds on $\|\tilde{G}\tilde{G}^T - B\|$ for sparse B with $\tilde{G}\tilde{G}^T \approx B$ (cont'd)

Recall $\|E\|_2 \le \mathbf{u} \max_k \dfrac{(D\Phi Dx)_k}{x_k}$ for every positive $x \in \mathbb{R}^n$

for $\Phi_{ij} := \min(\mu_i, \mu_j) + 1$

Lemma [Lange, 2025] Let $0 < v \in \mathbb{R}^n$ be sorted in ascending order and define $\Phi \in \mathbb{R}^{n \times n}$ by $\Phi_{ij} := \min(v_i, v_j)$.

Then for $0 < x \in \mathbb{R}^n$ the vector $\mathtt{w}$ computed by

$$\begin{aligned}&\mathtt{rcx} = \mathtt{cumsum}(\mathtt{x}, 1, '\mathtt{reverse}');\\ &\mathtt{vx} = \mathtt{v.} * \mathtt{x};\\ &\mathtt{w} = \mathtt{cumsum}(\mathtt{vx}) - \mathtt{vx} + \mathtt{v.} * \mathtt{rcx};\end{aligned}$$

is equal to Φx. $\mathcal{O}(n)$ ops

$\Rightarrow$ Improves the previous estimate by an order of magnitude.

Scaling and equilibration

Businger 1968: $A = A^H \;\Rightarrow\;$ optimal diagonal scaling is symmetric

$$\inf_{D_1,D_2\in\mathcal{D}_n} \kappa(D_1 A D_2) = \inf_{D\in\mathcal{D}_n} \kappa(DAD)$$

v.d. Sluis 1969: A s.p.d. with diagonal scaled to 1 $\;\Rightarrow$

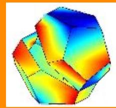
$$\kappa(A) \le q \min_{D\in\mathcal{D}_n} \kappa(D^H A D)$$

for q denoting the maximum number of nonzeros per row of A

Forsythe/Straus 1958, Bauer 1963:

Equilibrate $|A|$ to a scalar multiple of a doubly stochastic matrix

$\rightarrow$ Sinkhorn/Knopp algorithm



Scaling and equilibration for s.p.d. A

Scale A to have entries near 1

```
d = pow2(round(log2(1./sqrt(diag(A)))));
A = ( d .* A) .* d';
```

Sinkhorn/Knopp `d=1./(abs(A)*d)` starting with `d=ones(n,1)`

We use

```
d = 1./sqrt(diag(A));
for k = 1 : 4, d = 1./(abs(A) * d); end
A = (d. * A). * d';
```

i.e., 2 iteration steps scaling rows and columns.

Right hand side is scaled and transformed accordingly

Scaling and equilibration for symmetric/Hermitian A

Different diagonal scaling because A_{kk} may be zero

We choose the columnwise maximum $d_k := \Sigma_\ell |A_{k\ell}|$

We use

```
d = 1./vecnorm(A,2)';
for k = 1:4, d = 1./(abs(A)*d); end
A = (d.*A).*d';
```

Scaling and equilibration for general A

Knight 2008: Convergence of Sinkhorn/Knopp algorithm

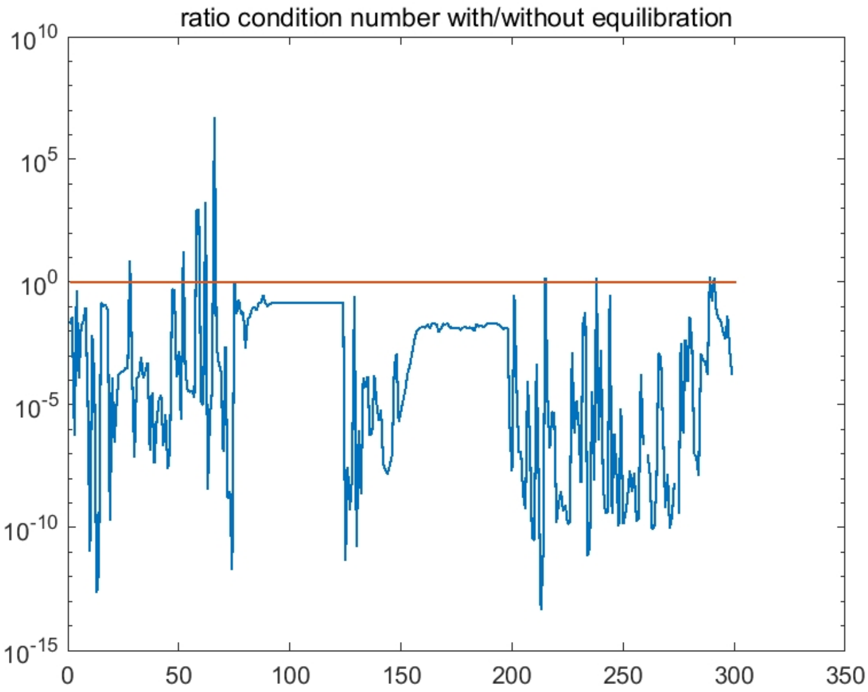
We use Matlab's **equilibrate** and add 2 Sinkhorn/Knopp iterations

```
[p,row,col] = equilibrate(A,'vector');
for k = 1:2
   col = 1./(abs(A(p,:)')*row);  row = 1./(abs(A(p,:))*col);
end
row = sign(row).*pow2(round(log2(abs(row))));
col = sign(row).*pow2(round(log2(abs(col))));
A = row.*A(p,:).*col';
```

Numerical evidence for the benefit of equilibration

All matrices from the Suite Sparse Matrix Collection with

$$10^3 \le m, n \le 10^5 \quad \text{and} \quad 10^7 \le \kappa(A) \le 10^{16} \quad \text{and} \quad \texttt{nnz(A)} \le 10^6$$



Matrix decomposition

For s.p.d. A we use `[R,FLAG,p] = chol(A,'vector');`

For $A = A^H$ we use `[L,D,p] = ldl(A,thresh,'vector');`
such that LDL^T approximates `A(p,p)`

The maximal value **`thresh`**$= 0.5$ gives good numerical results
Progress of "MA57" compared to previous versions of LDL^T

Another possibility with $LDL^T \approx$ `S(p,:)*A*S(:,p)`

```
[L,D,p,S] = ldl(A,thresh,'vector');
```

was sometimes useful, but often counterproductive

Matrix decomposition (cont'd)

Lemma For nonsingular A a block LDL^T decomposition of

$B := \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$ must produce D with all 2×2 blocks with zero diagonal

Matlab's `ldl` sets entries below a threshold to zero, but unfortunately also for the entries of D

→ In 54 out of 211 tests `ldl` produced a singular D

remedy:

```
[L,D,p] = ldl(B + 1e-50 * speye(n), thresh,'vector');
D(1:n+1:n^2) = 0;
```

The problem is fixed from Version 2026a of Matlab

Back
Close

Sparse symmetric (s.p.d.) matrices

1 function $[x, \delta]$ = verifySparseSPD(A,b)
2   If any $A_{kk} \le 0, [x, \delta]$ = verifySparseSym(A,b), return
3   Equilibrate A
4   Compute Cholesky factorization $\tilde{R}^T \tilde{R} \approx A$
5   If failed, $[x, \delta]$ = verifySparseSym(A,b), return
6   Compute $\tilde{s}(A, \tilde{R})$ and set $s := 0.9\tilde{s}$
7   `setround(-1); As = A - s * speye(n);`
8   `setround(0); [Rs, FLAG, p] = chol(As);`
9   If succeeded, goto step 13
10   Set rounding downwards and $As = As + (8s/10)I$; $s = s/5$
11   Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx As$ in rounding to nearest
12   If failed, $[x, \delta]$ = verifySparseSym(A,b), return
13   Compute upper bound α with $\|Rs^T Rs - As\|_2 \le \alpha$
14   If $\alpha \ge s$, compute α with $\|Rs^T Rs - As\|_2 \le \alpha$ using (**??**)
15   If $\alpha \ge s$, compute α with $\|Rs^T Rs - As\|_2 \le \alpha$ using (**??**)
16   If $\alpha \ge s$, verification failed, return
17    $[x, \delta]$ = ErrorBound$(A, b, s - \alpha, \text{“}solve\text{“})$ using $\tilde{R}$ for solve

Floating-point error estimates when using directed rounding

Jeannerod/R. (2013): $|\mathrm{float}(AB)_{ij} - (AB)_{ij}| \le \mu \mathbf{u}(|A||B|)_{ij}$

for a nearest rounding and at most μ nonzero products per entry

Ozaki et al. (2015): $|\mathrm{float}(AB)_{ij} - (AB)_{ij}| \le 2(\mu + 4)\mathbf{u}(AB)_{ij}$

for $A, B \ge 0$ and directed rounding

Lange/R. (2019): $|\mathrm{float}(AB)_{ij} - (AB)_{ij}| \le 2\mu \mathbf{u}(|A||B|)_{ij}$

for general A,B and directed rounding

Why is that important?

Why are floating-point error estimates in directed rounding important?

Goal: inclusion of AB

Typical approach:

`setround(0), C = A*B;` $\Rightarrow AB \in C \pm \mathbf{u}(|A||B|) \qquad (*)$

If not satisfactory:

`setround(-1), C1 = A*B;`

`setround(+1), C2 = A*B;` $\Rightarrow AB \in [C1, C2]$

Requires 3 matrix multiplications if $(*)$ not successful

Alternatively:

`setround(-1), C = A*B;` $\Rightarrow AB \in C \pm 2\mathbf{u}(|A||B|)$

If not satisfactory:

`setround(+1), C2 = A*B;` $\Rightarrow AB \in [C, C2]$

Floating-point residual error estimates for sparse A, B (cont'd)

Lemma Let $A \in \mathbb{F}^{m\times\ell}$ and $B \in \mathbb{F}^{\ell\times n}$ be given.

μ_i number of nonzero elements in A_{i*}

ν_j number of nonzero elements in B_{*j}

$\varrho_i = \|A_{i*}\|_2$ and $\sigma_j = \|B_{*j}\|_2$

Then using a nearest-rounding and any order of evaluation

$$\|\mathrm{float}(AB) - AB\|_2 \le \mathbf{u} \sum_{k=1}^{n} \min(\mu_k, \nu_k)\rho_k\sigma_k$$

without limit on n. For $C \in \mathbb{F}^{m\times n}$ and $E := \mathrm{float}(AB - C)$ it follows

$$\|\mathrm{float}(AB - C) - (AB - C)\|_2 \le \mathbf{u}\left(\|E\|_2 + \sum_{k=1}^{n} \min(\mu_k, \nu_k)\rho_k\sigma_k\right)$$

without limit on n.

For a faithful rounding basically replace $\mathbf{u}$ by $2\mathbf{u}$.

Sparse symmetric (s.p.d.) matrices

1 function $[x, \delta]$ = verifySparseSPD(A,b)
2 If any $A_{kk} \le 0$, $[x, \delta]$ = verifySparseSym(A,b), return
3 Equilibrate A
4 Compute Cholesky factorization $\tilde{R}^T \tilde{R} \approx A$
5 If failed, $[x, \delta]$ = verifySparseSym(A,b), return
6 Compute $\tilde{s}(A, \tilde{R})$ and set $s := 0.9\tilde{s}$
7 `setround(-1)`; `As = A - s * speye(n)`;
8 `setround(0)`; `[Rs, FLAG, p] = chol(As)`;
9 If succeeded, goto step 13
10 Set rounding downwards and $As = As + (8s/10)I$; $s = s/5$;
11 Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx As$ in rounding to nearest
12 If failed, $[x, \delta]$ = verifySparseSym(A,b), return
13 Compute α by Theorem 2 with $\|Rs^T Rs - As\|_2 \le \alpha$
14 If $\alpha \ge s$, compute α with $\|Rs^T Rs - As\|_2 \le \alpha$ using $\downarrow\uparrow$
15 If $\alpha \ge s$, compute α with $\|Rs^T Rs - As\|_2 \le \alpha$ using ext.prec.
16 If $\alpha \ge s$, verification failed, return
17 $[x, \delta]$ = ErrorBound$(A, b, s - \alpha$, "*solve*") using $\tilde{R}$ for solve

Back
Close

Narrow error bounds for the solution of a sparse linear system

$$\alpha \le \sigma_{\min}(A) \quad \Rightarrow \quad \|A^{-1}b - \tilde{x}\|_\infty \le \alpha^{-1}\|b - A\tilde{x}\|_2$$

Store approximate solution in 2 parts $\tilde{x} + \tilde{y}$ [R. 1980]

Notation: $[\![\cdot]\!]_{2,1}$ evaluated in extended and stored in double precision

```
[x̃, δ] = ErrorBound(A, b, s, "solve")          % ≤ σ_min(A)
    x̃ = solve(A, b)                             % A^{-1}b ≈ x̃
    ỹ = solve(A, [[b − Ax̃]]_{2,1})              % A^{-1}b ≈ x̃ + ỹ
    [x̃, ỹ] = TwoSum(x̃, ỹ)
    z̃ = solve(A, [[b − Ax̃ − Aỹ]]_{2,1})         % A^{-1}b ≈ x̃ + ỹ + z̃
    [x̃, ỹ] = TwoSum(x̃, ỹ + z̃)                   % A^{-1}b ≈ x̃ + ỹ
    setround(-1); ϱ = abs([[Ax̃ + Aỹ − b]]_{2,1})
    setround(+1); ϱ = max(ϱ, abs([[Ax̃ + Aỹ − b]]_{2,1}))
    δ = |ỹ| + vecnorm(ϱ)/s
```

implies entrywise estimates $|A^{-1}b - \tilde{x}| \le \delta$

That is the key to narrow inclusions.

Computational results

... to be displayed for s.p.d., symmetric and general matrices

Lower bounds for $\sigma_{\min}(A)$ for symmetric/Hermitian and general A

Factor $A \approx L_1 L_2$ such that L_1, L_2 have identical sets of singular values

Then

$$\sigma_{\min}(A) \approx \sigma_{\min}(L_1 L_2) \geq \sigma_{\min}(L_1)\sigma_{\min}(L_2) = \sigma_{\min}(L_1)^2 = \sigma_{\min}(L_1 L_1^T)$$

A true lower bound is obtained by

$$\sigma_{\min}(A) \geq \sigma_{\min}(L_1 L_2) - \|A - L_1 L_2\| \geq \sigma_{\min}(L_1 L_1^T) - \|A - L_1 L_2\|$$

For general matrices apply that scheme to $B := \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$

Method: split D such that $LDL^T \to L_1 L_2$

Back
Close

Factorization of 2×2 Hermitian matrix D

Three cases:

- D is 1×1
- D with zero diagonal
- general D

Third case $D = \begin{pmatrix} a & b \\ \bar{b} & c \end{pmatrix}$, define $d := \sqrt{(a-c)^2 + 4\bar{b}b}$

Eigenvalues $\lambda_{1,2} = \frac{1}{2}(a+c \pm d)$ and unitary eigenvectors $v_{1,2} = \begin{pmatrix} a - c \pm d \\ 2\bar{b} \end{pmatrix}$

For all cases

$$D = \hat{D} S \hat{D}^H \qquad \text{for symmetric } A$$

$$D = \hat{D} S P \hat{D}^H \qquad \text{for symmetric } A \text{ with zero diagonal}$$

for block diagonal $\hat{D}$, signature matrix S, permutation matrix P

Factorization of symmetric/Hermitian A

$$LDL^T \approx A, \;\; D \approx \hat{D} S \hat{D}^H, \;\; L_1 \approx L\hat{D}, \;\; L_2 = SL_1^H$$

$\Rightarrow$

$$\sigma_{\min}(A) \approx \sigma_{\min}(L_1 L_2) = \sigma_{\min}(L_1 L_1^T)$$

$$\|A - L_1 L_2\| \le \alpha$$

$$M \approx L_1 L_1^T \text{ and } \|M - L_1 L_1^T\| \le \beta$$

$$s \lesssim \sigma_{\min}(M) \;\; \text{and} \;\; \tilde{R}^T \tilde{R} \approx M - sI \;\; \text{and} \;\; \|M - sI - \tilde{R}^T \tilde{R}\| \le \gamma$$

$$\begin{aligned}
\sigma_{\min}(A) &\ge \sigma_{\min}(L_1 L_2) - \|A - L_1 L_2\|_2 \ge \sigma_{\min}(L_1 L_1^T) - \alpha \\
&\ge \sigma_{\min}(M) - \|L_1 L_1^T - M\|_2 - \alpha \ge \sigma_{\min}(M) - \beta - \alpha \\
&= \sigma_{\min}(sI + \tilde{R}^T \tilde{R} + M - sI - \tilde{R}^T \tilde{R}) - \beta - \alpha \\
&= \sigma_{\min}(sI + \tilde{R}^T \tilde{R}) - \gamma - \beta - \alpha \\
&\ge s - \alpha - \beta - \gamma \; .
\end{aligned}$$

1 function $[x, \delta]$ = verifySparseSym(A,b)
2 Equilibrate A
3 Compute $LDL^T(A)$
4 If D is singular, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
5 Compute approximate splitting $D \approx \hat{D} S \hat{D}^T$
6 Compute $L_1 \approx L\hat{D}$ and $L_2 = SL_1^T$
7 Compute $M \approx L_1 L_1^T$ in rounding upwards
8 Compute $\tilde{s}(M, L_1)$ and set $s := 0.9\tilde{s}$
9 Use fast estimate to compute α with $\|A - L_1 L_2\|_2 \leq \alpha$
10 If $\alpha \geq s$, improve α by $\downarrow\uparrow$
11 If $\alpha < s$, use fl.-pt. estimate to compute β with $\|M - L_1 L_1^T\|_2 \leq \beta$, else $\beta = \infty$
12 If $\alpha < s$ and $\alpha + \beta \geq s$, improve by $\downarrow\uparrow$
13 If $\alpha + \beta \geq s$, recompute M and improve α, β in ext.prec.
14 If $\alpha + \beta \geq s$, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
15 Compute $\hat{M} := M - sI$ in rounding downwards
16 Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx \hat{M}$ in rounding to nearest
17 If succeeded, goto step 20
18 Set rounding downwards and $\hat{M} = \hat{M} + (8s/10)I$; $s = s/5$;
19 Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx \hat{M}$ in rounding to nearest
20 If failed, $[x, \delta]$ = verifySparseGen(A,b), return
21 Compute γ by Theorem 2 with $\|\hat{M} - \tilde{R}^T \tilde{R}\|_2 \leq \gamma$
22 If $\alpha + \beta + \gamma \geq s$, improve γ by $\downarrow\uparrow$
23 If $\alpha + \beta + \gamma \geq s$, improve in ext. prec.
24 If $\alpha + \beta + \gamma \geq s$, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
25 $[x, \delta]$ = ErrorBound$(A, b, s - \alpha - \beta - \gamma,$ “*solve*“$)$ using LDL^T for solve

Factorization of $B := \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$

$$LDL^T \approx B, \quad D \approx \hat{D}SP\hat{D}^H, \quad L_1 \approx L\hat{D}, \quad L_2 = SPL_1^H$$

$\Rightarrow$

$$\sigma_{\min}(B) \approx \sigma_{\min}(L_1L_2) \geq \sigma_{\min}(L_1)\sigma_{\min}(L_2) = \sigma_{\min}(L_1)^2 = \sigma_{\min}(L_1L_1^T)$$

$$\|B - L_1L_2\| \leq \alpha$$

$$M \approx L_1L_1^T \text{ and } \|M - L_1L_1^T\| \leq \beta$$

$$s \lesssim \sigma_{\min}(M) \quad \text{and} \quad \tilde{R}^T\tilde{R} \approx M - sI \quad \text{and} \quad \|M - sI - \tilde{R}^T\tilde{R}\| \leq \gamma$$

$$\begin{aligned}
\sigma_{\min}(A) = \sigma_{\min}(B) &\geq \sigma_{\min}(L_1L_2) - \|B - L_1L_2\|_2 \geq \sigma_{\min}(L_1L_1^T) - \alpha \\
&\geq \sigma_{\min}(M) - \|L_1L_1^T - M\|_2 - \alpha \geq \sigma_{\min}(M) - \beta - \alpha \\
&= \sigma_{\min}(sI + \tilde{R}^T\tilde{R} + M - sI - \tilde{R}^T\tilde{R}) - \beta - \alpha \\
&= \sigma_{\min}(sI + \tilde{R}^T\tilde{R}) - \gamma - \beta - \alpha \\
&\geq s - \alpha - \beta - \gamma\ .
\end{aligned}$$

Least squares and underdetermined systems

Let $A \in \mathbb{C}^{m\times n}, b \in \mathbb{C}^m$

$$m > n: \quad \begin{pmatrix} 0 & A^H \\ A & -I_m \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad \Rightarrow \quad x = (A^H A)^{-1} A^H b = A^+ b$$

$$m < n: \quad \begin{pmatrix} -I_n & A^H \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \quad \Rightarrow \quad x = A^H y = A^H (A A^H)^{-1} b = A^+ b$$

Note that an inclusion proves the system matrix has full rank

Nearly symmetric matrices

Symmetry broken $A^T \neq A$ due to rounding errors

Nearly symmetric matrices

Symmetry broken $A^T \neq A$ due to rounding errors

Algorithm **`verifySparseSym`** is intended for symmetric matrices,

but works correctly for unsymmetric A as well !

$\delta := \|A^T - A\|$ should be small

mathematically no restriction on δ

Applies to nearly s.p.d. as well

1 function $[x, \delta]$ = verifySparseSym(A,b) $\%$ for $A^T \neq A$
2 Equilibrate A
→ 3 Compute $LDL^T(A)$
4 If D is singular, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
5 Compute approximate splitting $D \approx \hat{D} S \hat{D}^T$
6 Compute $L_1 \approx L\hat{D}$ and $L_2 = SL_1^T$
7 Compute $M \approx L_1 L_1^T$ in rounding upwards
8 Compute $\tilde{s}(M, L_1)$ and set $s := 0.9\tilde{s}$
→ 9 Use fast estimate to compute α with $\|A - L_1 L_2\|_2 \leq \alpha$
10 If $\alpha \geq s$, improve α by $\downarrow\uparrow$
11 If $\alpha < s$, use fl.-pt. estimate to compute β with $\|M - L_1 L_1^T\|_2 \leq \beta$, else $\beta = \infty$
12 If $\alpha < s$ and $\alpha + \beta \geq s$, improve by $\downarrow\uparrow$
13 If $\alpha + \beta \geq s$, recompute M and improve α, β in ext.prec.
14 If $\alpha + \beta \geq s$, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
15 Compute $\hat{M} := M - sI$ in rounding downwards
16 Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx \hat{M}$ in rounding to nearest
17 If succeeded, goto step 20
18 Set rounding downwards and $\hat{M} = \hat{M} + (8s/10)I$; $s = s/5$;
19 Compute Cholesky factor $\tilde{R}^T \tilde{R} \approx \hat{M}$ in rounding to nearest
20 If failed, $[x, \delta]$ = verifySparseGen(A,b), return
21 Compute γ by Theorem 2 with $\|\hat{M} - \tilde{R}^T \tilde{R}\|_2 \leq \gamma$
22 If $\alpha + \beta + \gamma \geq s$, improve γ by $\downarrow\uparrow$
23 If $\alpha + \beta + \gamma \geq s$, improve in ext. prec.
24 If $\alpha + \beta + \gamma \geq s$, verification failed, $[x, \delta]$ = verifySparseGen(A,b), return
→ 25 $[x, \delta]$ = ErrorBound$(A, b, s - \alpha - \beta - \gamma,$ “*solve*“$)$ using LDL^T for solve

→ First checking for least squares and underdetermined systems

```
function [xs,delta] = verifySparselss(A,b,nearsym)
% Approximate solution xs of Ax=b with rigorous error bound delta
  if nargin<3, nearsym = false; end
  [m,n] = size(A);
  if m>n                        % least squares problem
    B = [ sparse(n,n) A' ; A -speye(m) ];
    [xs,delta] = verifySparselss(B,[zeros(n,size(b,2));b],nearsym);
    xs = xs(1:n,:);
    delta = delta(1:n,:);
    return
  elseif m<n                    % underdetermined linear system
    B = [ -speye(n) A' ; A sparse(m,m) ];
    [xs,delta] = verifySparselss(B,[zeros(n,size(b,2));b],nearsym);
    xs = xs(1:n,:);
    delta = delta(1:n,:);
    return
  end

  ... ...
```

Black box algorithm for sparse linear systems (cont'd)

→ Real square matrix, real or complex right hand side(s)

```
... ...
if isreal(A)                % linear system with square matrix
  if isreal(b)              % A and b real
    xs = NaN;
    symm = isequal(A',A);
    if ( ~symm ) && ( ~nearsym )    % determine nearsym
      nearsym = all(abs(nonzeros(A'-A))<1e-15*norm(A,inf);
    end
    if symm || nearsym      % A symmetric or near symmetric
      [xs,delta] = verifySparseSPD(A,b);
    end
    if isnan(xs(1))         % SPD and Sym failed
    [xs,delta] = verifySparseGen(A,b);
  end
  else                      % A real, b complex
    [xs,delta] = verifySparselss(A,[real(b) imag(b)],nearsym);
    n = size(A,1);
    m = size(b,2);
    xs = complex(xs(:,1:m),xs(:,m+1:end));
    delta = reshape(vecnorm(reshape(delta,[],2),2,2),n,[]);
  end

... ...
```

Black box algorithm for sparse linear systems (cont'd)

→ Complex square matrix, real or complex right hand side(s)

```
... ...
     else                              % A complex, square matrix
       n = size(A,1);
       A = [real(A) -imag(A);imag(A) real(A)];
       b = [real(b);imag(b)];
       [xs,delta] = verifySparselss(A,b);
       xs = complex(xs(1:n,:),xs(n+1:end,:));
       delta = reshape(delta,n,[])';  % take care of multiple r.h.s.
       delta = reshape(vecnorm(reshape(delta,2,[]),2),size(b,2),[])';
     end
   end  % function verifySparselss
```

Comparison to Terao/Ozaki

1. set s = abs(eigs(B,1,'smallestabs')) for $B := \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$ and set $\theta := 0.5s$
2. Compute $LDL^T(B + \theta I)$
3. If the inertia of D is not $(n, 0, n)$, decrease θ and go to step 2
4. Compute ϱ with $\|B + \theta I - LDL^T\|_2 \leq \varrho$
5. if $|\theta| > \varrho$ then return $|\theta| - \varrho$ as lower bound for $\sigma_{\min}(A)$
6. If $\theta \leq \varrho$, restart from step 2 with larger $\theta > \varrho$ or verification fails

Advantage/disadvantage to Theorem 1: only one LDL^T decomposition of double-size matrix

Cons:
```
rho = NormBnd(G(p,p)-L*intval(D)*L', true);
```

Terao/Ozaki use Advanpix for extra precise bounds, therefore we do as well

Numerical results

All matrices from the Suite Sparse Matrix Collection with

$$10^3 \leq m,n \leq 10^5 \quad \text{and} \quad 10^7 \leq \kappa(A) \leq 10^{16} \quad \text{and} \quad \mathtt{nnz(A)} \leq 10^6$$

resulting in 306 test cases

Terao/Ozaki kindly provided their Matlab code

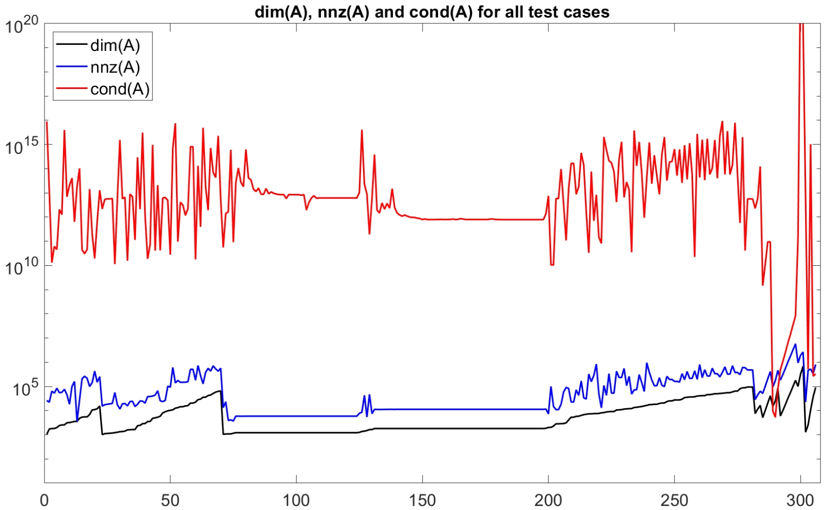


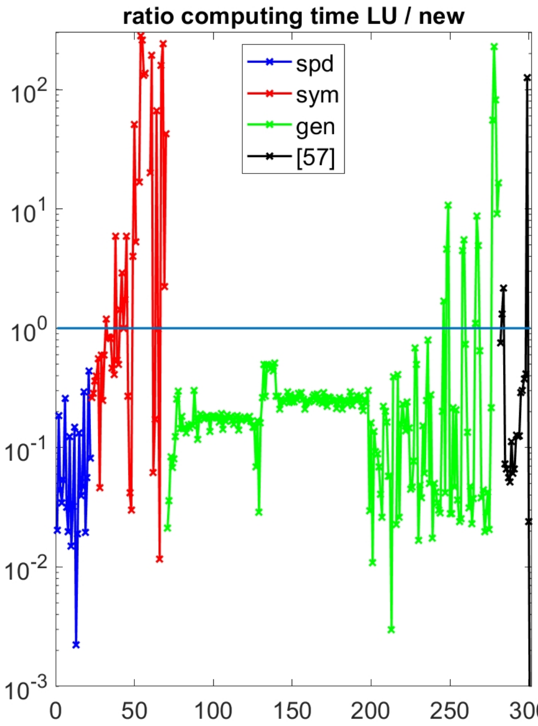
Tabelle 1: Test sets and success rate.

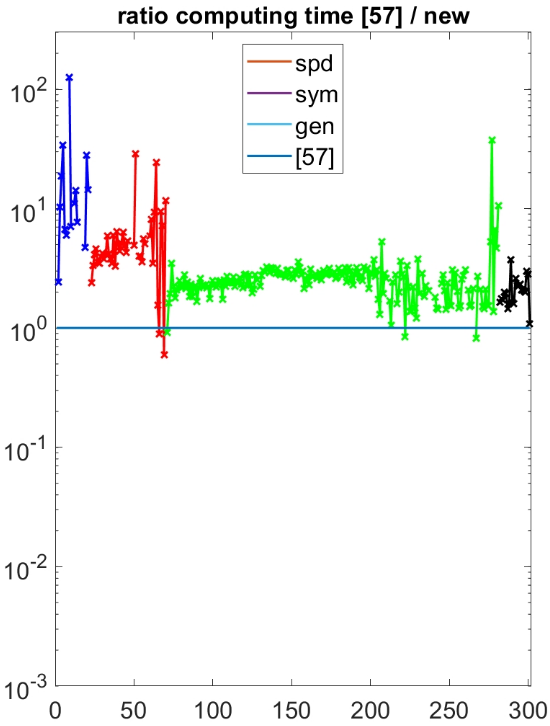
structure	success new			success Terao/Ozaki		
spd	22	out of	22	14	out of	22
sym	45	out of	48	42	out of	48
gen	210	out of	211	199	out of	211
Terao/Ozaki	20	out of	20	20	out of	20

Success new: 297 out of 301

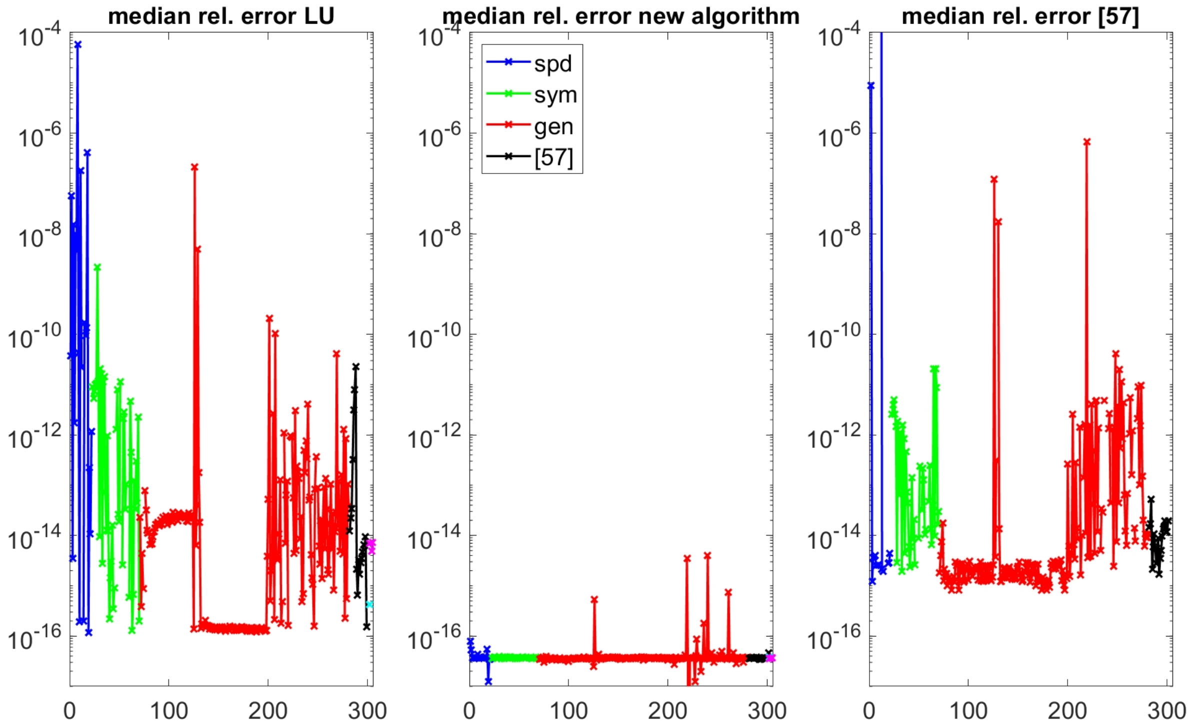
Success Terao/Ozaki: 275 out of 301

Ratio computing time lu/new and TO/new





Median relative error lu - new - Terao/Ozaki



Results for 100 randomly generated ill-conditioned test cases

```
A = sprand(n,n,dens,1/cnd)
```

with $n = 10^4$, density 0.001 and $\kappa(A) = 10^{15} \quad \Rightarrow \quad nnz(A) = 10^5$

	"new"	Terao/Ozaki
inclusions	failed in 3 out of 100 tests	failed in 33 out of 100 tests
median relative error	$3.7 \cdot 10^{-17}$	$1.6 \cdot 10^{-14}$
maximal relative error	$6.6 \cdot 10^{-11}$	1253.8
bounds containing 0 in some entries	0 out of 97 successful	26 out of 67 successful

From now on discussion of Algorithm II: inertia-based

The inertia of a 2×2 matrix computed in floating-point

Let $M := \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ be nonsingular with eigenvalues λ_1, λ_2

$$\lambda_1 + \lambda_2 = \operatorname{trace}(M) = a + c$$

$$\lambda_1 \lambda_2 = \det(M) = ac - b^2$$

$$\iota(M) \;=\; \operatorname{inertia}(M) \;=\; \#(\text{negative, zero, positive) eigenvalues}$$

$$ac - b^2 < 0 : \quad \iota(M) = (1, 0, 1)$$

$$ac - b^2 > 0 : \quad \iota(M) = \begin{cases} (0,0,2) & a + c < 0 \\ (2,0,0) & \text{otherwise} \end{cases}$$

$a, c \in \mathbb{F}: \quad \operatorname{fl}(a+c) < 0 \;\Leftrightarrow\; a + c < 0$

$p = \operatorname{fl}(ac),\ q = \operatorname{fl}(b^2): \;\; p - q \lesseqgtr 0 \;\Rightarrow\; ac \lesseqgtr b^2 \;\Leftrightarrow\; \det(M) \lesseqgtr 0$

`[p,e] = TwoProduct(a,c)` and `[q,f] = TwoProduct(b,b)` $\Rightarrow\; ac = p + e,\ b^2 = q + f$

i.e. $\operatorname{float}(\det(M)) = 0 \;\Leftrightarrow\; \operatorname{fl}(ac) = p = q = \operatorname{fl}(b^2) \;\Rightarrow\; ac - b^2 = e + f$

Accurate matrix residuals

Up to now we used extended precision of the Advanpix mp toolbox, e.g.

```
mp.Digits(34);
setround(-1); Q = double(abs(mp(A)*B - C));
setround(+1); Q = max( Q , double(abs(mp(A)*B - C) ));
```

computes `Q` with $|AB - C| \le$ `Q`

Terao/Ozaki used the Advanpix mp toolbox, so we did as well for a fair comparison

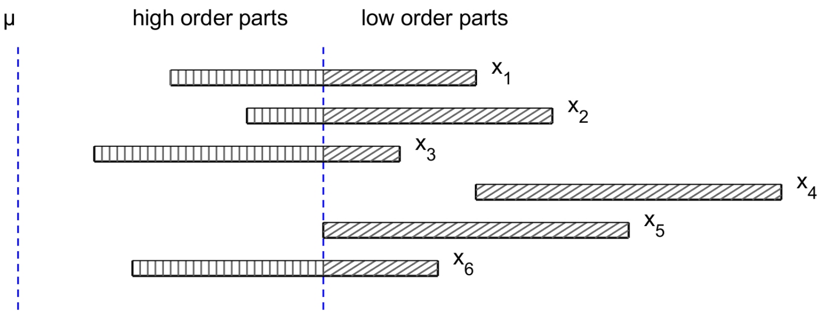
Recently we worked on an alternative

Lange, R.: Accurate floating-point matrix residuals, 49 pages, submitted

$$[\texttt{res}, \texttt{err}] = \texttt{prodK}(\texttt{A}, \texttt{B}, -1, \texttt{C}, \texttt{k});$$
$$\texttt{setround}(1)$$
$$\texttt{gamma} = \texttt{NormBnd}(\texttt{abs}(\texttt{res}) + \texttt{err}, \texttt{symm});$$

produces a result "as if" computed in $k/2 + 1$-fold precision

$\Rightarrow \quad |AB - C - \texttt{res}| \le \texttt{err} \quad$ and $\quad \|AB - C\|_2 \le \texttt{gamma} \quad$ with mathematical rigour



$$A = A^{(1)} + A^{(2)} + \ldots + A^{(k)} + \underline{A}^{(k)} \qquad \text{with} \quad \underline{A}^{(k)} := A - \sum_{i=1}^{k} A^{(i)}$$

$$AB = \sum_{i+j \le k+1} A^{(i)} B^{(j)} + \underbrace{\sum_{i=1}^{k} A^{(i)} \underline{B}^{(k+1-i)} + \underline{A}^{(k)} B}_{\text{remainder terms}} \qquad \text{error-free in floating-point}$$

K. Ozaki, T. Ogita, S. Oishi, S. Rump: Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications, Num. Alg., 59, 95–118, 2012.

Accurate matrix residuals $AB - C$ (cont'd)

Algorithm `prodK` for full input, `spProdK` for sparse

Both algorithms are implemented in Matlab by Marko Lange and fast

Recall that Advanpix is highly optimized C-code callable by Matlab

full	real data			complex data		
n	$t_{\texttt{vpa}}/t_{\texttt{mp}}$	$t_{\texttt{mp}}/t_{\texttt{prodK}}$	$t_{\texttt{prodK}}$	$t_{\texttt{vpa}}/t_{\texttt{mp}}$	$t_{\texttt{mp}}/t_{\texttt{prodK}}$	$t_{\texttt{prodK}}$
100	464	0.9	0.02	236	3.1	0.03
300	326	18.0	0.03	220	10.8	0.05
1000	306	30.1	0.21	206	62.6	0.48

sparse	real data			complex data		
n	$t_{\texttt{vpa}}/t_{\texttt{mp}}$	$t_{\texttt{mp}}/t_{\texttt{spProdK}}$	$t_{\texttt{spProdK}}$	$t_{\texttt{vpa}}/t_{\texttt{mp}}$	$t_{\texttt{mp}}/t_{\texttt{spProdK}}$	$t_{\texttt{spProdK}}$
1000	1325	0.3	0.06	1010	0.3	0.10
3000	2437	1.0	0.08	3466	0.6	0.09
10000	-	0.6	0.25	-	0.8	0.22
30000	-	1.0	0.53	-	1.0	0.49

Maximally accurate inclusions by $A^{-1}b \approx \tilde{x} + \tilde{y} + \tilde{z}$

Advanpix respects the rounding mode only for extended, not higher precision

In our `prodK` and `spProdK` higher precision is possible

$[\tilde{x}, \delta] = \text{ErrorBound3}(A, b, s, \text{“}solve\text{“})$

$\quad \tilde{x} = \text{solve}(A, b) \qquad \% \ A^{-1}b \approx \tilde{x}$

$\quad \tilde{y} = \text{solve}(A, [\![b - A\tilde{x}]\!]_{2,1}) \qquad \% \ A^{-1}b \approx \tilde{x} + \tilde{y}$

$\quad [\tilde{x}, \tilde{y}] = \text{TwoSum}(\tilde{x}, \tilde{y})$

$\quad \tilde{z} = \text{solve}(A, [\![b - A\tilde{x} - A\tilde{y}]\!]_{2,1}) \qquad \% \ A^{-1}b \approx \tilde{x} + \tilde{y} + \tilde{z}$

$\quad [\tilde{x}, \tilde{y}, \tilde{z}] = \text{spProdK}(1, \tilde{x}, 1, \tilde{y}, 1, \tilde{z}, 4) \qquad \% \ A^{-1}b \approx \tilde{x} + \tilde{y} + \tilde{z}$

$\quad \tilde{z} = \text{solve}(A, [\![b - A\tilde{x} - A\tilde{y} - A\tilde{z}]\!]_{3,1}) \qquad \% \ A^{-1}b \approx \tilde{x} + \tilde{y} + \tilde{z}$

$\quad \text{setround(-1)}; \ c_1 = \tilde{y} + \tilde{z}; \ \varrho = \text{abs}\left([\![A\tilde{x} + A\tilde{y} + A\tilde{z} - b]\!]_{3,1}\right)$

$\quad \text{setround(+1)}; \ c_2 = \tilde{y} + \tilde{z}; \ \varrho = \max\left(\varrho, \text{abs}\left([\![A\tilde{x} + A\tilde{y} + A\tilde{z} - b]\!]_{3,1}\right)\right)$

$\quad \delta = \max(|c_1|, |c_2|) + \texttt{vecnorm}(\varrho)/s$

Algorithm II based on inertia for symmetric/Hermitian matrix

Let $s \lesssim \sigma_{\min}(A)$ and

$$\|A - sI - Q_-\|_2 \le \alpha_- \quad \text{and} \quad \|A + sI - Q_+\|_2 \le \alpha_+$$

Then $\iota(Q_-) = \iota(Q_+) \;\Rightarrow\; \qquad \sigma_{\min}(A) \ge s - \max(\alpha_-, \alpha_+)$

Let $A_{-s} := A - sI$ and compute $L_{-s}D_{-s}L_{-s}^T(A_{-s})$

Split $D_{-s} \approx \hat{D}_{-s} S_- \hat{D}_{-s}^{\,T}$ as before and set $L_1 \approx LD_{-s}, L_2 = S_-L_1^T$

and compute α_- with $\|A_{-s} - L_1L_2\|_2 \le \alpha_-$

$\Rightarrow \iota(L_1L_2) = \iota(L_1S_-L_1^T) = \iota(S_-)$ for the real signature matrix S_-

accurate bounds $\|A_{\pm s} - L_1L_2\|_2 \le \alpha_{\pm s}$ follow (better than $\|A - LDL^T\|_2$)

Back
Close

Algorithm II based on inertia for symmetric/Hermitian matrix (cont'd)

If $\|A_{\pm s} - L_1 L_2\|_2 \le \alpha_{\pm s}$ not accurate enough we use LDL^T instead:

```
1  function p = residualBoundLDLT(A,L,D)
2    [C1,C2,E1] = spProdK(D,L',2);
3    [C,E2] = spProdK(L,C1,L,C2,-1,A,2);
4    alpha1 = NormBnd(abs(C)+E2,false);
5    setround(1)
6    alpha = NormBnd(L,false)*NormBnd(E1,false) + alpha1;
7  end  % function residualBoundLDLT
```

Line 2: $\left|C_1 + C_2 - DL^T\right| \le E_1$

Line 3: $\left|LC_1 + LC_2 - A - C\right| \le E_2$

$$
\begin{aligned}
\|A - LDL^T\|_2 &= \|L(DL^T - C_1 - C_2) + C + L(C_1 + C_2) - A - C\|_2 \\
&\le \|L\|_2 \|E_1\|_2 + \|\ |C| + |L(C_1 + C_2) - A - C|\ \|_2 \\
&\le \|L\|_2 \|E_1\|_2 + \|\ |C| + E_2\ \|_2 \\
&\le \|L\|_2 \|E_1\|_2 + \alpha_1 \\
&\le \alpha
\end{aligned}
$$

1 function $[x,\delta]$ = verifySparseSym(A,b)
2 Equilibrate A
3 Compute $LDL^T(A)$
4 If D is singular, verification failed, $[x,\delta]$ = verifySparseGen(A,b), return
5 Compute $\tilde{s}(A,L,D)$ and set $s := 0.9\tilde{s}$, $\Phi = true$
6 Rounding downwards, $A_s := A - sI$ and compute $L_s D_s L_s^T(A_s)$
7 Compute approximate splitting $D_s \approx \hat{D}_s S \hat{D}_s{}^T$
8 Compute $L_1 \approx L D_s$ and $L_2 = S L_1^T$
9 Compute α_- with $\|A_s - L_1 L_2\|_2 \le \alpha_-$
→10 If $\alpha_- \ge s$, improve α_- by $\downarrow\uparrow$
11 If $\alpha_- < s$, $\nu_- = \texttt{sum}(\texttt{S}) > \texttt{0}$, goto Step 13
→12 Compute α_- with $\|A_s - L_s D_s L_s^T\|_2 \le \alpha_-$, $\nu_- = \pi(D_s)$
13 If $\alpha_- \ge s$, first verification failed, go to Step 22
14 Rounding upwards, $A_s := A + sI$ and compute $L_s D_s L_s^T(A_s)$
15 Compute approximate splitting $D_s \approx \hat{D}_s S \hat{D}_s{}^T$
16 Compute $L_1 \approx L D_s$ and $L_2 = S L_1^T$
17 Compute α_+ with $\|A_s - L_1 L_2\|_2 \le \alpha_+$
→18 If $\alpha_+ \ge s$, improve α_+ by $\downarrow\uparrow$
19 If $\alpha_+ < s$, $\nu_+ = \texttt{sum}(\texttt{S}) > \texttt{0}$, goto Step 21
→20 Compute α_+ with $\|A_s - L_s D_s L_s^T\|_2 \le \alpha_+$, $\nu_+ = \pi(D_s)$
21 Set $\alpha = \max(\alpha_-, \alpha_+)$, if $\alpha < s$, go to Step 23
→22 If Φ, $\Phi = false$, $s = s/5$, goto Step 6, else $\nu_- = 0$
23 If $\nu_- \ne \nu_+$, verification failed, $[x,\delta]$ = verifySparseGen(A,b), return
24 $[x,\delta]$ = ErrorBound$(B, [0; b], s - \alpha,$ “*solve*“$)$ using LDL^T for solve

Algorithm II based on inertia for general matrix

We use the augmented matrix $B := \begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$

One shift suffices because $\iota(B) = (n, 0, n)$ for nonsingular A

As in the symmetric case we use $L_1 L_2$ rather than LDL^T

Algorithm II based on inertia for general matrix

1 function $[x, \delta]$ = verifySparseGen(A,b)
2 Equilibrate A
3 Let B the augmented matrix
4 Compute $LDL^T(B)$
5 If $nnz(D) < 2n,$ compute $LDL^T(B)$ for slightly shifted B
6 If $nnz(D) < 2n,$ verification failed, return
7 Compute $\tilde{s}(B, L, D) \lesssim \sigma_{\min}(B)$ and set $s := 0.9\tilde{s},\ \Phi = true$
8 Rounding downwards, $B_s := B - sI$ and compute $L_s D_s L_s^T(B_s)$
9 Compute approximate splitting $D_s \approx \hat{D}_s S \hat{D}_s{}^T$
10 Compute $L_1 \approx L D_s$ and $L_2 = S L_1^T$
11 Use fl.-pt. estimates to compute α with $\|B_s - L_1 L_2\|_2 \le \alpha$
12 If $\alpha \ge s,$ improve α by $\downarrow\uparrow$
13 If $\alpha \ge s,\ \nu = \texttt{sum(Ds)} > 0,$ else improve α by $\texttt{spProdK},\ \nu = \pi(D_s)$
14 If $\alpha \ge s,$ go to Step 16
15 If $\Phi,\ \Phi = false,\ s = s/5,$ goto Step 8, else $\nu = 0$
16 If $\nu \ne n,$ verification failed, return
17 $[x, \delta]$ = ErrorBound$(B, [0; b], s - \alpha,$ "$solve$"$)$ using LDL^T for solve

Back
Close

Input data with tolerances

Let $\mathbf{A} \in \mathbb{IF}^{n \times n}$ and $\mathbf{b} \in \mathbb{IF}^{n,k}$, i.e.,

$$\mathbf{A} = [\underline{A}, \overline{A}] = \{A : \underline{A} \le A \le \overline{A}\} \quad \text{for} \quad \underline{A}, \overline{A} \in \mathbb{F}^{n \times n}$$

For $A \in \mathbf{A}$ and $b \in \mathbf{b}$ and $\tilde{x} \in \mathbb{R}^n$ define $M \approx \mathrm{mid}\mathbf{A}$ and $\Delta := A - M$

$$\begin{aligned} |A^{-1}b - \tilde{x}| &= |A^{-1}(b - A\tilde{x})| = |(M + \Delta)^{-1}(b - A\tilde{x})| \\ &= |(I + M^{-1}\Delta)^{-1}M^{-1}(b - A\tilde{x})| \\ &\le \frac{\|M^{-1}(b - A\tilde{x})\|_\infty}{1 - \|M^{-1}\Delta\|_\infty}\,\mathbf{e} \\ &\le \frac{\sigma_{\min}(M)^{-1}\|\mathbf{b} - \mathbf{A}\tilde{x}\|_2}{1 - \sigma_{\min}(M)^{-1}\|\mathrm{rad}\mathbf{A}\|_2}\,\mathbf{e} \end{aligned}$$

provided that $\sigma_{\min}(M)^{-1}\|\mathrm{rad}\mathbf{A}\|_2 < 1$

Systems of nonlinear equations

$f : \mathbf{D} \to \mathbb{R}^n$ continuously differentiable with compact, convex $\mathbf{D} \subseteq \mathbb{R}^{\mathbf{n}}$

$$x, \tilde{x} \in \mathbf{D} \;\Rightarrow\; \exists \xi_1, \ldots, \xi_n \in x \underline{\cup} \tilde{x} : \; f(x) = f(\tilde{x}) + \begin{pmatrix} \nabla f_1(\xi_1) \\ \cdots \\ \nabla f_n(\xi_n) \end{pmatrix} (x - \tilde{x})$$

$\texttt{X = hull(x,xs), Y = f(gradientinit(X))} \Rightarrow$

$$\forall x \in \texttt{X} : \; f(x) \in \texttt{Y.x} \quad \text{and} \quad \nabla(f_k(x)) \in \texttt{Y.dx}_{k*}$$

$\texttt{Ys = f(intval(xs))} \quad \Rightarrow \quad f(x) \in \texttt{Ys} + \texttt{Y.dx} * (\texttt{X} - \texttt{xs})$

Back
Close

Inclusion theorem for nonlinear system

Theorem [R. 1983] Let $f : D \to \mathbb{R}^n$ with $f \in C^1$ and $\tilde{x} \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{IR}^n$, $R \in \mathbb{R}^{n \times n}$ with $0 \in \mathbf{X}$ and $\tilde{x} + \mathbf{X} \subseteq D$ be given. Suppose

$$S(\mathbf{X}, \tilde{x}) := -Rf(\tilde{x}) + \left\{ I - RJ_f(\tilde{x} + \mathbf{X}) \right\} \mathbf{X} \subseteq \operatorname{int}(\mathbf{X}) \ .$$

Then R and all matrices $M \in J_f(\tilde{x} + \mathbf{X})$ are nonsingular, and there is a unique root $\hat{x}$ of f in $\tilde{x} + S(\mathbf{X}, \tilde{x})$.

`Y = f(gradientinit(xs+X))` $\Rightarrow J_f(\tilde{x} + \mathbf{X}) \in$ `Y.dx`

Let $C :=$ `Y.dx.mid` and $\Delta :=$ `Y.dx.rad`

$\Rightarrow \quad S(\mathbf{X}, \tilde{x}) \subseteq C^{-1} \left(-f(\tilde{x}) - \Delta \cdot \mathbf{X} \right)$ sparse linear system with interval r.h.s.

Executable Matlab code for nonlinear system

Approximation part

```
function [X,kxs,kY] = verifySparseNlss(f,xs)
  setround(0)
  n = size(xs,1); phi = 1e-14*sqrt(n);
  dxs = abs(xs); kxs = 0;
  while ( kxs < 10 )             % at most 10 Newton iterations
    kxs = kxs + 1; xsold = xs;
    y = f(gradientinit(xs));    % function value and gradient
    xs = xs - y.dx\y.x;         % approximate Newton iteration
    d = abs(xs-xsold);
    if all(d<.5*abs(xs)) && ( norm(d,inf)<=phi*norm(xs,inf) )
      break
    end
  end
                        ...   ...
```

Inclusion part

```
                  ...   ...
  ys = -f(intval(xs));          % inclusion of f(xs)
  Y = mag(ys);                  % magnitude of ys
  kY = 0; setround(1)
  while ( kY < 10 )
    kY = kY + 1;
    X = 1.01*Y + realmin;       % epsilon-inflation
    JJ = f(gradientinit(midrad(xs,X)));
    J = JJ.dx;                  % inclusion of Jacobian
    b = midrad( ys.mid , ys.rad + J.rad*X );
    [Ys,delta] = verifySparselss(J.mid,b);       % sparse linear system
    Y = abs(Ys) + delta;        % r.h.s. of (7.7)
    if all( Y < X )
      X = midrad(xs,Y);         % inclusion successful
    return
  end
  X = intval(NaN(size(xs)));    % inclusion failed
end  % function verifySparseNlss
```

Numerical results

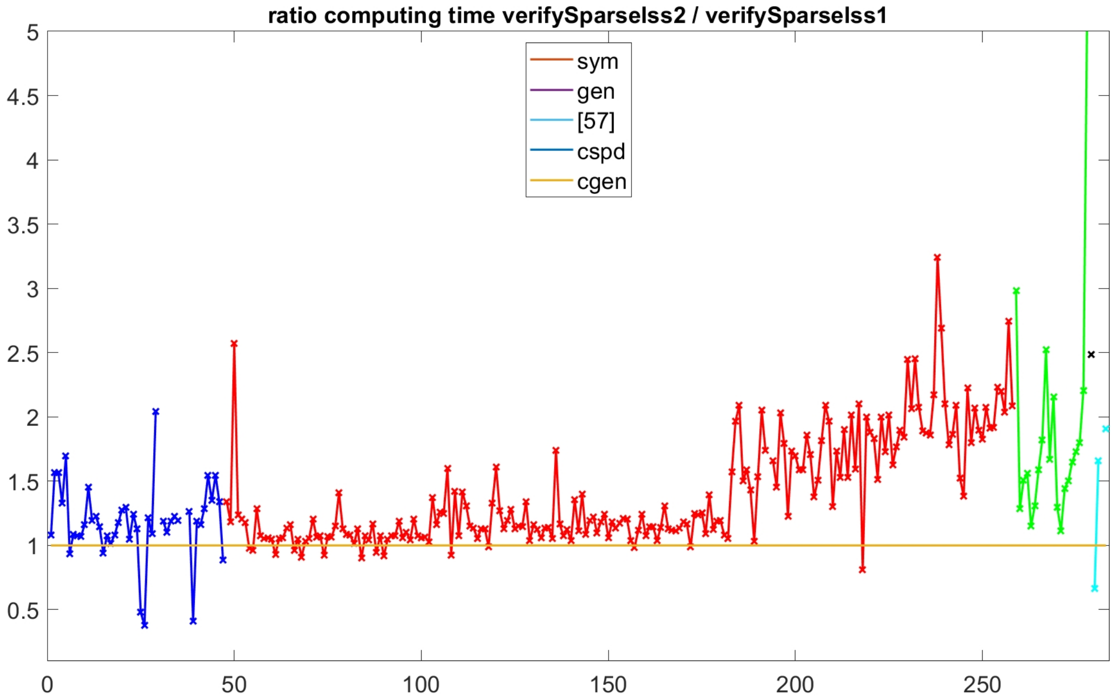
As before all matrices from the Suite Sparse Matrix Collection with

$$10^3 \le m, n \le 10^5 \quad \text{and} \quad 10^7 \le \kappa(A) \le 10^{16} \quad \text{and} \quad \texttt{nnz(A)} \le 10^6$$

	backslash				Sparse1				Sparse2			
structure	poor	fail			poor	fail			poor	fail		
real spd	2	0	of	35	0	0	of	35	0	0	of	35
sym	2	0	of	96	0	0	of	96	0	0	of	96
gen	1	0	of	249	0	2	of	249	0	0	of	249
tests TO	0	2	of	20	0	0	of	20	0	0	of	20
complex spd	1	0	of	1	0	1	of	1	0	0	of	1
gen	0	0	of	2	0	0	of	2	0	0	of	2
random	35	0	of	100	0	1	of	100	0	0	of	100
rect. orig	1	0	of	29	0	1	of	29	0	0	of	29
rect. transp	1	0	of	29	0	1	of	29	0	0	of	29
$\sum$	43	2		561	0	6		561	0	0		561

“poor” means rel. errors > 0.1

Timing first and second algorithm

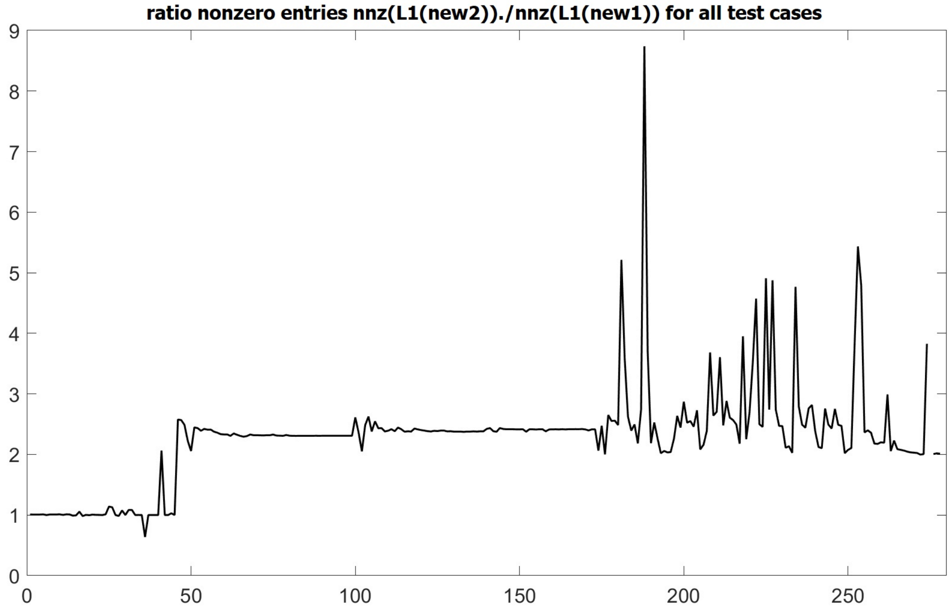


In the median `verifySparselss1` 1.22 times faster than `verifySparselss2`, at most 5.2

Conversely `verifySparselss2` up to 2.6 times faster than `verifySparselss1`

Why is `verifySparselss1` faster than `verifySparselss2` ?

$LDL^T(A \pm sI)$ may cause more fill-in than $LDL^T(A)$:



Test cases for nonlinear systems with sparse Jacobian

Function p09, p13 and p14 in

J.J. Moré, D.C. Sorensen, K.E. Hillstrom, and B.S. Garbow: The MINPACK project.
In W. Cowell ed., Sources and Development of Math. S/W, pages 88-111. Prentice Hall, 1984.

Moré/Cosnard: discretization of

$$MC: \qquad u'' = .5 * (u + t + 1)^3 \quad \text{with} \quad u(0) = u(1) = 0$$

and initial approximation $x_k = t_k(t_k - 1)$ for $t_k = k/(n + 1)$.

Abbott/Brent: discretization of

$$AB: \qquad 3y''y + (y')^2 = 0 \quad \text{with} \quad y(0) = 0 \text{ and } y(1) = 20$$

with true solution $20x^{3/4}$ and initial approximation `10*ones(n,1)`.

Matlab's nonlinear system solver `fsolve`, no derivatives

Test results for nonlinear systems with sparse Jacobian

problem	n	*cond*	$t_{Sparse1}$	$t_{Sparse2}$	$t_{\texttt{fsolve}}$	*relerr Sparse1* median	*relerr Sparse1* max	*relerr Sparse2* median	*relerr Sparse2* max	*relerr* `fsolve` median	*relerr* `fsolve` max
p09	1,000	4.0e5	0.041	0.049	0.100	1.3e-8	6.8e-6	1.3e-8	6.8e-6	0.107	0.195
	10,000	4.0e7	0.146	0.146	3.18	1.3e-5	0.068	1.3e-5	0.068	0.200	0.364
	20,000	1.6e8	0.356	0.329		1.2e-4	0.606	1.2e-4	0.604	out of memory	
	30,000	3.6e8	0.557	0.547		NaN	NaN	NaN	NaN	out of memory	
p13	1000	2.5	0.088	0.054	0.182	5.1e-13	1.3e-12	4.0e-13	1.0e-12	2.5e-13	6.3e-13
	10,000	2.5	0.206	0.166	19.6	5.2e-12	1.3e-11	4.1e-12	1.0e-11	2.9e-12	1.2e-11
	100,000	2.5	1.628	1.803		5.2e-11	1.3e-10	4.1e-11	1.0e-10	out of memory	
	1,000,000	2.5	18.21	20.06		5.2e-10	1.3e-9	4.1e-10	1.0e-9	out of memory	
	10,000,000	2.5	484.4	87.6		5.6e-9	1.4e-8	NaN	NaN	out of memory	
p14	1,000	1.8	0.725	0.284	1.458	5.0e-13	2.4e-13	5.1e-13	2.4e-13	1.2e-13	3.5e-9
	10,000	1.8	0.617	0.726	128.0	6.2e-12	1.3e-11	4.9e-12	2.3e-12	1.2e-12	2.5e-12
	100,000	1.8	11.6	31.5		6.0e-11	1.3e-10	1.3e-10	5.9e-11	out of memory	
	1,000,000	1.8	207.9	3408		6.0e-10	1.3e-9	1.3e-9	6.0e-10	out of memory	
MC	1,000	4.0e5	0.080	0.076	0.182	4.8e-12	4.9e-6	3.8e-12	3.9e-6	6.8e-4	8.3e-4
	10,000	4.0e7	0.258	0.296	15.9	0.23	2.3e-9	0.18	1.8e-9	5.3e-4	0.13
	100,000	4.0e9	3.17	26.4		2.7e-7	0.014	2.7e-7	0.014	out of memory	
	1,000,000	4.0e11	31.5	2935		NaN	NaN	NaN	NaN	out of memory	
AB	1,000	5.3e5	0.157	0.116	0.587	3.9e-13	4.5e-11	2.1e-13	2.3e-11	1.0e-9	1.1e-7
	3,000	4.7e6	0.384	0.261	5.06	1.9e-12	9.7e-10	7.4e-13	3.8e-10	4.6e-8	1.2e-5
	4,000	8.4e6	0.322	0.482	9.97	NaN	NaN	1.2e-12	4.0e-10	out of memory	
	5,000	1.3e7	0.246	0.506		NaN	NaN	NaN	NaN	out of memory	

Test results for nonlinear systems with sparse Jacobian

problem			*times [sec]*			*relerr Sparse1*		*relerr Sparse2*		*relerr* `fsolve`	
	n	*cond*	$t_{Sparse1}$	$t_{Sparse2}$	$t_{\texttt{fsolve}}$	*median*	*max*	*median*	*max*	*median*	*max*
p09	1,000	4.0e5	0.041	0.049	0.100	1.3e-8	6.8e-6	1.3e-8	6.8e-6	0.107	0.195
	10,000	4.0e7	0.146	0.146	3.18	1.3e-5	0.068	1.3e-5	0.068	0.200	0.364
	20,000	1.6e8	0.356	0.329		1.2e-4	0.606	1.2e-4	0.604	out of memory	
	30,000	3.6e8	0.557	0.547		NaN	NaN	NaN	NaN	out of memory	
	200,000	*1.6e10*	*3.63*	*3.61*		*2.5e-16*	*1.5e-13*	*2.5e-16*	*1.5e-13*	*out of memory*	
p13	1000	2.5	0.088	0.054	0.182	5.1e-13	1.3e-12	4.0e-13	1.0e-12	2.5e-13	6.3e-13
	10,000	2.5	0.206	0.166	19.6	5.2e-12	1.3e-11	4.1e-12	1.0e-11	2.9e-12	1.2e-11
	100,000	2.5	1.628	1.803		5.2e-11	1.3e-10	4.1e-11	1.0e-10	out of memory	
	1,000,000	2.5	18.21	20.06		5.2e-10	1.3e-9	4.1e-10	1.0e-9	out of memory	
	10,000,000	2.5	484.4	87.6		5.6e-9	1.4e-8	NaN	NaN	out of memory	
p14	1,000	1.8	0.725	0.284	1.458	5.0e-13	2.4e-13	5.1e-13	2.4e-13	1.2e-13	3.5e-9
	10,000	1.8	0.617	0.726	128.0	6.2e-12	1.3e-11	4.9e-12	2.3e-12	1.2e-12	2.5e-12
	100,000	1.8	11.6	31.5		6.0e-11	1.3e-10	1.3e-10	5.9e-11	out of memory	
	1,000,000	1.8	207.9	3408		6.0e-10	1.3e-9	1.3e-9	6.0e-10	out of memory	
MC	1,000	4.0e5	0.080	0.076	0.182	4.8e-12	4.9e-6	3.8e-12	3.9e-6	6.8e-4	8.3e-4
	10,000	4.0e7	0.258	0.296	15.9	0.23	2.3e-9	0.18	1.8e-9	5.3e-4	0.13
	100,000	4.0e9	3.17	26.4		2.7e-7	0.014	2.7e-7	0.014	out of memory	
	1,000,000	4.0e11	31.5	2935		NaN	NaN	NaN	NaN	out of memory	
AB	1,000	5.3e5	0.157	0.116	0.587	3.9e-13	4.5e-11	2.1e-13	2.3e-11	1.0e-9	1.1e-7
	3,000	4.7e6	0.384	0.261	5.06	1.9e-12	9.7e-10	7.4e-13	3.8e-10	4.6e-8	1.2e-5
	4,000	8.4e6	0.322	0.482	9.97	NaN	NaN	1.2e-12	4.0e-10	out of memory	
	5,000	1.3e7	0.246	0.506		NaN	NaN	NaN	NaN	out of memory	
	90,000	*4.3e9*	*3.59*	*16.1*		*2.5e-16*	*4.3e-16*	*2.5e-16*	*3.4e-16*	*out of memory*	

Data in italic: Inclusion of $f(\tilde{x})$ with extra precision

Back
Close

Benchmark problems for a sparse linear system

Problem 1: synthetic saddle-point system with random r.h.s. of norm 1

$$K = \begin{pmatrix} A & B \\ B^T & \varepsilon I_m \end{pmatrix},$$

such as Stokes flow, mixed finite-element discretizations, or KKT systems in optimization.
Velocity block $A = \nu I_n$ with small diffusion parameter $\nu = 10^{-14}$, constraint matrix $B \in \mathbb{R}^{n\times m}$ is random sparse matrix
Tikhonov regularization $\epsilon = 10^{-16}$ to avoid exact singularity.
Hence K is indefinite and ill-conditioned with eigenvalues spanning many orders of magnitude.

Problem 2: saddle-point system arising from a finite-difference discretization of 2-dim incompressible Stokes equations,

$$-\nu\Delta\mathbf{u} + \nabla p = \mathbf{f}, \qquad \nabla\cdot\mathbf{u} = 0,$$

on a rectangular, anisotropic grid resulting in the symmetric saddle-point system

$$K = \begin{pmatrix} A & 0 & D_x^T \\ 0 & A & D_y^T \\ D_x & D_y & 0 \end{pmatrix},$$

with right-hand side vector f chosen as synthetic load.

Results of Matlab's backslash, Terao/Ozaki and our algorithm

	problem 1			problem 2		
	backslash	T/O	new	backslash	T/O	new
n		75,000			60,000	
nnz(A)		125,000			357,594	
condest(A)		$8.0 \cdot 10^{16}$			$4.7 \cdot 10^{14}$	
time [sec]	0.056	2.45	2.75	> 24 hours	ϱ failed	4.5
median rel. error	0	0.72	$1.5 \cdot 10^{-16}$			$1.6 \cdot 10^{-16}$
wrong sign	104	9957	0			0
max rel. error	1	$6.5 \cdot 10^{15}$	$1.0 \cdot 10^{-8}$			$2.2 \cdot 10^{-16}$

Verification of an eigenproblem of 3-dimensional Navier-Stokes equation

Using mixed finite elements on a cubic domain

Communicated by Xuefeng Liu with 2 discretizations:

		"backslash"		inclusion	
n	nnz(A)	time [sec]	median rel. error	time [sec]	median rel. error
30,424	3,056,247	94	$1.9 \cdot 10^{-14}$	11	$4.0 \cdot 10^{-17}$
247,956	28,167,243	out of memory after 12 hours		78	$3.9 \cdot 10^{-17}$

The original matrix data was unsymmetric.

Without using "nearly symmetric" the computing time was
small: 251 instead of 11 sec
large: 311 instead of 78 sec

Papers on my homepage (google "rump tuhh")

The results in this talk allow a good upper bound for $\|A^{-1}\|_2$ for large, sparse A

Open problem: Good upper bound for $\|A^{-1}\|_\infty$ for large, sparse A

Back
Close

INTLAB: The Matlab/Octave toolbox for reliable computing

Everything presented is included in INTLAB Version 14.1

Toolboxes in INTLAB include:

- interval, affine and slope arithmetic
- gradients, Hessians and Taylor series
- ODE toolbox implementing AWA and Taylor models
- univariate and multivariate polynomials
- simulation of IEEE754 binary and base-β arithmetic in specifiable precision
- double-double and our pair arithmetic
- Galois fields GF[p]
- long: multiple precision floating-point numbers, point and interval

and many problem solving routines

An overview on INTLAB and latest developments can be found at

`https://www.tuhh.de/ti3/paper/rump/INTLAB2026final.pdf`