



WIKIPEDIA
The Free Encyclopedia

[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


[Español](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)



Levinson recursion

From Wikipedia, the free encyclopedia

Levinson recursion or **Levinson–Durbin recursion** is a procedure in [linear algebra](#) to [recursively](#) calculate the solution to an equation involving a [Toeplitz matrix](#). The [algorithm](#) runs in $\Theta(n^2)$ time, which is a strong improvement over [Gauss–Jordan elimination](#), which runs in $\Theta(n^3)$.

The Levinson–Durbin algorithm was proposed first by [Norman Levinson](#) in 1947, improved by [James Durbin](#) in 1960, and subsequently improved to $4n^2$ and then $3n^2$ multiplications by W. F. Trench and S. Zohar, respectively.

Other methods to process data include [Schur decomposition](#) and [Cholesky decomposition](#). In comparison to these, Levinson recursion (particularly split Levinson recursion) tends to be faster computationally, but more sensitive to computational inaccuracies like [round-off errors](#).

The Bareiss algorithm for [Toeplitz matrices](#) (not to be confused with the general [Bareiss algorithm](#)) runs about as fast as Levinson recursion, but it uses $O(n^2)$ space, whereas Levinson recursion uses only $O(n)$ space. The Bareiss algorithm, though, is [numerically stable](#),^{[1][2]} whereas Levinson recursion is at best only weakly stable (i.e. it exhibits numerical stability for [well-conditioned](#) linear systems).^[3]

Newer algorithms, called *asymptotically fast* or sometimes *superfast* Toeplitz algorithms, can solve in $\Theta(n \log^p n)$ for various p (e.g. $p = 2$,^{[4][5]} $p = 3$ ^[6]). Levinson recursion remains popular for several reasons; for one, it is relatively easy to understand in comparison; for another, it can be faster than a superfast algorithm for small n (usually $n < 256$).^[7]

Contents

- 1 Derivation
 - 1.1 Background
 - 1.2 Introductory steps
 - 1.3 Obtaining the backward vectors
 - 1.4 Using the backward vectors
- 2 Block Levinson algorithm
- 3 See also
- 4 Notes
- 5 References

Derivation [\[edit \]](#)

Background [\[edit \]](#)

Matrix equations follow the form:

$$\vec{y} = \mathbf{M} \vec{x}.$$

The Levinson–Durbin algorithm may be used for any such equation, as long as **M** is a known [Toeplitz matrix](#) with a nonzero main diagonal. Here \vec{y} is a known [vector](#), and \vec{x} is an unknown vector of numbers x_i yet to be determined.

For the sake of this article, \hat{e}_i is a vector made up entirely of zeroes, except for its i th place, which holds the value one. Its length will be implicitly determined by the surrounding context. The term N refers to the width of the matrix above – **M** is an $N \times N$ matrix. Finally, in this article, superscripts refer to an *inductive index*, whereas subscripts denote indices. For example (and definition), in this article, the matrix **T**^{*n*} is an $n \times n$ matrix which copies the upper left $n \times n$ block from **M** – that is, $T^n_{ij} = M_{ij}$.

T^{*n*} is also a Toeplitz matrix; meaning that it can be written as:

$$\mathbf{T}^n = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \dots & t_{-n+1} \\ t_1 & t_0 & t_{-1} & \dots & t_{-n+2} \\ t_2 & t_1 & t_0 & \dots & t_{-n+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & t_{n-3} & \dots & t_0 \end{bmatrix}.$$

Introductory steps [\[edit \]](#)

The algorithm proceeds in two steps. In the first step, two sets of vectors, called the *forward* and *backward* vectors, are established. The forward vectors are used to help get the set of backward vectors; then they can be immediately discarded. The backwards vectors are necessary for the second step, where they are used to build the solution desired.

Levinson–Durbin recursion defines the n^{th} "forward vector", denoted \vec{f}^n , as the vector of length n which satisfies:

$$\mathbf{T}^n \vec{f}^n = \hat{e}_1.$$

The n^{th} "backward vector" \vec{b}^n is defined similarly; it is the vector of length n which satisfies:

$$\mathbf{T}^n \vec{b}^n = \hat{e}_n.$$

An important simplification can occur when \mathbf{M} is a [symmetric matrix](#); then the two vectors are related by $b_i^n = f_{n+1-i}^n$ —that is, they are row-reversals of each other. This can save some extra computation in that special case.

Obtaining the backward vectors [\[edit \]](#)

Even if the matrix is not symmetric, then the n^{th} forward and backward vector may be found from the vectors of length $n - 1$ as follows. First, the forward vector may be extended with a zero to obtain:

$$\mathbf{T}^n \begin{bmatrix} \vec{f}^{n-1} \\ 0 \end{bmatrix} = \begin{bmatrix} & & & t_{-n+1} & \\ & & & t_{-n+2} & \\ & & & \vdots & \\ & & & t_0 & \\ t_{n-1} & t_{n-2} & \dots & & \end{bmatrix} \begin{bmatrix} \vec{f}^{n-1} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \epsilon_f^n \end{bmatrix}.$$

In going from \mathbf{T}^{n-1} to \mathbf{T}^n , the extra *column* added to the matrix does not perturb the solution when a zero is used to extend the forward vector. However, the extra *row* added to the matrix *has* perturbed the solution; and it has created an unwanted error term ϵ_f which occurs in the last place. The above equation gives it the value of:

$$\epsilon_f^n = \sum_{i=1}^{n-1} M_{ni} f_i^{n-1} = \sum_{i=1}^{n-1} t_{n-i} f_i^{n-1}.$$

This error will be returned to shortly and eliminated from the new forward vector; but first, the backwards vector must be extended in a similar (albeit reversed) fashion. For the backwards vector,

$$\mathbf{T}^n \begin{bmatrix} 0 \\ \vec{b}^{n-1} \end{bmatrix} = \begin{bmatrix} t_0 & \dots & t_{-n+2} & t_{-n+1} \\ \vdots & & & \\ t_{n-2} & & \mathbf{T}^{n-1} & \\ t_{n-1} & & & \end{bmatrix} \begin{bmatrix} 0 \\ \vec{b}^{n-1} \end{bmatrix} = \begin{bmatrix} \epsilon_b^n \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

As before, the extra column added to the matrix does not perturb this new backwards vector; but the extra row does. Here we have another unwanted error ϵ_b with value:

$$\epsilon_b^n = \sum_{i=2}^n M_{1i} b_{i-1}^{n-1} = \sum_{i=1}^{n-1} t_{-i} b_i^{n-1}.$$

These two error terms can be used to eliminate each other. Using the linearity of matrices,

$$\forall(\alpha, \beta) \mathbf{T} \left(\alpha \begin{bmatrix} \vec{f} \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ \vec{b} \end{bmatrix} \right) = \alpha \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \epsilon_f \end{bmatrix} + \beta \begin{bmatrix} \epsilon_b \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

If α and β are chosen so that the right hand side yields \hat{e}_1 or \hat{e}_n , then the quantity in the parentheses will fulfill the definition of the n^{th} forward or backward vector, respectively. With those alpha and beta chosen, the vector sum in the parentheses is simple and yields the desired result.

To find these coefficients, α_f^n, β_f^n are such that :

$$\vec{f}_n = \alpha_f^n \begin{bmatrix} \vec{f}_{n-1} \\ 0 \end{bmatrix} + \beta_f^n \begin{bmatrix} 0 \\ \vec{b}_{n-1} \end{bmatrix}$$

and respectively α_b^n, β_b^n are such that :

$$\vec{b}_n = \alpha_b^n \begin{bmatrix} \vec{f}_{n-1} \\ 0 \end{bmatrix} + \beta_b^n \begin{bmatrix} 0 \\ \vec{b}_{n-1} \end{bmatrix}.$$

By multiplying both previous equations by \mathbf{T}^n one gets the following equation:

$$\begin{bmatrix} 1 & \epsilon_b^n \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ \epsilon_f^n & 1 \end{bmatrix} \begin{bmatrix} \alpha_f^n & \alpha_b^n \\ \beta_f^n & \beta_b^n \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Now, all the zeroes in the middle of the two vectors above being disregarded and collapsed, only the following equation is left:

$$\begin{bmatrix} 1 & \epsilon_b^n \\ \epsilon_f^n & 1 \end{bmatrix} \begin{bmatrix} \alpha_f^n & \alpha_b^n \\ \beta_f^n & \beta_b^n \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

With these solved for (by using the Cramer 2x2 matrix inverse formula), the new forward and backward vectors are:

$$\vec{f}_n = \frac{1}{1 - \epsilon_b^n \epsilon_f^n} \begin{bmatrix} \vec{f}_{n-1} \\ 0 \end{bmatrix} - \frac{\epsilon_f^n}{1 - \epsilon_b^n \epsilon_f^n} \begin{bmatrix} 0 \\ \vec{b}_{n-1} \end{bmatrix}$$

$$\vec{b}_n = \frac{1}{1 - \epsilon_b^n \epsilon_f^n} \begin{bmatrix} 0 \\ \vec{b}_{n-1} \end{bmatrix} - \frac{\epsilon_b^n}{1 - \epsilon_b^n \epsilon_f^n} \begin{bmatrix} \vec{f}_{n-1} \\ 0 \end{bmatrix}.$$

Performing these vector summations, then, gives the n^{th} forward and backward vectors from the prior ones. All that remains is to find the first of these vectors, and then some quick sums and multiplications give the remaining ones. The first forward and backward vectors are simply:

$$\vec{f}^1 = \vec{b}^1 = \begin{bmatrix} 1 \\ M_{11} \end{bmatrix} = \begin{bmatrix} 1 \\ t_0 \end{bmatrix}.$$

Using the backward vectors [\[edit \]](#)

The above steps give the N backward vectors for \mathbf{M} . From there, a more arbitrary equation is:

$$\vec{y} = \mathbf{M} \vec{x}.$$

The solution can be built in the same recursive way that the backwards vectors were built. Accordingly, \vec{x} must be generalized to a sequence \vec{x}^n , from which $\vec{x}^N = \vec{x}$.

The solution is then built recursively by noticing that if

$$\mathbf{T}^{n-1} \begin{bmatrix} x_1^{n-1} \\ x_2^{n-1} \\ \vdots \\ x_{n-1}^{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix}.$$

Then, extending with a zero again, and defining an error constant where necessary:

$$\mathbf{T}^n \begin{bmatrix} x_1^{n-1} \\ x_2^{n-1} \\ \vdots \\ x_{n-1}^{n-1} \\ 0 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ \epsilon_x^{n-1} \end{bmatrix}.$$

We can then use the n^{th} backward vector to eliminate the error term and replace it with the desired formula as follows:

$$\mathbf{T}^n \left(\begin{bmatrix} x_1^{n-1} \\ x_2^{n-1} \\ \vdots \\ x_{n-1}^{n-1} \\ 0 \end{bmatrix} + (y_n - \epsilon_x^{n-1}) \vec{b}^n \right) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}.$$

Extending this method until $n = N$ yields the solution \vec{x} .

In practice, these steps are often done concurrently with the rest of the procedure, but they form a coherent unit and deserve to be treated as their own step.





Block Levinson algorithm [[edit](#)]

If \mathbf{M} is not strictly Toeplitz, but **block** Toeplitz, the Levinson recursion can be derived in much the same way by regarding the block Toeplitz matrix as a Toeplitz matrix with matrix elements (Musicus 1988). Block Toeplitz matrices arise naturally in signal processing algorithms when dealing with multiple signal streams (e.g., in **MIMO** systems) or cyclo-stationary signals.

See also [[edit](#)]


- Split Levinson recursion**
- Linear prediction**
- Autoregressive model**

Notes [[edit](#)]


- ↑ Bojanczyk et al. (1995).
- ↑ Brent (1999).
- ↑ Krishna & Wang (1993).
- ↑ http://www.maths.anu.edu.au/~brent/pd/rpb143tr.pdf 
- ↑ http://etd.gsu.edu/theses/available/etd-04182008-174330/unrestricted/kimitei_symon_k_200804.pdf 
- ↑ http://web.archive.org/web/20070418074240/http://saaz.cs.gsu.edu/papers/sfast.pdf 
- ↑ http://www.math.niu.edu/~ammarr/papers/amgr88.pdf 




References [[edit](#)]

Defining sources




- Levinson, N. (1947). "The Wiener RMS error criterion in filter design and prediction." *J. Math. Phys.*, v. 25, pp. 261–278.
- Durbin, J. (1960). "The fitting of time series models." *Rev. Inst. Int. Stat.*, v. 28, pp. 233–243.
- Trench, W. F. (1964). "An algorithm for the inversion of finite Toeplitz matrices." *J. Soc. Indust. Appl. Math.*, v. 12, pp. 515–522.
- Musicus, B. R. (1988). "Levinson and Fast Choleski Algorithms for Toeplitz and Almost Toeplitz Matrices." *RLE TR* No. 538, MIT. [[1](#)] 
- Delsarte, P. and Genin, Y. V. (1986). "The split Levinson algorithm." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-34(3), pp. 470–478.

Further work

- Bojanczyk, A.W.; Brent, R.P.; De Hoog, F.R.; Sweet, D.R. (1995). "On the stability of the Bareiss and related Toeplitz factorization algorithms". *SIAM Journal on Matrix Analysis and Applications* **16**: 40–57. doi:10.1137/S0895479891221563 .
- Brent R.P.** (1999), "Stability of fast algorithms for structured linear systems", *Fast Reliable Algorithms for Matrices with Structure* (editors—T. Kailath, A.H. Sayed), ch.4 (**SIAM**).

- Bunch, J. R. (1985). "Stability of methods for solving Toeplitz systems of equations." *SIAM J. Sci. Stat. Comput.*, v. 6, pp. 349–364. [2] 
- Krishna, H.; Wang, Y. (1993). "The Split Levinson Algorithm is weakly stable" . *SIAM Journal on Numerical Analysis* **30** (5): 1498–1508. doi:10.1137/0730078 .

Summaries

- Bäckström, T. (2004). "2.2. Levinson–Durbin Recursion." *Linear Predictive Modelling of Speech – Constraints and Line Spectrum Pair Decomposition*. Doctoral thesis. Report no. 71 / Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing. Espoo, Finland. [3] 
- Claerbout, Jon F. (1976). "Chapter 7 – Waveform Applications of Least-Squares." *Fundamentals of Geophysical Data Processing*. Palo Alto: Blackwell Scientific Publications. [4] 
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 2.8.2. Toeplitz Matrices" , *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8
- Golub, G.H., and Loan, C.F. Van (1996). "Section 4.7 : Toeplitz and related Systems" *Matrix Computations*, Johns Hopkins University Press

Categories: [Matrices](#) | [Numerical analysis](#)

This page was last modified on 4 March 2015, at 00:43.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

