Recursive Algorithms for Online Parameter Estimation

The recursive estimation algorithms in the System Identification Toolbox™ can be separated into two categories:

- Infinite-history algorithms — These algorithms aim to minimize the error between the observed and predicted outputs for all time steps from the beginning of the simulation. The System Identification Toolbox supports infinite-history estimation in:
  - Recursive command-line estimators for the least-squares linear regression, AR, ARX, ARMA, ARMAX, OE, and BJ model structures
  - Simulink® Recursive Least Squares Estimator and Recursive Polynomial Model Estimator blocks
- Finite-history algorithms — These algorithms aim to minimize the error between the observed and predicted outputs for a finite number of past time steps. The toolbox supports finite-history estimation for linear-in-parameters models:
  - Recursive command-line estimators for the least-squares linear regression, AR, ARX, and OE model structures
  - Simulink Recursive Least Squares Estimator block
  - Simulink Recursive Polynomial Model Estimator block, for AR, ARX, and OE structures only

  Finite-history algorithms are typically easier to tune than the infinite-history algorithms when the parameters have rapid and potentially large variations over time.

Recursive Infinite-History Estimation

General Form of Infinite-History Recursive Estimation

The general form of the infinite-history recursive estimation algorithm is as follows:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t))$$

$\hat{\theta}(t)$ is the parameter estimate at time *t*. *y(t)* is the observed output at time *t*, and $\hat{y}(t)$ is the prediction of *y(t)* based on observations up to time *t-1*. The gain, *K(t)*, determines how much the current prediction error $y(t) - \hat{y}(t)$ affects the update of the parameter estimate. The estimation algorithms minimize the prediction-error term $y(t) - \hat{y}(t)$.

The gain has the following form:

$$K(t) = Q(t)\psi(t)$$

The recursive algorithms supported by the System Identification Toolbox product differ based on different approaches for choosing the form of *Q(t)* and computing $\psi(t)$. Here, $\psi(t)$ represents the gradient of the predicted model output $\hat{y}(t|\theta)$ with respect to the parameters θ.

The simplest way to visualize the role of the gradient $\psi(t)$ of the parameters, is to consider models with a linear-regression form:

$$y(t) = \psi^T(t)\theta_0(t) + e(t)$$

In this equation, $\psi(t)$ is the *regression vector* that is computed based on previous values of measured inputs and outputs. $\theta_0(t)$ represents the true parameters. *e(t)* is the noise source (*innovations*), which is assumed to be white noise. The specific form of $\psi(t)$ depends on the structure of the polynomial model.

For linear regression equations, the predicted output is given by the following equation:

$$\hat{y}(t) = \psi^T(t)\hat{\theta}(t-1)$$

For models that do not have the linear regression form, it is not possible to compute exactly the predicted output and the gradient $\psi(t)$ for the current parameter estimate $\hat{\theta}(t-1)$. To learn how you can compute approximation for $\psi(t)$ and $\hat{\theta}(t-1)$ for general model structures, see the section on recursive prediction-error methods in [1].

Types of Infinite-History Recursive Estimation Algorithms

The System Identification Toolbox software provides the following infinite-history recursive estimation algorithms for online estimation:

- Forgetting Factor
- Kalman Filter
- Normalized and Unnormalized Gradient

The forgetting factor and Kalman Filter formulations are more computationally intensive than gradient and unnormalized gradient methods. However, they typically have better convergence properties.

Forgetting Factor. The following set of equations summarizes the *forgetting factor* adaptation algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t))$$

$$\hat{y}(t) = \psi^T(t)\hat{\theta}(t-1)$$

$$K(t) = Q(t)\psi(t)$$

$$Q(t) = \frac{P(t-1)}{\lambda + \psi^T(t)P(t-1)\psi(t)}$$

$$P(t) = \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\psi(t)\psi(t)^T P(t-1)}{\lambda + \psi(t)^T P(t-1)\psi(t)}\right)$$

The software ensures $P(t)$ is a positive-definite matrix by using a square-root algorithm to update it [2]. The software computes `P` assuming that the residuals (difference between estimated and measured outputs) are white noise, and the variance of these residuals is 1. R_2`/2 * P` is approximately equal to the covariance matrix of the estimated parameters, where R_2 is the true variance of the residuals.

Q(t) is obtained by minimizing the following function at time *t*:

$$\sum_{k=1}^{t} \lambda^{t-k}(y(k) - \hat{y}(k))^2$$

See section 11.2 in [1] for details.

This approach discounts old measurements exponentially such that an observation that is τ samples old carries a weight that is equal to λ^τ times the weight of the most recent observation. $\tau = {}^{1}\!/_{1-\lambda}$ represents the *memory horizon* of this algorithm. Measurements older than $\tau = {}^{1}\!/_{1-\lambda}$ typically carry a weight that is less than about 0.3.

λ is called the forgetting factor and typically has a positive value between `0.98` and `0.995`. Set $\lambda = 1$ to estimate time-invariant (constant) parameters. Set $\lambda < 1$ to estimate time-varying parameters.

Note

The forgetting factor algorithm for λ = 1 is equivalent to the Kalman filter algorithm with R_1=0 and R_2=1. For more information about the Kalman filter algorithm, see Kalman Filter.

Kalman Filter. The following set of equations summarizes the *Kalman filter* adaptation algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t))$$

$$\hat{y}(t) = \psi^T(t)\hat{\theta}(t-1)$$

$$K(t) = Q(t)\psi(t)$$

$$Q(t) = \frac{P(t-1)}{R_2 + \psi^T(t)P(t-1)\psi(t)}$$

$$P(t) = P(t-1) + R_1 - \frac{P(t-1)\psi(t)\psi(t)^T P(t-1)}{R_2 + \psi(t)^T P(t-1)\psi(t)}$$

The software ensures $P(t)$ is a positive-definite matrix by using a square-root algorithm to update it [2]. The software computes `P` assuming that the residuals (difference between estimated and measured outputs) are white noise, and the variance of these residuals is 1. R_2* `P` is the covariance matrix of the estimated parameters, and R_1 / R_2 is the covariance matrix of the parameter changes. Where, R_1 is the covariance matrix of parameter changes that you specify.

This formulation assumes the linear-regression form of the model:

$$y(t) = \psi^T(t)\theta_0(t) + e(t)$$

Q(t) is computed using a Kalman filter.

This formulation also assumes that the true parameters $\theta_0(t)$ are described by a random walk:

$$\theta_0(t) = \theta_0(t-1) + w(t)$$

w(t) is Gaussian white noise with the following covariance matrix, or *drift matrix* R_1:

$$Ew(t)w^T(t) = R_1$$

R_2 is the variance of the innovations *e(t)* in the following equation:

$$y(t) = \psi^T(t)\theta_0(t) + e(t)$$

The Kalman filter algorithm is entirely specified by the sequence of data *y(t)*, the gradient $\psi(t)$, R_1, R_2, and the initial conditions $\theta(t=0)$ (initial guess of the parameters) and $P(t=0)$ (covariance matrix that indicates parameters errors).

Note

It is assumed that R_1 and *P*(t = 0) matrices are scaled such that R_2 = 1. This scaling does not affect the parameter estimates.

Normalized and Unnormalized Gradient. In the linear regression case, the gradient methods are also known as the *least mean squares* (LMS) methods.

The following set of equations summarizes the *unnormalized gradient* and *normalized gradient* adaptation algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t))$$

$$\hat{y}(t) = \psi^T(t)\hat{\theta}(t-1)$$

$$K(t) = Q(t)\psi(t)$$

In the unnormalized gradient approach, *Q(t)* is given by:

$$Q(t) = \gamma$$

In the normalized gradient approach, *Q(t)* is given by:

$$Q(t) = \frac{\gamma}{|\psi(t)|^2 + Bias}$$

The normalized gradient algorithm scales the adaptation gain, γ, at each step by the square of the two-norm of the gradient vector. If the gradient is close to zero, this can cause jumps in the estimated parameters. To prevent these jumps, a bias term is introduced in the scaling factor.

These choices of *Q(t)* for the gradient algorithms update the parameters in the negative gradient direction, where the gradient is computed with respect to the parameters. See pg. 372 in [1] for details.

Recursive Finite-History Estimation

The finite-history estimation methods find parameter estimates $\theta(t)$ by minimizing

$$\sum_{k=t-N+1}^{t} (y(k) - \hat{y}(k|\theta))^2,$$

where $y(k)$ is the observed output at time k, and $\hat{y}(k|\theta)$ is the predicted output at time k. This approach is also known as sliding-window estimation. Finite-history estimation approaches minimize prediction errors for the last N time steps. In contrast, infinite-history estimation methods minimize prediction errors starting from the beginning of the simulation.

The System Identification Toolbox supports finite-history estimation for the linear-in-parameters models (AR and ARX) where predicted output has the form $\hat{y}(k|\theta) = \Psi(k)\theta(k-1)$. The software constructs and maintains a buffer of regressors $\psi(k)$ and observed outputs $y(k)$ for $k = t\text{-}N\text{+}1, t\text{-}N\text{+}2, \ldots, t\text{-}2, t\text{-}1, t$. These buffers contain the necessary matrices for the underlying linear regression problem of minimizing $\|\Psi_{buffer}\theta - y_{buffer}\|_2^2$ over θ. The software solves this linear regression problem using QR factoring with column pivoting.

References

[1] Ljung, L. *System Identification: Theory for the User*. Upper Saddle River, NJ: Prentice-Hall PTR, 1999.

[2] Carlson, N.A. "Fast triangular formulation of the square root filter." *AIAA Journal*, Vol. 11, Number 9, 1973, pp. 1259-1265.

[3] Zhang, Q. "Some Implementation Aspects of Sliding Window Least Squares Algorithms." *IFAC Proceedings*. Vol. 33, Issue 15, 2000, pp. 763-768.

See Also

Recursive Least Squares Estimator | Recursive Polynomial Model Estimator | `recursiveAR` | `recursiveARMA` | `recursiveARMAX` | `recursiveARX` | `recursiveBJ` | `recursiveLS` | `recursiveOE`

Related Topics

- What Is Online Estimation?