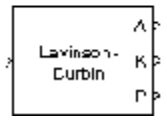
Levinson-Durbin

Solve linear system of equations using Levinson-Durbin recursion

Library

Math Functions / Matrices and Linear Algebra / Linear System Solvers

Description



The Levinson-Durbin block solves the *n*th-order system of linear equations

$$Ra = b$$

for the particular case where *R* is a Hermitian, positive-definite, Toeplitz matrix and *b* is identical to the first column of *R* shifted by one element and with the opposite sign.

$$\begin{bmatrix} r(1) & r^*(2) & \cdots & r^*(n) \\ r(2) & r(1) & \cdots & r^*(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(n) & r(n-1) & \cdots & r(1) \end{bmatrix} \begin{bmatrix} a(2) \\ a(3) \\ \vdots \\ a(n+1) \end{bmatrix} = \begin{bmatrix} -r(2) \\ -r(3) \\ \vdots \\ -r(n+1) \end{bmatrix}$$

The input to the block, *r* = `[`*r*`(1) r(2) ... r(`*n*`+1)]`, can be a 1-D or 2-D vector (row or column). It contains lags *0* through *n* of an autocorrelation sequence, which appear in the matrix *R*.

The block can output the polynomial coefficients, *A*, the reflection coefficients, *K*, and the prediction error power, *P*, in various combinations. The **Output(s)** parameter allows you to enable the *A* and *K* outputs by selecting one of the following settings:

- `A` — Port `A` outputs *A*=`[1 a(2) a(3) ... a(`*n*`+1)]`, the solution to the Levinson-Durbin equation. *A* has the same dimension as the input. The elements of the output can also be viewed as the coefficients of an *n*th-order autoregressive (AR) process (see below).
- `K` — Port `K` outputs *K*=`[k(1) k(2) ... k(`*n*`)]`, which contains *n* reflection coefficients, and has the same dimension as the input, less one element. (A scalar input causes an error when you select `K`.) Reflection coefficients are useful for realizing a lattice representation of the AR process described below.
- `A and K` — The block outputs both representations at their respective ports. (A scalar input causes an error when you select `A and K`.)

Both *A* and *K* are always 1-D vectors.

The prediction error power, *P*, (a scalar), is output when you select the **Output prediction error power (P)**

check box. *P* represents the power of the output of an FIR filter with taps *A* and input autocorrelation described by *r*, where *A* represents a prediction error filter and *r* is the input to the block. (In this case, *A* is a whitening filter.)

When you select the **If the value of lag 0 is zero, A=[1 zeros], K=[zeros], P=0** check box (default), an input whose *r*(1) element is zero generates a zero-valued output. When you do *not* select this check box, an input with *r*(1) = 0 generates NaNs in the output. In general, an input with *r*(1) = 0 is invalid because it does not construct a positive-definite matrix *R*; however, it is common for blocks to receive zero-valued inputs at the start of a simulation. The check box allows you to avoid propagating NaNs during this period.

Applications

One application of the Levinson-Durbin formulation above is in the Yule-Walker AR problem, which concerns modeling an unknown system as an autoregressive process. Such a process would be modeled as the output of an all-pole IIR filter with white Gaussian noise input. In the Yule-Walker problem, the use of the signal's autocorrelation sequence to obtain an optimal estimate leads to an *Ra* = *b* equation of the type shown above, which is most efficiently solved by Levinson-Durbin recursion. In this case, the input to the block represents the autocorrelation sequence, with r(1) being the zero-lag value. The output at the block's A port then contains the coefficients of the autoregressive process that optimally models the system. The coefficients are ordered in descending powers of *z*, and the AR process is minimum phase. The prediction error, *G*, defines the gain for the unknown system, where $G = \sqrt{P}$.

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 + a(2)z^{-1} + \ldots + a(n+1)z^{-n}}$$

The output at the block's K port contains the corresponding reflection coefficients, [*k*(1) *k*(2) ... *k*(n)], for the lattice realization of this IIR filter. The Yule-Walker AR Estimator block implements this autocorrelation-based method for AR model estimation, while the Yule-Walker Method block extends the method to spectral estimation.

Another common application of the Levinson-Durbin algorithm is in linear predictive coding, which is concerned with finding the coefficients of a moving average (MA) process (or FIR filter) that predicts the next value of a signal from the current signal sample and a finite number of past samples. In this case, the input to the block represents the signal's autocorrelation sequence, with *r*(1) being the zero-lag value, and the output at the block's A port contains the coefficients of the predictive MA process (in descending powers of *z*).

$$H(z) = A(z) = 1 + a(2)z^{-1} + \ldots + a(n+1)z^{-n}$$

These coefficients solve the optimization problem below.

$$\min_{\{a_i\}} E\left[\left|x_n - \sum_{i=1}^{N} a_i x_{n-i}\right|^2\right]$$

Again, the output at the block's K port contains the corresponding reflection coefficients, [k(1) k(2) ... k(n)], for the lattice realization of this FIR filter. The Autocorrelation LPC block in the Linear Prediction library implements this autocorrelation-based prediction method.

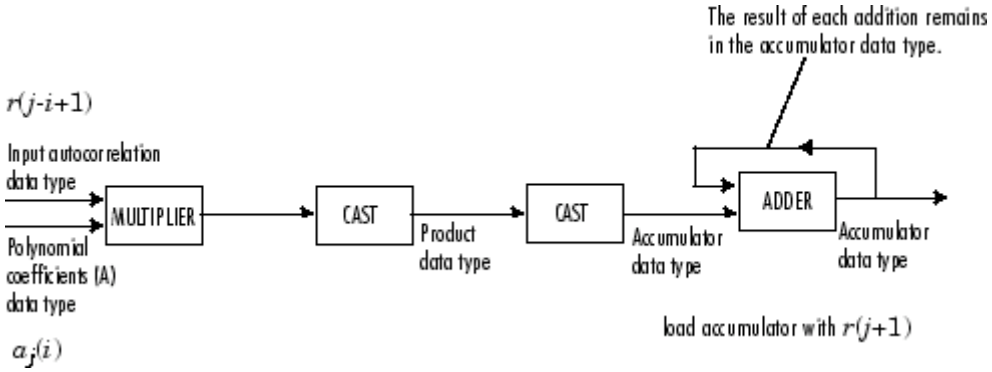
Fixed-Point Data Types

The diagrams in this section show the data types used within the Levinson-Durbin block for fixed-point signals.

After initialization, *n* updates are performed. At the (*j*+1) update,

$$\text{value in accumulator} = r(j+1) + \sum a_j(i) \times r(j-i+1)$$

The diagram below displays the fixed-point data types used in this calculation:



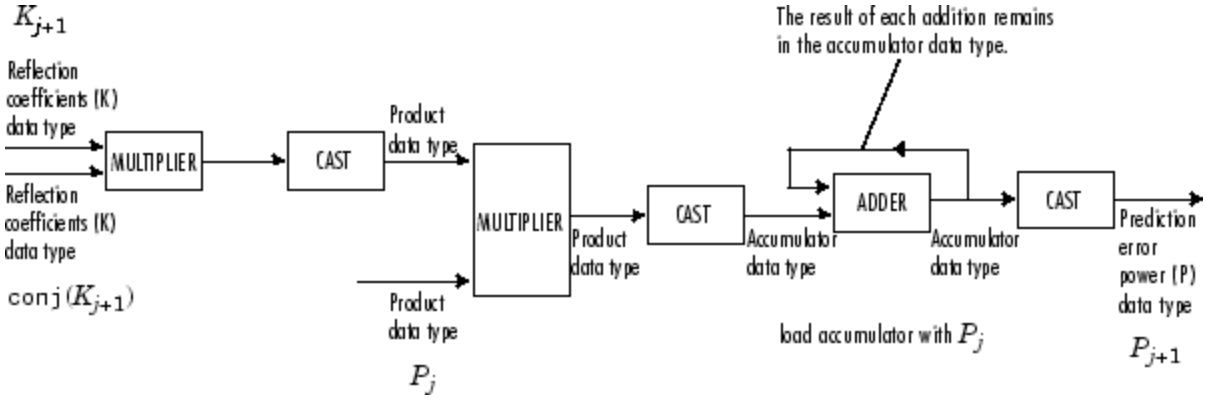
The reflection coefficients *K* are then updated according to

$$K_{j+1} = \text{value in accumulator} / P_j$$

The prediction error power *P* is then updated according to

$$P_{j+1} = P_j - P_j \times K_{j+1} \times \text{conj}(K_{j+1})$$

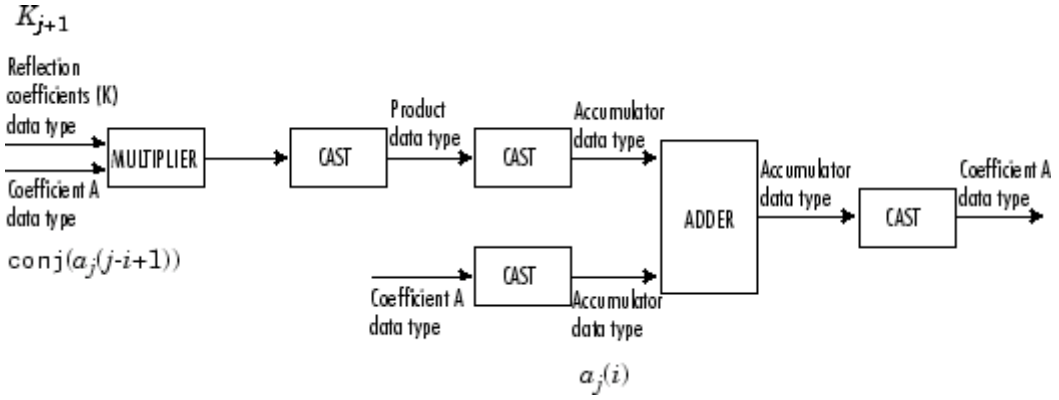
The diagram below displays the fixed-point data types used in this calculation:



The polynomial coefficients *A* are then updated according to

$$a_{j+1}(i) = a_j(i) + K_{j+1} \times \text{conj}(a_j(j-1+i))$$

The diagram below displays the fixed-point data types used in this calculation:

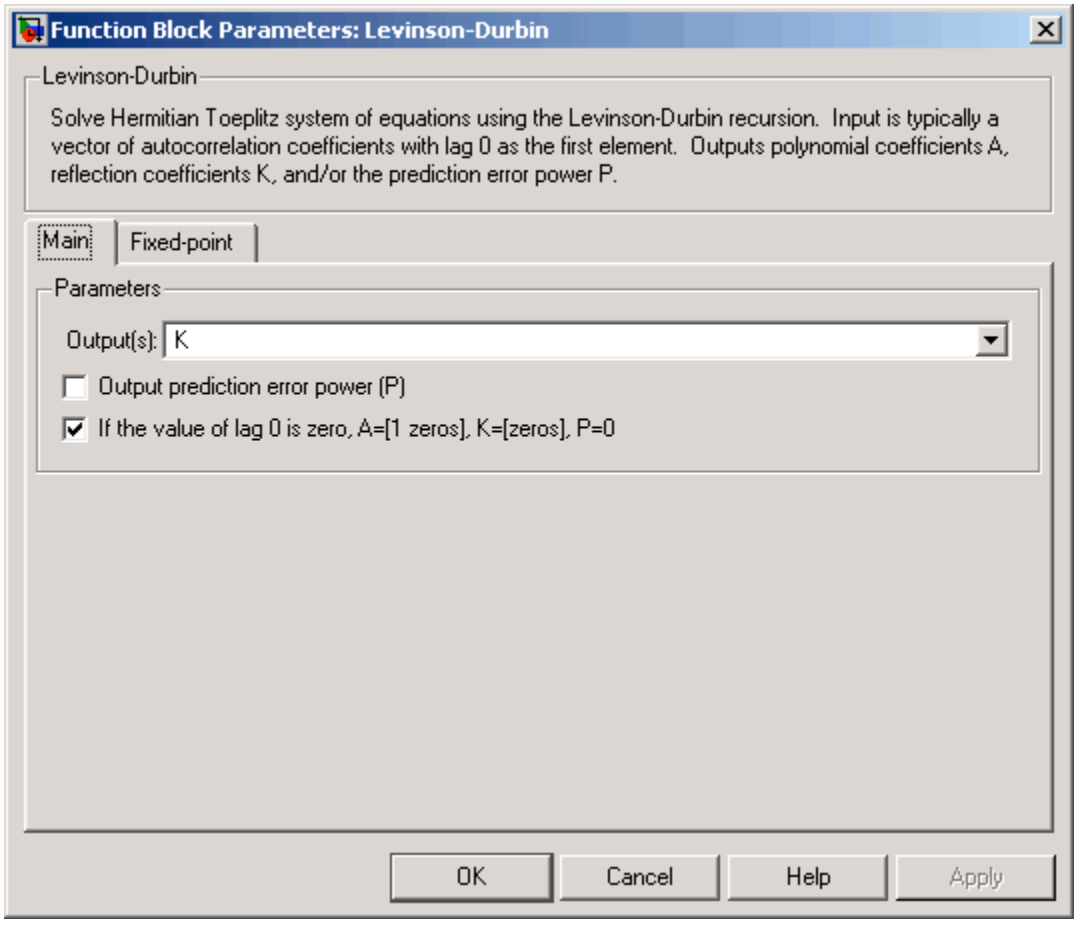


Algorithm

The algorithm requires $O(n^2)$ operations, and is thus much more efficient for large n than standard Gaussian elimination, which requires $O(n^3)$ operations.

Dialog Box

The **Main** pane of the Levinson-Durbin block dialog appears as follows:



Output(s)

Specify the solution representation of R*a* = b to output: model coefficients (`A`), reflection coefficients

(`K`), or both (`A and K`). For scalar inputs, this parameter must be set to `A`.

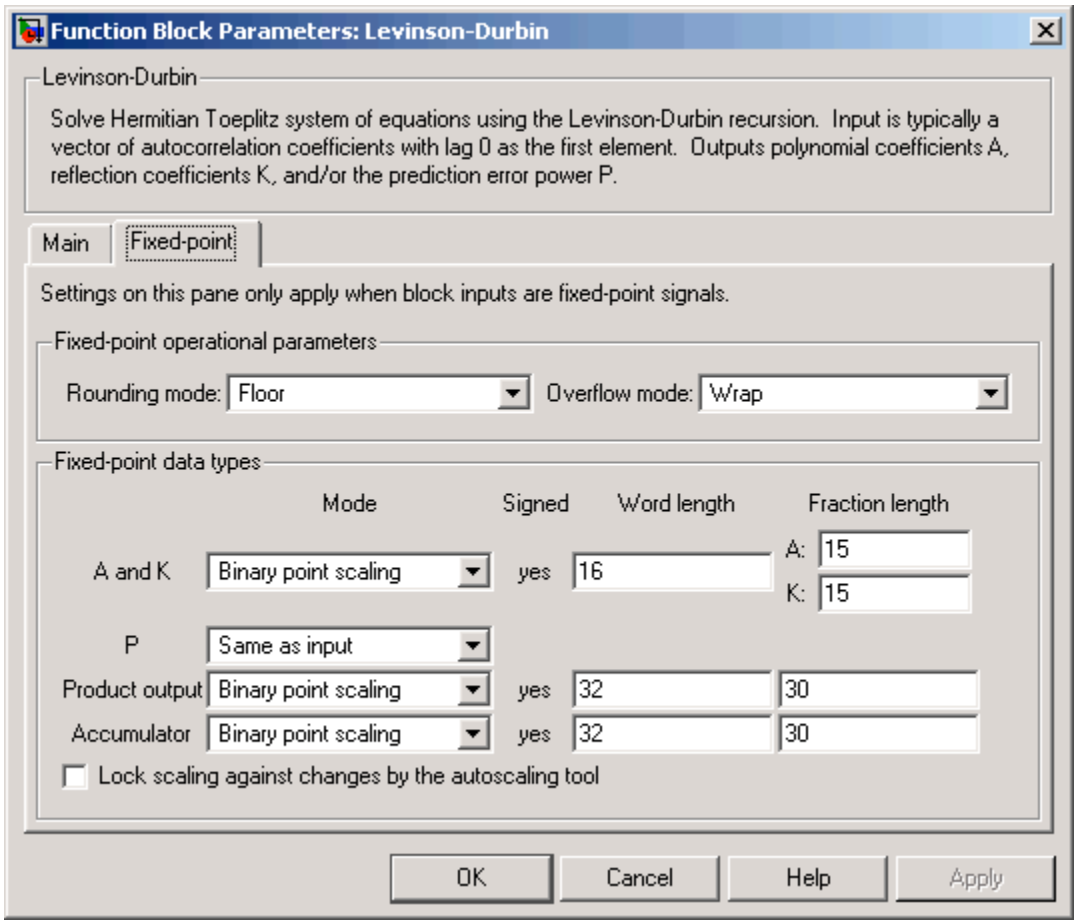
Output prediction error power (P)

Select to output the prediction error at port `P`.

If the value of lag 0 is zero, A=[1 zeros], K=[zeros], P=0

Set to output a zero-vector for inputs having `r(1) = 0`. Otherwise, the block outputs `NaN`s for these inputs.

The **Fixed-point** pane of the Levinson-Durbin block dialog appears as follows:



Rounding mode

Select the rounding mode for fixed-point operations.

Overflow mode

Select the overflow mode for fixed-point operations.

A

Use this parameter to designate how you would like to specify the word and fraction lengths of the polynomial coefficients (*A*). Refer to Fixed-Point Data Types for illustrations depicting the use of the polynomial coefficients data type in this block.

- When you select `Binary point scaling`, you are able to enter the word length and fraction length of *A*, in bits.

- When you select `Slope and bias scaling`, you are able to enter the word length, in bits, and the slope of *A*. This block requires power-of-two slope and a bias of zero.

K

Use this parameter to designate how you would like to specify the word and fraction lengths of the reflection coefficients (*K*). Refer to Fixed-Point Data Types for illustrations depicting the use of the reflection coefficients data type in this block.

- When you select `Binary point scaling`, you are able to enter the word length and fraction length of *K*, in bits.
- When you select `Slope and bias scaling`, you are able to enter the word length, in bits, and the slope of *K*. This block requires power-of-two slope and a bias of zero.

P

Use this parameter to designate how you would like to specify the word and fraction lengths of the prediction error power (*P*). Refer to Fixed-Point Data Types for illustrations depicting the use of the prediction error power data type in this block.

- When you select `Same as input`, these characteristics will match those of the input to the block.
- When you select `Binary point scaling`, you are able to enter the word length and fraction length of *P*, in bits.
- When you select `Slope and bias scaling`, you are able to enter the word length, in bits, and the slope of *P*. This block requires power-of-two slope and a bias of zero.

Product output

Use this parameter to designate how you would like to specify the product output word and fraction lengths. Refer to Fixed-Point Data Types for illustrations depicting the use of the product output data type in this block.

- When you select `Same as input`, these characteristics will match those of the input to the block.
- When you select `Binary point scaling`, you are able to enter the word length and fraction length of the product output, in bits.
- When you select `Slope and bias scaling`, you are able to enter the word length, in bits, and the slope of the product output. This block requires power-of-two slope and a bias of zero.

Accumulator

Use this parameter to designate how you would like to specify the accumulator word and fraction lengths. Refer to Fixed-Point Data Types for illustrations depicting the use of the accumulator data type in this block.

- When you select `Same as product output`, these characteristics will match those of the product output.
- When you select `Same as input`, these characteristics will match those of the input to the block.
- When you select `Binary point scaling`, you are able to enter the word length and fraction length of the accumulator, in bits.
- When you select `Slope and bias scaling`, you are able to enter the word length, in bits, and the slope of the accumulator. This block requires power-of-two slope and a bias of zero.

References

Golub, G. H., and C. F. Van Loan. Sect. 4.7 in *Matrix Computations*. 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

Ljung, L. *System Identification: Theory for the User*. Englewood Cliffs, NJ: Prentice Hall, 1987. Pgs. 278-280.

Kay, Steven M., *Modern Spectral Estimation: Theory and Application*. Englewood Cliffs, NJ: Prentice Hall, 1988.

Supported Data Types

- Double-precision floating point
- Single-precision floating point
- Fixed point (signed only)
- 8-, 16-, and 32-bit signed integers

To learn how to convert your data types to the above data types in MATLAB and Simulink, see Supported Data Types and How to Convert to Them.

See Also

Cholesky Solver	Signal Processing Blockset
LDL Solver	Signal Processing Blockset
Autocorrelation LPC	Signal Processing Blockset
LU Solver	Signal Processing Blockset
QR Solver	Signal Processing Blockset
Yule-Walker AR Estimator	Signal Processing Blockset
Yule-Walker Method	Signal Processing Blockset
`levinson`	Signal Processing Toolbox

See Solving Linear Systems for related information.

Least Squares Polynomial Fit LMS Adaptive Filter