ALGORITHMS FOR SOLVING (BANDED PLUS) SEMISEPARABLE MATRICES LINEAR SYSTEMS AND ITS APPLICATION IN NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

Zhiwei Zhou,[1]

[1] **Department of Mathematics, Faculty of Natural Sciences, Imperial College London.**

INTRODUCTION

The properties of semiseparable matrices and related computations are of increasing interest due to their application to the numerical solution of partial differential equations (PDEs), ref. Vandebril, Barel, et al., 2005. To see this, following the results in the paper, Vandebril, Van Barel, and Mastronardi, 2005, which gives the definition of a symmetric semiseparable matrices of semiseparability rank 1, generate by $u, v \in \mathbb{R}^n$, in form $S = triu(uv^T) + tril(vu^T, 1)$, and proves that the inverse of any tridiagonal matrix with non-zero upper and lower bands is a semiseparable matrix form as the previously definition. Combining with LU decomposition, ref. Golub and Van Loan, 2013, and the following shows use on finite difference matrix decomposition.

$U^{-1} = TRIU(XY^T)$

For (block) bidiagonal U with main (block) diagonal (D) d and super diagonal (B) b, for bidiagonal matrix $U \in \mathbb{R}^{n \times n}$, $x, y \in \mathbb{R}^n$,

$$x := U^{-1} e_n;$$
$$y_i := (d_i x_i)^{-1}, \text{ for } i = 1, 2, \ldots, n$$

and for block bidiagonal matrix, $U \in \mathbb{R}^{(ku+r)\times(ku+r)}$, $X \in \mathbb{R}^{(ku+r)\times u}$,

$$X := U^{-1} E_k;$$
$$Y_i^T := (D_i X_i)^{-1} \text{ for } i = 1, 2, \ldots, k; \qquad Y_0^T := (D_0 X_0)^+$$

ALGORITHM

Input : An upper banded matrix $U \in \mathbb{R}^{n\times n}$ and its bandwidth u.
Output: Its inverse U^{-1} and $X, Y \in \mathbb{R}^{n\times u}$ such that $U^{-1} = \text{triu}(XY^{\text{T}})$.

```
// n = ku + r where 0 <= r < u
k ← quotient of n divided by u ;                                  // floor(n/u)
r ← remainder of n divided by u ;                                  // rem(n,u)
E_k ← last block column of I_{n×n} ;                     // one(U)[:, n-u+1:n]
X ← BackSubstitution(U, E_k)
for i ← k to 1 do
    D_i ← i-th diagonal block of U ;   // U[(i-1)*u+r+1:i*u+r, (i-1)*u+r+1:i*u+r]
    X_i ← i-th block of X ;                              // X[(i-1)*u+r+1:i*u+r, :]
    Y_i^T ← (D_i X_i)^{-1}, i-th block of Y^T ;          // Yt[:,(i-1)*u+r+1:i*u+r]
end
if r = 0 then return U^{-1} := triu(XY^T), X, Y
else need find Ỹ_0^T using Moore-Penrose right inverse
    D̃_0 ∈ R^{r×r} ← first diagonal block of U ;                         // U[1:r, 1:r]
    X̃_0 ∈ R^{r×u} ← first block of X ;                                    // X[1:r, :]
    Ỹ_0^T ∈ R^{u×r} ← (D̃_0 X̃_0)^+ = (D̃_0 X̃_0)^T (D̃_0 X̃_0 (D̃_0 X̃_0)^T)^{-1}, first block of Y^T
end
return U^{-1} := triu(XY^T), X, Y
```

Algorithm 1: Inverse of banded matrix in outer product form

APPLICATION

For finite difference method for 1D (and 2D) BVPs, ref. Saad, 2003,

$$-\Delta u(x) = f(x) \text{ for } x \in (0, L),$$
$$u(0) = u(L) = 0.$$

Divide $[0, L]$ evenly into $n + 1$ points, $x_i = i \times h$, $h = L/(n + 1)$ and applying the center difference approximation, $u(x_i) \approx u_i$. With the known $u(x_0) = u_0$ and $u(x_{n+1}) = u_{n+1}$,

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f(x_i) = h^2 f_i$$
$$\Rightarrow T\boldsymbol{x} = \boldsymbol{f}$$

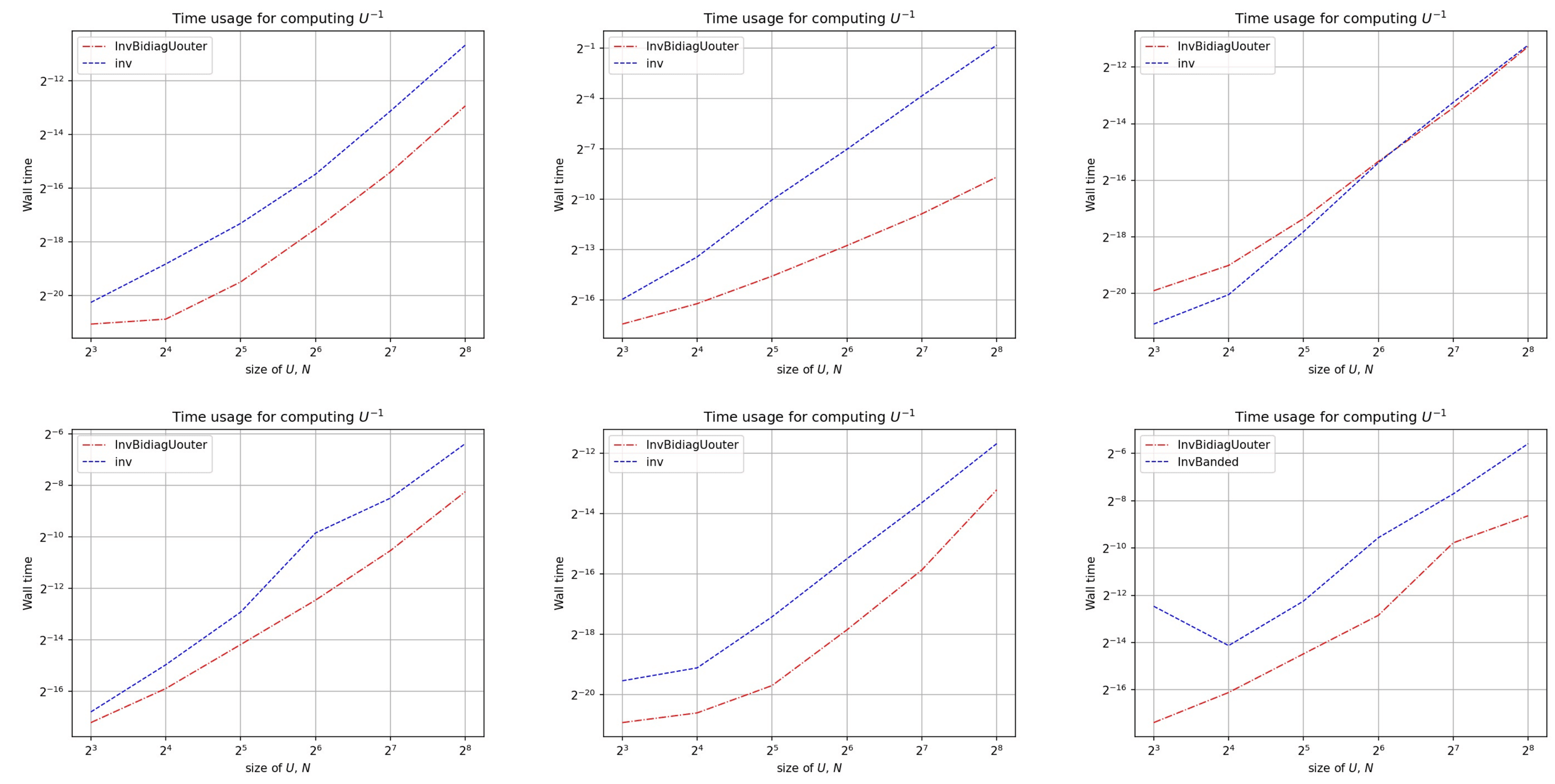
As mentioned in introduction,

$$\begin{aligned} T^{-1} &= (LU)^{-1} = U^{-1}L^{-1} = triu(xy^T)\, triu(pq^T) \\ &= triu(uv^T) + tril(vu^T) = S. \end{aligned} \tag{1}$$

then (1) can be solved by 2 triangular systems in $O(n^2)$, more, the storage is $O(n)$ rather than $O(n^2)$. In 2D problems, the Laplacian Operator is a block tridaigonal matrix or $\Delta \otimes I + I \otimes \Delta$, then →

PERFORMANCE

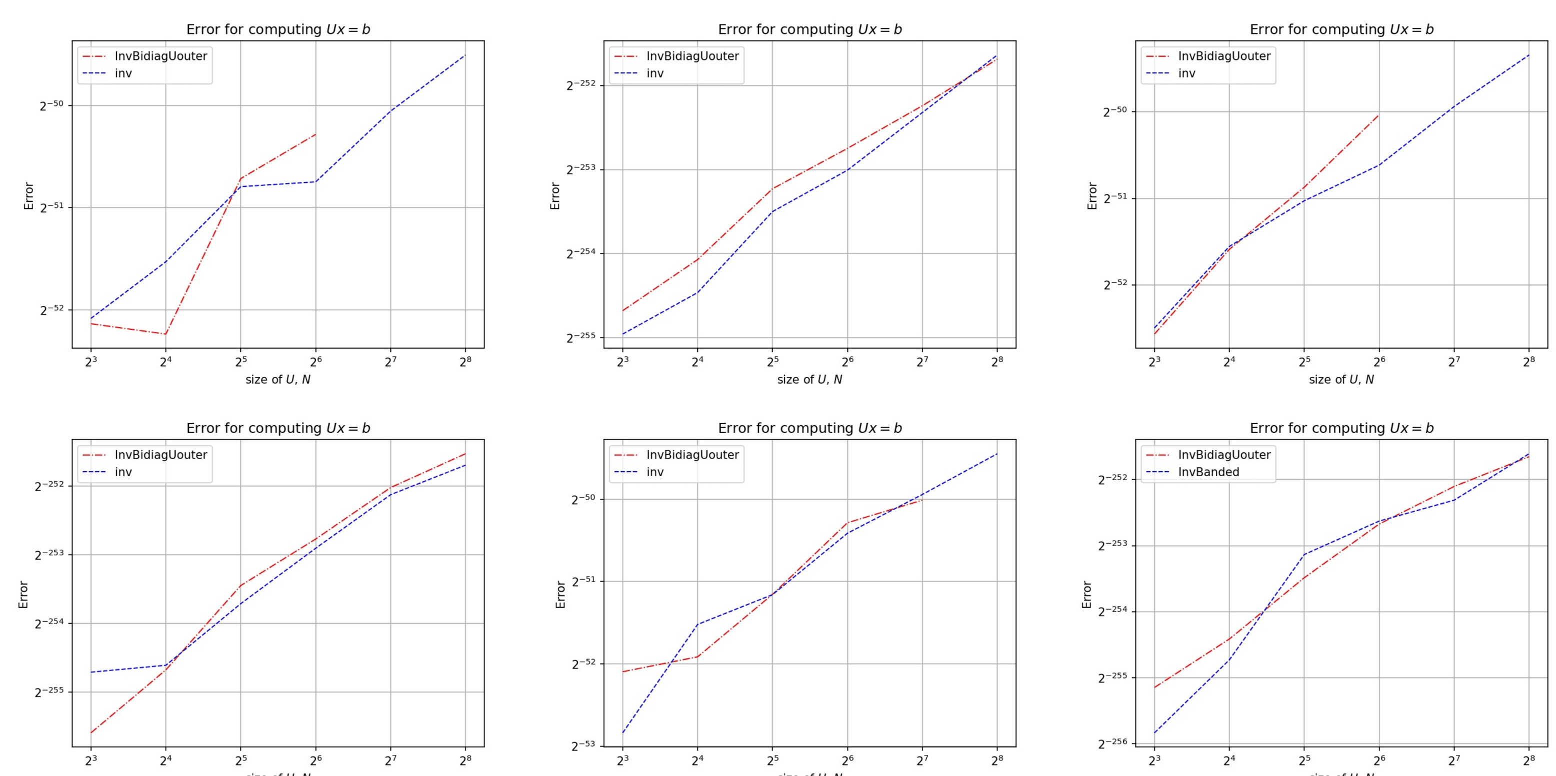
The following figure compares the performance of a modified version of Algorithm 1 and the bulit-in function `inv` through benchmarking the wall-time of computing the inverse of bidiagonal U, U^{-1}, under different sizes and types of U.



What can be seen is that Algorithm 1 performs better overall under both `float64` and `BigFloat`, while the optimisation offered by the structured matrix is huge. The time complexity for using algorithm has been done also, ref. Hunger, 2005.

BACKWARD ERROR

At the same time the introduction of outer product form does introduce numerical instability, which is why the previous test requires `BigFloat` type for entries. The figure below gives a comparison of the backward error, $\|x - \tilde{x}\|$, where the missing data points in red lines under `float64` indicate a precision overflow (`NaN`) during the computations.



Due to the outer product issues, mixed forward-backward error analysis may need to be introduced, and the instability of the algorithm may be improved by linear or exponential rescaling and storage of the numbers in the operation, which is the work at hand, ref. Higham, 2002.

REFERENCES

Golub, Gene H and Charles F Van Loan (2013). *Matrix computations*. JHU press.

Higham, Nicholas J (2002). *Accuracy and stability of numerical algorithms*. SIAM.

Hunger, Raphael (2005). *Floating point operations in matrix-vector calculus*. Vol. 2019. Munich University of Technology, Inst. for Circuit Theory and Signal ...

Saad, Yousef (2003). *Iterative methods for sparse linear systems*. SIAM.

Vandebril, Raf, M Van Barel, et al. (2005). “A bibliography on semiseparable matrices”. In: *Calcolo* 42.3, pp. 249–270.

Vandebril, Raf, Marc Van Barel, and Nicola Mastronardi (2005). “A note on the representation and definition of semiseparable matrices”. In: *Numerical Linear Algebra with Applications* 12.8, pp. 839–858.

APPLICATION (CONTINUED)

→ extending all content to the block case may work for 2D BVPs.