Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Español
Français
日本語
Edit links

Lanczos algorithm

From Wikipedia, the free encyclopedia
(Redirected from Lanczos iteration)

For the interpolation method, see Lanczos resampling.

The **Lanczos algorithm** is an iterative algorithm devised by Cornelius Lanczos[1] that is an adaptation of power methods to find the most useful eigenvalues and eigenvectors of an n^{th} order linear system with a limited number of operations, m, where m is much smaller than n. Although computationally efficient in principle, the method as initially formulated was not useful, due to its numerical instability. In 1970, Ojalvo and Newman [2] showed how to make the method numerically stable and applied it to the solution of very large engineering structures subjected to dynamic loading. This was achieved using a method for purifying the vectors to any degree of accuracy, which when not performed, produced a series of vectors that were highly contaminated by those associated with the lowest natural frequencies. In their original work, these authors also suggested how to select a starting vector (i.e. use a random number generator to select each element of the starting vector) and suggested an empirically determined method for determining m, the reduced number of vectors (i.e. it should be selected to be approximately 1 ½ times the number of accurate eigenvalues desired). Soon thereafter their work was followed by Paige [3][4] who also provided an error analysis. In 1988, Ojalvo [5] produced a more detailed history of this algorithm and an efficient eigenvalue error test. Currently, the method is widely used in a variety of technical fields and has seen a number of variations.

Contents [hide]

Power method for finding eigenvalues [edit]

Main article: Power iteration

The power method for finding the largest eigenvalue of a matrix A can be summarized by noting that if x_0 is a random vector and $x_{n+1} = Ax_n$, then in the large n limit, $x_n / \|x_n\|$ approaches the normed eigenvector corresponding to the largest magnitude eigenvalue.

If $A = U \operatorname{diag}(\sigma_i) U'$ is the eigendecomposition of A, then $A^n = U \operatorname{diag}(\sigma_i^n) U'$. As n gets very large, the diagonal matrix of eigenvalues will be dominated by whichever eigenvalue is largest (neglecting the case of two or more equally large eigenvalues, of course). As this happens, $x_n^* x_{n+1} / x_n^* x_n$ will converge to the largest eigenvalue and $x_n / \|x_n\|$ to the associated eigenvector. If the largest eigenvalue is multiple, then x_n will converge to a vector in the subspace spanned by the eigenvectors associated with those largest eigenvalues. Having found the first eigenvector/value, one can then successively restrict the algorithm to the null space of the known eigenvectors to get the second largest eigenvector/values and so on.

In practice, this simple algorithm does not work very well for computing very many of the eigenvectors because any round-off error will tend to introduce slight components of the more significant eigenvectors back into the computation, degrading the accuracy of the computation. Pure power methods also can converge slowly, even for the first eigenvector.

Lanczos method [edit]

During the procedure of applying the power method, while getting the ultimate eigenvector $A^{n-1}v$, we also got a series of vectors $A^j v,\ j = 0, 1, \cdots, n-2$ which were eventually discarded. As n is often taken to be quite large, this can result in a large amount of disregarded information. More advanced algorithms, such as Arnoldi's algorithm and the Lanczos algorithm, save this information and use the Gram–Schmidt process or Householder algorithm to reorthogonalize them into a basis spanning the Krylov subspace corresponding to the matrix A.

The algorithm [edit]

The Lanczos algorithm can be viewed as a simplified Arnoldi's algorithm in that it applies to Hermitian matrices. The m'th step of the algorithm transforms the matrix A into a tridiagonal matrix T_{mm}; when m is equal to the dimension of A, T_{mm} is similar to A.

Definitions [edit]

We hope to calculate the tridiagonal and symmetric matrix $T_{mm} = V_m^* A V_m$.

The diagonal elements are denoted by $\alpha_j = t_{jj}$, and the off-diagonal elements are denoted by $\beta_j = t_{j-1,j}$.

Note that $t_{j-1,j} = t_{j,j-1}$, due to its symmetry.

Iteration [edit]

(Note: Following these steps alone will **not** give you the correct eigenvalue and eigenvectors. More consideration must be applied to correct for the numerical errors. See the section Numerical stability in the following.)

There are in principle four ways to write the iteration procedure. Paige[1972] and other works show that the following procedure is the most numerically stable.[6][7]

Algorithm Lanczos

$v_1 \leftarrow$ random vector with norm 1.
$v_0 \leftarrow 0$
$\beta_1 \leftarrow 0$

for $j = 1, 2, \cdots, m-1$
  $w_j \leftarrow A v_j$
  $\alpha_j \leftarrow w_j \cdot v_j$
  $w_j \leftarrow w_j - \alpha_j v_j - \beta_j v_{j-1}$
  $\beta_{j+1} \leftarrow \|w_j\|$
  $v_{j+1} \leftarrow w_j / \beta_{j+1}$
endfor

$w_m \leftarrow A v_m$
$\alpha_m \leftarrow w_m \cdot v_m$
return

- "←" is a shorthand for "changes to". For instance, "*largest* ← *item*" means that the value of *largest* changes to the value of *item*.
- "**return**" terminates the algorithm and outputs the value that follows.

Here, $x \cdot y$ represents the dot product of vectors x and y.

After the iteration, we get the α_j and β_j which construct a tridiagonal matrix

$$T_{mm} = \begin{pmatrix} \alpha_1 & \beta_2 & & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \beta_{m-1} & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & & \beta_m & \alpha_m \end{pmatrix}$$

The vectors v_j (**Lanczos vectors**) generated on the fly construct the transformation matrix

$$V_m = (v_1, v_2, \cdots, v_m),$$

which is useful for calculating the eigenvectors (see below). In practice, it could be saved after generation (but takes a lot of memory), or could be regenerated when needed, as long as one keeps the first vector v_1. At each iteration the algorithm executes a matrix-vector multiplication and 7n further floating point operations.

Solve for eigenvalues and eigenvectors [edit]

After the matrix T_{mm} is calculated, one can solve its eigenvalues $\lambda_i^{(m)}$ and their corresponding eigenvectors $u_i^{(m)}$ (for example, using the QR algorithm or Multiple Relatively Robust Representations (MRRR)). The eigenvalues and eigenvectors of T can be obtained in as little as $\mathcal{O}(m^2)$ work with MRRR; obtaining just the eigenvalues is much simpler and can be done in $\mathcal{O}(m^2)$ work with spectral bisection.

It can be proved that the eigenvalues are approximate eigenvalues of the original matrix A.

The Ritz eigenvectors y_i of A can be calculated by $y_i = V_m u_i^{(m)}$, where V_m is the transformation matrix whose column vectors are $v_1, v_2, \cdots, v_m$.

Numerical stability [edit]

Stability means how much the algorithm will be affected (i.e. will it produce the approximate result close to the original one) if there are small numerical errors introduced and accumulated. Numerical stability is the central criterion for judging the usefulness of implementing an algorithm on a computer with roundoff.

For the Lanczos algorithm, it can be proved that with *exact arithmetic*, the set of vectors $v_1, v_2, \cdots, v_{m+1}$ constructs an *orthonormal* basis, and the eigenvalues/vectors solved are good approximations to those of the original matrix. However, in practice (as the calculations are performed in floating point arithmetic where inaccuracy is inevitable), the orthogonality is quickly lost and in some cases the new vector could even be linearly dependent on the set that is already constructed. As a result, some of the eigenvalues of the resultant tridiagonal matrix may not be approximations to the original matrix. Therefore, the Lanczos algorithm is not very stable.

Users of this algorithm must be able to find and remove those "spurious" eigenvalues. Practical implementations of the Lanczos algorithm go in three directions to fight this stability issue:[6][7]

1. Prevent the loss of orthogonality
2. Recover the orthogonality after the basis is generated
3. After the good and "spurious" eigenvalues are all identified, remove the spurious ones.

Variations [edit]

Variations on the Lanczos algorithm exist where the vectors involved are tall, narrow matrices instead of vectors and the normalizing constants are small square matrices. These are called "block" Lanczos algorithms and can be much faster on computers with large numbers of registers and long memory-fetch times.

Many implementations of the Lanczos algorithm restart after a certain number of iterations. One of the most influential restarted variations is the implicitly restarted Lanczos method,[8] which is implemented in ARPACK.[9] This has led into a number of other restarted variations such as restarted Lanczos bidiagonalization.[10] Another successful restarted variation is the Thick-Restart Lanczos method,[11] which has been implemented in a software package called TRLan.[12]

Nullspace over a finite field [edit]

Main article: Block Lanczos algorithm

In 1995, Peter Montgomery published an algorithm, based on the Lanczos algorithm, for finding elements of the nullspace of a large sparse matrix over GF(2); since the set of people interested in large sparse matrices over finite fields and the set of people interested in large eigenvalue problems scarcely overlap, this is often also called the *block Lanczos algorithm* without causing unreasonable confusion.[*citation needed*]

Applications [edit]

Lanczos algorithms are very attractive because the multiplication by A is the only large-scale linear operation. Since weighted-term text retrieval engines implement just this operation, the Lanczos algorithm can be applied efficiently to text documents (see Latent Semantic Indexing). Eigenvectors are also important for large-scale ranking methods such as the HITS algorithm developed by Jon Kleinberg, or the PageRank algorithm used by

Google.

Lanczos algorithms are also used in Condensed Matter Physics as a method for solving Hamiltonians of strongly correlated electron systems.[13]

Lanczos algorithm has also been used in the formulation of the Levenberg-Marquardt or the Gauss-Newton optimization for solving nonlinear inverse problems (such as generating computational models of oil and gas reservoirs given observed production data).[14]

Implementations [edit]

The NAG Library contains several routines[15] for the solution of large scale linear systems and eigenproblems which use the Lanczos algorithm.

MATLAB and GNU Octave come with ARPACK built-in. Both stored and implicit matrices can be analyzed through the *eigs()* function (Matlab/Octave).

A Matlab implementation of the Lanczos algorithm (note precision issues) is available as a part of the Gaussian Belief Propagation Matlab Package. The GraphLab[16] collaborative filtering library incorporates a large scale parallel implementation of the Lanczos algorithm (in C++) for multicore.

The PRIMME library also implements a Lanczos like algorithm.

References [edit]

1. ^ Lanczos, C. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", J. Res. Nat'l Bur. Std. 45, 225-282 (1950).
2. ^ Ojalvo, I.U. and Newman, M.,"Vibration modes of large structures by an automatic matrix-reduction method", AIAA J., 8 (7), 1234-1239 (1970).
3. ^ Paige, C.C., "The computation of eigenvalues and eigenvectors of very large sparse matrices", the U. of London Ph.D. thesis (1971).
4. ^ Paige, C.C., "Computational variants of the Lanczos method for the eigenproblem", J. Inst. Maths Applics 10, 373-381 (1972).
5. ^ Ojalvo, I.U., "Origins and advantages of Lanczos vectors for large dynamic systems", Proc. 6th Modal Analysis Conference (IMAC), Kissimmee, FL, 489-494 (1988).
6. ^ [a] [b] Cullum; Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations* **1**. ISBN 0-8176-3058-9.
7. ^ [a] [b] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. ISBN 0-470-21820-7.
8. ^ D. Calvetti, L. Reichel, and D.C. Sorensen (1994). "An Implicitly Restarted Lanczos Method for Large Symmetric Eigenvalue Problems". *Electronic Transactions on Numerical Analysis* **2**: 1–21.
9. ^ R. B. Lehoucq, D. C. Sorensen, and C. Yang (1998). *ARPACK Users Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM. doi:10.1137/1.9780898719628.
10. ^ E. Kokiopoulou and C. Bekas and E. Gallopoulos (2004). "Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization". *Appl. Numer. Math.* **49**: 39–61. doi:10.1016/j.apnum.2003.11.011.
11. ^ Kesheng Wu and Horst Simon (2000). "Thick-Restart Lanczos Method for Large Symmetric Eigenvalue Problems". *SIAM Journal on Matrix Analysis and Applications* (SIAM) **22** (2): 602–616. doi:10.1137/S0895479898334605.
12. ^ Kesheng Wu and Horst Simon (2001). "TRLan software package".
13. ^ Chen, HY; Atkinson, W.A.; Wortis, R. (July 2011). "Disorder-induced zero-bias anomaly in the Anderson-Hubbard model: Numerical and analytical calculations". *Physical Review B* **84** (4). doi:10.1103/PhysRevB.84.045113.
14. ^ History matching production data and uncertainty assessment with an efficient TSVD parameterization algorithm, Journal of Petroleum Science and Engineering http://www.sciencedirect.com/science/article/pii/S0920410513003227
15. ^ The Numerical Algorithms Group. "Keyword Index: Lanczos". *NAG Library Manual, Mark 23*. Retrieved 2012-02-09.
16. ^ GraphLab

External links [edit]

- Golub and van Loan give very good descriptions of the various forms of Lanczos algorithms in their book *Matrix Computations*
- Andrew Ng et al., an analysis of PageRank
- Lanczos and conjugate gradient methods B. A. LaMacchia and A. M. Odlyzko, Solving Large Sparse Linear Systems Over Finite Fields.

Key concepts	Floating point · Numerical stability
Problems	Matrix multiplication (algorithms) · Matrix decompositions · Linear equations · Sparse problems
Hardware	CPU cache · TLB · Cache-oblivious algorithm · SIMD · Multiprocessing
Software	BLAS · Specialized libraries · General purpose software

Categories: Numerical linear algebra