The Journal of Open Source Software

pentapy: A Python toolbox for pentadiagonal linear systems

Sebastian Müller[1, 2]

1 Department of Computational Hydrosystems, UFZ – Helmholtz Centre for Environmental Research, Leipzig, Germany **2** Institute of Earth and Environmental Sciences, University Potsdam, Potsdam, Germany

DOI: 10.21105/joss.01759

Software

- Review
- Repository
- Archive

Submitted: 18 September 2019
Published: 08 October 2019

Summary

Pentadiagonal linear systems of equations arise in many areas of science and engineering: e.g., when solving differential equations, in interpolation problems, or in numerical schemes like finite differences. A specific example is the radial symmetric groundwater flow equation with consecutive rings of different constant transmissivity and radial boundary conditions, which can be expressed as a pentadiagonal equation system (Avci & Ufuk Sahin, 2014; Sindalovskiy, 2017).

Pentadiagonal matrices are banded, being determined by their diagonal, first and second upper minor-diagonals, as well as first and second lower minor-diagonals. These matrices are sparse and can be stored efficiently in a flattened matrix with $5n - 6$ scalars.

A pentadiagonal linear system is given by the equation: $M \cdot X = Y$, where M is a banded quadratic $n \times n$ matrix of the form:

$$M = \begin{pmatrix} d_1 & d_1^{(1)} & d_1^{(2)} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ d_2^{(-1)} & d_2 & d_2^{(1)} & d_2^{(2)} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ d_3^{(-2)} & d_3^{(-1)} & d_3 & d_3^{(1)} & d_3^{(2)} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & d_4^{(-2)} & d_4^{(-1)} & d_4 & d_4^{(1)} & d_4^{(2)} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & d_{n-2}^{(-2)} & d_{n-2}^{(-1)} & d_{n-2} & d_{n-2}^{(1)} & d_{n-2}^{(2)} \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & d_{n-1}^{(-2)} & d_{n-1}^{(-1)} & d_{n-1} & d_{n-1}^{(1)} \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & d_n^{(-2)} & d_n^{(-1)} & d_n \end{pmatrix}$$

Here, d_i are the diagonal entries and $d_i^{(j)}$ represent the j-th minor diagonal.

Recently, Askar & Karawia (2015) presented two algorithms to solve the linear systems of equations for X, `PTRANS-I` and `PTRANS-II`, applying first transformation to a triangular matrix and then, respectively, backward and forward substitution. `pentapy` provides Cython (Behnel et al., 2011) implementations of these algorithms and a set of tools to convert matrices to row-wise or column-wise flattened matrices and vice versa.

Since the algorithms have weak points, for example when the first or last diagonal entry is zero, `pentapy` also provides interfaces to solvers from SciPy (Jones, Oliphant, Peterson, &

others, 2001–2019), like `scipy.linalg.solve_banded` (Lapack) and `scipy.sparse.linalg.spsolve`. The solver can be selected by a keyword argument.

The performance comparison in figure 1, done with `perfplot` (Schlömer, 2019), shows that the implementations of `pentapy` are almost one order of magnitude faster than the SciPy algorithms for banded or sparse matrices. The linear algebra solver of NumPy (Oliphant & others, 2019) served as a standard reference, which disregards the special structure of the equation system.

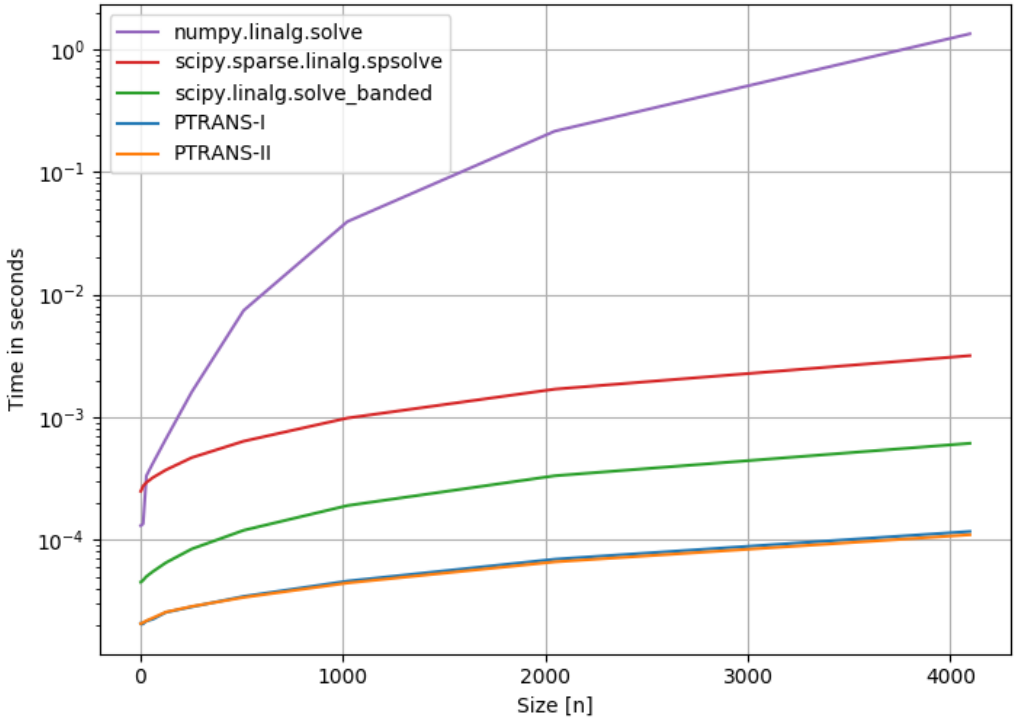

Figure 1: Performance comparison of pentapy, Lapack, SciPy and NumPy routines depending on the matrix size. (system specifications: i5-6200U with 2.3GHz, 16GB RAM)

`pentapy` is designed to provide a fast solver for the special case of a pentadiagonal linear system. To the best of the author's knowledge, this package outperforms the current algorithms for solving pentadiagonal systems in Python. The solver can handle different input formats of the coefficient matrix, i.e., a flattend matrix or a quadratic matrix.

Acknowledgements

I acknowledge the supervision by Prof. Sabine Attinger, Prof. Alraune Zech, Prof. Peter Dietrich and Dr. Falk Heße and herewith want to thank them for their trust and support. I also want to thank Dr. Lennart Schüler for his constant help and the pleasant partnership during the work on the GeoStat Framework. This research was funded by the German Federal Environmental Foundation.

References

Askar, S., & Karawia, A. (2015). On solving pentadiagonal linear systems via transformations. *Mathematical Problems in Engineering*, *2015*. doi:10.1155/2015/232456

Avci, C. B., & Ufuk Sahin, A. (2014). Assessing radial transmissivity variation in heterogeneous aquifers using analytical techniques. *Hydrological Processes*, *28*(23), 5739–5754. doi:10.1002/hyp.10064

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science Engineering*, *13*(2), 31–39. doi:10.1109/MCSE.2010.118

Jones, E., Oliphant, T., Peterson, P., & others. (2001–2019). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Oliphant, T., & others. (2019). NumPy. *GitHub repository*. https://github.com/numpy/numpy; GitHub.

Schlömer, N. (2019). Perfplot. *GitHub repository*. https://github.com/nschloe/perfplot; GitHub.

Sindalovskiy, L. N. (2017). *Aquifer test solutions*. Springer International Publishing. doi:10.1007/978-3-319-43409-4