Enclosing interval matrix decompositions

Damien Massé

LabSTICC - Robex
Université de Bretagne Occidentale
Brest, France

Robex, dec. 2024

Goal of this work

Well-known linear algebra algorithms:

- Matrix inversion (A^{-1});
- LU decomposition ($A = LU$);
- Cholesky decomposition ($A = LL^T$);
- QR decomposition ($A = QR$);
- (others ?)

The precision and stability of these algorithms have been widely studied. Nevertheless, the design of interval guaranteed results is not so clear.

Example

Let's consider:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1.25 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 3 & 3 & 3.5 & 3 \end{pmatrix}$$

Inverting A by "full pivot" LU decomposition gives (*approximated*):

$$A^{-1} \simeq B = \begin{pmatrix} 0 & 4 & -4 & 0 \\ 7 & 1.94 \cdot 10^{-16} & -1 & -2 \\ -6 & 0 & 0 & 2 \\ 0 & -4 & 5 & 0 \end{pmatrix}$$

We know that B is "almost" an inverse of A, but how much should it be inflated to guarantee the inclusion of A^{-1}.

LU decomposition problem

The PLUQ-decomposition (as represented by Eigen) is $PLUQ$ with :

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$L \simeq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0,8 & 1 & 0 \\ 0.285 & 0.114 & 0.143 & 1 \end{pmatrix} \quad U \simeq \begin{pmatrix} 3.5 & 3 & 3 & 3 \\ 0 & 1.25 & 1 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.143 \end{pmatrix}$$

Inflate L and U, *keeping them low-triangular and up-triangular* to ensure the equality $A \in P[L][U]Q$ (P and Q cannot change as permutation matrices)?
And if we start with an interval matrix $[A]$, can we use the LU decomposition of $midpoint([A])$ to compute a decomposion $[A] \subseteq P[L][U]Q$?

General approach

To compute an operation on a matrix, or give a bound on its result, we can use:

1. a known algorithm, e.g. Gauss-Jordan algorithm;
2. infinite development, e.g. we know that, when $\|\mathrm{Id} - A\| \leq 1$:

$$(\mathrm{Id} - A)^{-1} = \sum_{k \geq 0} A^k$$

3. infinite iteration, e.g. when $\|\mathrm{Id} - A\| \leq 1$, the sequence:

$$S_0 = \mathrm{Id} \quad S_{n+1} = \mathrm{Id} + AS_n$$

(of course, these approaches are not mutually exclusive)

Using Gauss-Jordan

Direct Gauss-Jordan algorithm on intervals does not work well. Using preconditionning works better: from B approximating the inverse of A, we look for Δ such that $A^{-1} = (\mathrm{Id} - \Delta) \cdot B$.
Then, we have:

$$\Delta = \mathrm{Id} - (BA)^{-1}$$

We can then try to inverse $(BA)^{-1}$, which is (hopefully) a near-identity matrix. If $(BA)^{-1}$ is (almost) centered to Id, we may just have to bound the radius if $(BA)^{-1}$ around Id.

Gauss-Jordan algorithm (bound)

Inplace Gauss-Jordan algorithm to compute $S(M) = (\mathrm{Id} - M)^{-1} - \mathrm{Id}$ (without row-pivoting).

```
procedure GaussJordanBound(M: (interval) Matrix)
    for all ind : indices do
        pivot ← 1/(1-M(ind,ind))                          ▷ Suppose M(ind,ind)<1
        for all row : indices≠ind, col : indices≠ind do
            M(row,col) ← M(row,col) + M(row,ind)*pivot*M(ind,col)
        end for
        for all ind2 : indices≠ind do
            M(ind,ind2) ← M(ind,ind2)*pivot
            M(ind2,ind) ← M(ind2,ind)*pivot
        end for
        M(ind,ind) ← M(ind,ind)*pivot
    end for
end procedure
```

When M is non-negative and $\|M\| < 1$, the algorithm succeeds and all operations are monotonic (assuming $x \mapsto 1/(1-x)$ is atomic). We can compute a safe bound with floating-point computations (rounding up) on the matrix of magnitudes.

Example

Let's consider:

$$[A] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 \\ 1.25 & 0 & 0 & [-1, -0.9] \\ 3 & 3 & [3.3, 3.5] & 3 \end{pmatrix}$$

Using the bound version of Gauss-Jordan algorithm:

$$[A^{-1}] = \begin{pmatrix} 0 & [-4, -2.34] & [2.67, 4] & 0 \\ [6, 11] & [5.66, 9] & [-8, -5.33] & [-3.33, -1.67] \\ [-10, -5] & [-\varepsilon, \varepsilon] & [-\varepsilon, \varepsilon] & [1.67, 3.33] \\ 0 & [-5, 3.33] & [2.66, 4] & 0 \end{pmatrix}$$

Discussion

+ Easy to apply, at least for some decompositions (inversion, LU)
+ Quite fast (cubic complexity, no successive iterations),
– Result dependent on the order of traversal of the indices
– Probable accumulation of errors on big matrices

Norm-based bound

The norm-based bound is used to directly enclose:

$$S(M) = \sum_{i\ \geq 1} M^i = (\mathrm{Id} - M)^{-1} - \mathrm{Id}$$

Using a multiplicative matrix norm:

$$\|A + B\| \leq \|A\| + \|B\| \qquad \|AB\| \leq \|A\| \cdot \|B\|$$

then:

$$\|M\| < 1 \Rightarrow \|S(M)\| \leq \frac{\|M\|}{1 - \|M\|}$$

If $\|M\| \geq \max_{ij} |M_{ij}|$, we can bound $S(M)$ inside the box $\|S(M)\| * E$ where E is the interval matrix in which all coefficients are [-1,1].

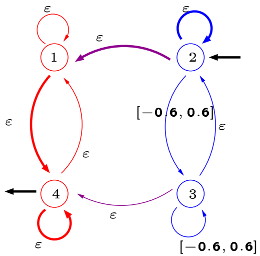
Discussion

+ Can be applied on most problems
+ Fastest, independant of order of traversal
+ No accumulation of errors for big matrices
– Not precise, mostly useful on punctual matrices
– Does not handle 0 values well on block matrices

Path-based algorithm

"Intermediate" algorithm: consider $S(M) = \sum_{i \geq 1} M^i$. The coefficients of $S(M)$ can be linked to paths on the graph associated to the matrix M.
E.g. let's consider $[M]$:

$$[M] \simeq \begin{pmatrix} \varepsilon & 0 & 0 & \varepsilon \\ \varepsilon & \varepsilon & [-0.6, 0.6] & \varepsilon \\ \varepsilon & \varepsilon & [-0.6, 0.6] & \varepsilon \\ \varepsilon & 0 & 0 & \varepsilon \end{pmatrix}$$



Example of path from 2 to 4 of length 4:
$2 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 4$.
This path "defines" the term $M_{2,2}M_{2,1}M_{1,4}M_{4,4}$ which is a "part" of $(M^4)_{2,4}$.

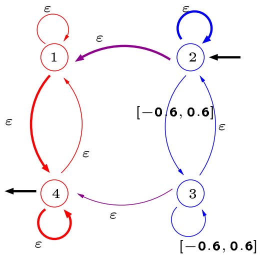
Paths

Then:

$$(S(M))_{i,j} = \sum_{\ell \geq 1} \left(\sum_{\pi \in \Pi_{ij}^{\ell}} \prod_{k=0}^{l-1} M_{\pi_k, \pi_{k+1}} \right)$$

where Π_{ij}^{ℓ} is the set of paths from i to j of length ℓ (each path π being of the form):

$$\pi_0 = i \to \pi_1 \to \ldots \to \pi_\ell = j$$



Example of path from 2 to 4 of length 4. This path defines the term $M_{2,2} M_{2,1} M_{1,4} M_{4,4}$. The sum of these terms for all paths of length 4 from 2 to 4 is $(M^4)_{2,4}$.

Path-based bounding algorithm

Possible algorithm to bound $S(M)$ (with $M \geq 0$) based on this.

1. For each i, j, compute the maximum term $M_{u,v}$ appearing in all paths from i to j (hence a product of length l is bounded by $(M_{u,v})^l$). This is a (max,max) Floyd algorithm (cubic complexity).
2. Bound the number of paths, for example by bounding the number $k(i,j)$ of indices which can appear from i to j (easly obtained from the Floyd algorithm).
3. Then:

$$(S(M))_{i,j} \leq \frac{M_{u,v}}{1 - k(i,j)M_{u,v}}$$

Comparison

Let's consider:

$$[A] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 \\ 1.25 & 0 & 0 & [-1, -0.9] \\ 3 & 3 & [3.3, 3.5] & 3 \end{pmatrix}$$

norm-based bounding

$$\begin{pmatrix} [-8, 8] & [-10.5, 4.17] & [-3.33, 10] & [-2.5, 2.5] \\ [0.5, 16.5] & [-5.15, 14.67] & [-13.33, \varepsilon] & [-5, \varepsilon] \\ [-15.5, 0.5] & [-7.33, 7.33] & [-6.67, 6.67] & [-\varepsilon, 5] \\ [-8, 8] & [-11.5, 3.17] & [-3.33, 10] & [-2.5, 2.5] \end{pmatrix}$$

Floyd-based bounding (refined)

$$\begin{pmatrix} 0 & [-4.17, 2.17] & [2.5, 4.17] & 0 \\ [4.63, 12.38] & [0.75, 13.92] & [-12.5, -0.83] & [-3.75, -1.25] \\ [-11.375, -3.62] & [-4.59, 4.59] & [-4.17, 4.17] & [1.25, 3.75] \\ 0 & [-5.17, 3.17] & [2.5, 4.17] & 0 \end{pmatrix}$$

Gauss-Jordan bounding

$$\begin{pmatrix} 0 & [-4, -2.34] & [2.67, 4] & 0 \\ [6, 11] & [5.66, 9] & [-8, -5.33] & [-3.33, -1.67] \\ [-10, -5] & [-\varepsilon, \varepsilon] & [-\varepsilon, \varepsilon] & [1.67, 3.33] \\ 0 & [-5, 3.33] & [2.66, 4] & 0 \end{pmatrix}$$

Discussion

- + A bit faster than Gauss-Jordan
- + No accumulation of errors for big matrices
- + Handles 0 values well on block matrices
- – Slower than norm-based bounds
- – Precision depends on the form of the matrix (sometimes worse than the norm).

LU decomposition

Full-pivot LU decomposition decomposes a matrix M into:

$$M = P^{-1} L U Q^{-1}$$

where P and Q are permutation matrices, L is lower-triangular and U is upper-triangular. In Eigen decomposition, L is unit-lower-triangular and the diagonal coefficients of U are sorted with the 0 at the end.
Not unique and not "stable": given a existing decomposition of M, P and Q may not be kept for all matrices "close" to M.

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \mathrm{Id}_2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} (0)\, \mathrm{Id}_1$$

$$\begin{pmatrix} 0 \\ \varepsilon \end{pmatrix} \neq \mathrm{Id}_2 \begin{pmatrix} 1 \\ \alpha \end{pmatrix} (\beta)\, \mathrm{Id}_1$$

Approximation

Let's consider M and (P, L, U, Q) an approximate decomposition:

$$M \simeq P^{-1} L U Q^{-1} \qquad PMQ = LU$$

To handle rectangular matrices, we extend PMQ with a Id-block, and adapt L and U adequately.

$$\begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} (2) \;\Rightarrow\; \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.5 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

We look for a slight modification of L and U to enclose a "correct" decomposition of M:

$$PMQ = L(\mathrm{Id} - \Delta L)(\mathrm{Id} - \Delta U)U$$

(ΔL is stricly lower, ΔU is upper)

Triangular inversion

From

$$PMQ = L(\mathrm{Id} - \Delta L)(\mathrm{Id} - \Delta U)U$$

to

$$[L^{-1}]PMQ[U^{-1}] = (\mathrm{Id} - \Delta L)(\mathrm{Id} - \Delta U)$$

But while L is non-singular, U can be singular. $\Rightarrow$ we modify its diagonal to make it non-singular (replacing too small coefficients on the diagonal by a threshold e).

Note that L and U are punctual, we can expect $[L^{-1}]$ and $[U^{-1}]$ to be quasi-punctual (e.g. computed by substitution). The result should be close to Id (for punctual, non-singular matrices).

Example

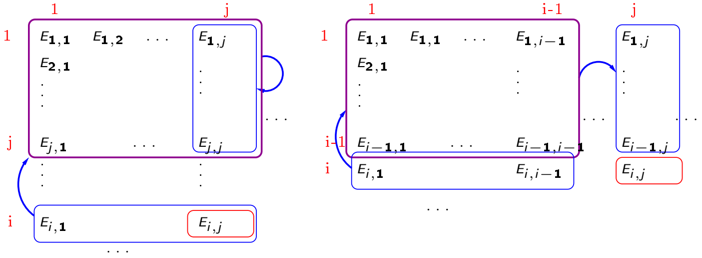
$$[M] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 \\ 1.25 & 0 & 0 & [-1, -0.9] \\ 3 & 3 & [3.3, 3.5] & 3 \end{pmatrix}$$

$$\mathrm{Id} - [L^{-1}]PMQ[U^{-1}] \simeq \begin{pmatrix} [-0.03, 0.03] & [-0.07, 0.07] & [-0.64, 0.64] & [-0.75, 0.75] \\ 0 & 0 & [-0.21, 0.21] & 0 \\ 0 & [0, \varepsilon] & [-0.17, 0.17] & 0 \\ [-0.01, 0.01] & [-0.02, 0.02] & [-0.35, 0.35] & [-0.22, 0.22] \end{pmatrix}$$

With $\mathrm{E} = \mathrm{Id} - [L^{-1}]PMQ[U^{-1}]$, We want to solve:

$$(\mathrm{Id} - \Delta L)(\mathrm{Id} - \Delta U) = \mathrm{Id} - \mathrm{E}$$

Exact value



$i > j$ (lower) $i \leq j$ (upper)

Lower coefficients ($i > j$):

$$(\Delta L)_{i,j} = E_{i,j} + E_{i,[1,j]} \left(\sum_{\ell \geq 0} (E_{[1,j],[1,j]})^{\ell} \right) E_{[1,j],j}$$

Upper coefficients ($2 \leq i \leq j$):

$$(\Delta U)_{i,j} = E_{i,j} + E_{i,[1,i-1]} \left(\sum_{\ell \geq 0} (E_{[1,i-1],[1,i-1]})^{\ell} \right) E_{[1,i-1],j}$$

Application

To enclose the inverse, we needed to bracket $S(E) = (\mathrm{Id} - E)^{-1} = \sum_{\ell \geq 1} E^{\ell}$. Here, we need to enclose $S_k = S(E_{[1,k],[1,k]})$ with k varying from 1 to $N - 1 \longrightarrow$ same techniques.

- Both Gauss-Jordan and Floyd-Based algorithms compute bounds for successive S_k incrementally;
- In the case of ∞ coefficients, we should consider $0 * \infty = 0$. It works well for Id-blocks on the right/bottom of the decomposition.
- No need to compute $S(E)$ itself (the last step).

Example

$$E \simeq \begin{pmatrix} [-0.03, 0.03] & [-0.07, 0.07] & [-0.64, 0.64] & [-0.75, 0.75] \\ 0 & 0 & [-0.21, 0.21] & 0 \\ 0 & [0, \varepsilon] & [-0.17, 0.17] & 0 \\ [-0.087, 0.087] & [-0.020, 0.020] & [-0.35, 0.35] & [-0.22, 0.22] \end{pmatrix}$$

With Gauss-Jordan-based bounds:

$$\Delta L \simeq \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & [0, \varepsilon] & 0 & 0 \\ [-0.089, 0.089] & [-0.021, 0.021] & [-0.44, 0.44] & 0 \end{pmatrix}$$

$$\Delta U \simeq \begin{pmatrix} [-0.03, 0.03] & [-0.07, 0.07] & [-0.64, 0.64] & [-0.75, 0.75] \\ 0 & 0 & [-0.21, 0.21] & 0 \\ 0 & 0 & [-0.17, 0.17] & 0 \\ 0 & 0 & 0 & [-0.23, 0.23] \end{pmatrix}$$

Example (cont'd)

$$[M] = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & -1 \\ 1.25 & 0 & 0 & [-1, -0.9] \\ 3 & 3 & [3.3, 3.5] & 3 \end{pmatrix}$$

$$P^{-1}[L][U]Q^{-1} \simeq \begin{pmatrix} [0.89, 1.11] & [0.89, 1.11] & [0.94, 1.06] & [0.74, 1.30] \\ 1 & 0 & 0 & [-1.08, -0.91] \\ 1.25 & 0 & 0 & [-1, -0.9] \\ [2.82, 3.18] & [2.82, 3.18] & [3.3, 3.5] & [2.69, 3.31] \end{pmatrix}$$

Note: Eigen may not select the "best" pivot w.r.t. to the uncertainties (it used 4th line, 3rd column as the first pivot), but it knows only the midpoint. Implanting a different strategy (based on the absolute value of the midpoint and the radius of the interval) may be possible but not easy.

Cholesky decomposition (LL^T)

Cholesky decomposition decompose a *symmetric positive matrix* M in the form of :

$$M = LL^T$$

where L is lower-triangular.
A similar approach to the LU can be considered, in which we would have to solve:

$$(\mathrm{Id} - \Delta L)(\mathrm{Id} - \Delta L^T) = (\mathrm{Id} - \mathrm{Err})$$

However the diagonal “shared” by ΔL and ΔL^T modify slightly the computation.

Cholesky decomposition (LDL^T)

Another Cholesky decomposition (which avoids square root computations) is of the form:

$$M = LDL^T$$

where L is unit-lower-triangular, and D is diagonal.
We consider an approximative decomposition $M \simeq LD_0L^T$. Our goal is to enclose ΔL (stricly lower triangular) and D such that:

$$M = L(\mathrm{Id} - \Delta L)D(\mathrm{Id} - \Delta L^T)L^T$$

(D is not built from D_0, in fact D_0 will not be used).

Problem formulation

From:

$$M = L(\mathrm{Id} - \Delta L)D(\mathrm{Id} - \Delta L^T)L^T$$

we get:

$$[L^{-1}]M[(L^T)^{-1}] = (\mathrm{Id} - \Delta L)D(\mathrm{Id} - \Delta L^T)$$

We can expect $A = [L^{-1}]M[(L^T)^{-1}]$ to be "almost diagonal". With V the diagonal of A, we pose:

$$A = V(\mathrm{Id} - \ E)$$

where the diagonal of E is 0 (note that E is *not* symmetric).

Result

$$(\mathrm{Id} - \Delta L)D(\mathrm{Id} - \Delta L^T) = A = V(\mathrm{Id} - \ E)$$

Diagonal coefficients:

$$D_j = V_j \left(E_{j,[1,j-1]} \left(\sum_{\ell \geq 0} (E_{[1,j-1],[1,j-1]})^\ell \right) E_{[1,j-1],j} \right)$$

Lower coefficients ($i > j$):

$$(\Delta L)_{i,j} = V_j^{-1}(A_{i,j} + A_{i,[1,j]} \left(\sum_{\ell \geq 0} (E_{[1,j],[1,j]})^\ell \right) E_{[1,j],j})$$

"Upper" coefficients (transpose) ($i > j$):

$$(\Delta L)_{i,j} = (\Delta L)^T_{j,i} = E_{j,i} + E_{j,[1,j]} \left(\sum_{\ell \geq 0} (E_{[1,j],[1,j]})^\ell \right) E_{[1,j],i}$$

Example

$$[M] = \begin{pmatrix} 4 & 1 & 1 & -1 \\ 1 & [4, 4.5] & 0 & 0 \\ 1 & 0 & 3 & [-0.2, 0] \\ -1 & 0 & [-0.2, 0] & [2, 3] \end{pmatrix}$$

$$P^{-1}[L][D][L^T]P \simeq \begin{pmatrix} [3.94, 4.06] & [0.88, 1.13] & [0.98, 1.02] & [-1.02, -0.98] \\ [0.87, 1.13] & [4, 4.5] & [-0.04, 0.04] & [-0.04, 0.04] \\ [0.98, 1.02] & [-0.04, 0.04] & [2.99, 3.01] & [-0.21, 0.01] \\ [-1.02, -0.98] & [-0.04, 0.04] & [-0.21, 0.01] & [1.96, 3.04] \end{pmatrix}$$

Application to orthogonal matrices

An orthogonal matrix $Q \in \mathrm{SO}_n(\mathbb{R})$ satisfies $QQ^T = \mathrm{Id}$. "Almost orthogonal" matrix would be $QQ^T \simeq \mathrm{Id}$.
If a (non-singular) matrix A satisfies $QQ^T = AA^T$ then $A^{-1}Q \in \mathrm{SO}_n(\mathbb{R})$ (with a Cholesky decomposition $QQ^T = LDL^T$, $D^{-0.5}L^{-1}Q \in \mathrm{SO}_n(\mathbb{R})$).

We can use this approach to enclose a orthogonal matrix "around" Q (e.g. when Q is the "quasi-orthogonal" result of a floating-point computation).

Example

We consider:

$$Q = \begin{pmatrix} 0.8 & -0.3 & -0.1 \\ 0.3 & 0.6 & -0.6 \\ -0.2 & -0.4 & -0.7 \end{pmatrix}$$

We use here only the enclosing algorithm developed for decomposing $\mathrm{Id} - E$ on QQ^T (i.e. no preliminary Cholesky decomposition). The result is:

$$[Q'] \simeq \begin{pmatrix} 0.93 & -0.35 & -0.12 \\ 0.19 & 0.73 & -0.66 \\ [-0.32, -0.31] & [-0.59, -0.58] & [-0.75, -0.74] \end{pmatrix}$$

This approach does not find the “closest” orthogonal matrix, but can be used to “terminate” approximate methods.

More general approach

Instead of trying to solve $(\mathrm{Id} - \Delta)(\mathrm{Id} - \Delta^T) = \mathrm{Id} - E$, we can decompose Δ as a sum of infinite terms:

$$(\mathrm{Id} - \Delta_1 - \Delta_2 \ldots)(\mathrm{Id} - \Delta_1^T - \Delta_2^T \ldots) = \mathrm{Id} - E,$$

each term being of decreasing order.
Then, we have:

$$\begin{array}{rcl} \Delta_1 + \Delta_1^T & = & E \\ \forall n \geq 2,\ \Delta_n + \Delta_n^T & = & \displaystyle\sum_{i=1}^{n-1} \Delta_i \Delta_{n-i}^T \end{array}$$

+ we can select whatever we want to decompose E to $\Delta_1 + \Delta_1^T$ (e.g. $\Delta_1 = \Delta_1^T$, since E is symmetric)
− exact expression of Δ_n is harder, bounding relies more on norms.

Norm-based bounding

Looking for symmetric Δ_i (i.e. matrix square root):

$$\begin{aligned} \Delta_1 = \Delta_1^T &= \frac{E}{2} \\ \forall n \geq 2,\ \Delta_n = \Delta_n^T &= \frac{\sum_{i=1}^{n-1} \Delta_i \Delta_{n-i}}{2} \end{aligned}$$

To bound $\|\Delta_i\|$, we look for a formula of the form:

$$\Delta_i = \Phi(i) \frac{E^i}{2^{2i-1}}$$

We can achieve that if $\Phi(n)$ satisfies for $n \geq 2$:

$$\Phi(n) = \sum_{i=1}^{n-1} \Phi(i) \Phi(n-i)$$

Norm-based bounding

With $\Phi(1) = 1$ and

$$\forall n \geq 2, \Phi(n) = \sum_{i=1}^{n-1} \Phi(i)\Phi(n-i)$$

$\Phi_n = C_{n-1}$ where $C_n = \frac{(2n)!}{(n+1)!n!}$ are the *Catalan numbers*.

To conclude (matrix square root development):

$$\Delta_i \leq C_{i-1} \frac{E^i}{2^{2i-1}}$$

and we want to bound $\sum_{i \geq N} \|\Delta_i\|$.

Bounding (cont'd)

Theorem

$$\text{if } \rho \leq 1/4, \text{ then } \sum_{i\geq 0} \rho^i C_i = \frac{1}{2\rho}(1 - \sqrt{1-4\rho})$$

Hence, with $\|E\| \leq 1$*:*

$$\sum_{i\geq 0} \|\Delta_i\| \leq 1 - \sqrt{1 - \|E\|}$$

Two bounds for the residual $\sum_{i\geq N} \|\Delta_i\|$:

- the "exact" one (with $\|E\| \leq 1$) (hard to compute):

$$\sum_{i\geq N} \|\Delta_i\| \leq 1 - \sqrt{1 - \|E\|} - \sum_{i=1}^{N-1} \frac{C_{i-1}}{2^{2i-1}} \|E\|^i$$

- the "approximate" one (with $\|E\| < 1$):

$$\sum_{i\geq N} \|\Delta_i\| \leq \frac{C_{N-1}\|E\|^N}{2^{2N-1}(1 - \|E\|)} \leq \frac{\|E\|^N}{2(1 - \|E\|)}$$

QR decomposition

QR decomposition decomposes a rectangular matrix A into $A = QR$ where Q is orthogonal and R is upper-triangular (with pivoting, the decomposition is $AP = QR$ where P is a permutation matrix).
From an existing approximate decomposition $A \simeq QR$:

- if the goal is to find $A \in [Q'][R']$ with $Q' \in \mathrm{SO}_n(\mathbb{R})$ and $[R']$ is *almost* upper-triangular, just compute $[Q']$ from Q, then $[R'] = [Q']^T A$.
- To get a "real" QR decomposition, we can use an "infinite"-sum approach with a norm-based bound.

Equations

From an approximate decompositon $A \simeq QR$, we look for an equality of the form:

$$A = Q(\mathrm{Id} - \Delta Q_1 - \Delta Q_2 - \ldots) \ldots (\mathrm{Id} - \Delta R_1 - \Delta R_2 - \ldots)R$$

where the ΔR_i are upper-triangular, and $Q(\mathrm{Id} - \Delta Q_1 - \Delta Q_2 - \ldots)$ is orthogonal.
I.e.

$$(\mathrm{Id} - \Delta Q_1 - \Delta Q_2 - \ldots) \ldots (\mathrm{Id} - \Delta R_1 - \Delta R_2 - \ldots) = Q^{-1} A R^{-1}$$

$$(\mathrm{Id} - \Delta Q_1 - \Delta Q_2 - \ldots)(\mathrm{Id} - \Delta Q_1^T - \Delta Q_2^T - \ldots) = (Q^T Q)^{-1}$$

Note: case A not square or R singular should be more formalised.

Let's pose $E_r = \mathrm{Id} - Q^{-1} A R^{-1}$ and $E_q = \mathrm{Id} - (Q^T Q)^{-1}$ (note that E_q is symmetric).

Systems

Then we have the following systems:

$$\begin{cases} \Delta Q_1 + \Delta Q_1^T = E_q \\ \Delta Q_1 + \Delta R_1 = E_r \end{cases}$$

$$\begin{cases} \Delta Q_n + \Delta Q_n^T = \sum_{k=1}^{n-1} \Delta Q_k \Delta Q_{n-k}^T & \text{(symmetric)} \\ \Delta Q_n + \Delta R_n = \sum_{k=1}^{n-1} \Delta Q_k \Delta R_{n-k} \end{cases}$$

This systems enable to compute Q_i and R_i, e.g. for $i = 1$:

- we have $\Delta Q_1 = E_r$ for the lower triangular part
- using $\Delta Q_1 + \Delta Q_1^T = E_q$ we deduce the upper triangular part and diagonal
- back to $\Delta Q_1 + \Delta R_1 = E_r$ we compute R_1.

If $\mathcal{N}$ is the maximum of $\|E_q\|_\alpha, \|E_r\|_\alpha$ with $\alpha \in \{1, \infty\}$:

$$\text{for all } \alpha \in \{1, \infty\}, \|Q_1\|_\alpha \leq 3\mathcal{N} \qquad \|R_1\|_\alpha \leq 3\mathcal{N}$$

Error bounds

Extending the result to successive products, we get:

$$\sum_{i \geq N} \|\Delta Q_i\| \leq \frac{(3\mathcal{N})^N}{2(1-3\mathcal{N})}$$

$$\sum_{i \geq N} \|\Delta R_i\| \leq \frac{(3\mathcal{N})^N}{2(1-3\mathcal{N})}$$

The bound is quite rough, but can be used for punctual decompositions.

$$[A] = \begin{pmatrix} [0.2, 0.21] & -2.3 & -0.1 & -0.6 \\ 1.3 & 0.6 & -0.6 & -1.2 \\ -0.2 & -1.4 & -0.7 & 1.3 \\ -0.2 & -1.5 & -0.2 & 1.4 \end{pmatrix}$$

$$[Q][R]P^{-1} \simeq \begin{pmatrix} [0.17, 0.23] & [-2.32, -2.27] & [-0.16, -0.03] & [-0.65, -0.54] \\ [1.28, 1.32] & [0.58, 0.62] & [-0.65, -0.55] & [-1.24, -1.16] \\ [-0.22, -0.18] & [-1.42, -1.38] & [-0.75, -0.65] & [1.26, 1.34] \\ [-0.22, -0.18] & [-1.52, -1.48] & [-0.25, -0.15] & [1.36, 1.44] \end{pmatrix}$$

Future work

- Implementation into Codac.
- More tests, larger and complex matrices.
- Comparison with other tools (Intlab).
- Better boundings (e.g. for QR decomposition).
- Specific applications, on the contraction of matrix product, linear programming, etc.
- Other computations.