Nonlinear Local Optimization and Zero-Finding Functions in Matlab

Garey Ramey

University of California, San Diego

November 2018

1 Overview

Matlab provides a suite of built-in functions for use in solving nonlinear optimization and zero-finding problems.

Unconstrained optimization: `fminsearch`, `fminunc`

Constrained optimization: `fminbnd`, `fmincon`

Zero-finding: `fzero`, `fsolve`

Let $f(X, c_1, ..., c_k)$ be the function to be analyzed, where f is real-valued (or vector-valued for `fsolve`), X is a scalar, vector or matrix, and $c_1, ..., c_K$ are Matlab variables (e.g., numerical array, string, cell). f may be expressed as a user-defined m-file:

$$\texttt{function} \quad y = \texttt{myfun}(X, c1, ..., cK)$$
$$y = f(X, c1, ..., cK);$$

The parameter values $c_1, ..., c_K$ are assigned in the Workspace:

$$c1 = c_1\_val; \quad ... \quad cK = c_K\_val;$$

The Matlab solver calculates the solution X_sol by calling `myfun` as part of an anonymous function:

$$[X\_sol,\ f\_val] = \texttt{solver}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ parameters,\ options)\,;$$

where *parameters* is a list of parameters of the solver, *options* is a list of user-selected options governing the solver, and f_val is the value of f evaluated at X_sol.

As an alternative to creating a separate m-file, f may be expressed as an anonymous function within the solver:

$$[X\_sol,\ f\_val] = \texttt{solver}\,(@\,(X)\ f(X, c1, ..., cK),\ parameters,\ options)\,;$$

Moreover, if f has no parameters, then the m-file may be called directly:

$$[X\_sol,\ f\_val] = \texttt{solver}\,(@\texttt{myfun},\ parameters,\ options)\,;$$

2 Local minimization

a Multivariate minimization The solvers `fminsearch` and `fminunc` compute local solutions to problems of the form

$$\min_{x_1,...x_M} f(x_1, ...x_M, c_1, ..., c_K),$$

where the variables $x_1, ..., x_M$ are real scalars.

Let X be a scalar, vector or matrix that represents the M variables; e.g., $X = [x_1\ ...\ x_M]$. As an initial condition, the solvers require a numerical array $X0$ that is conformable with X. The following commands compute a solution using the default options:

$$[X\_sol,\ f\_val] = \texttt{fminsearch}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0)\,;$$

$$[X\_sol,\ f\_val] = \texttt{fminunc}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0)\,;$$

Generally speaking, the algorithms in `fminunc` make use of linear approximations, and are well-suited for smooth functions. On the other hand, `fminsearch` is suited for nonsmooth functions, but it can be slower when there are many variables.

b Univariate minimization with interval constraint The solver `fminbnd` computes local solutions to the problem

$$\min_{x} f(x, c_1, ..., c_K) \quad \text{s.t.} \quad x_1 \leq x \leq x_2,$$

where x_1, x and x_2 are scalars such that $x_1 < x_2$.

The solver requires the values of x_1 and x_2 to be input as parameters. The solution with default options is computed by

$$[x\_sol,\ f\_val] = \texttt{fminbnd}\,(@\,(x)\ \texttt{myfun}(x, c1, ..., cK),\ x1, x2)\,;$$

The algorithm in `fminbnd` is well-suited for nonsmooth functions.

c Multivariate minimization with linear constraints The solver `fmincon` computes local solutions to the problem

$$\min_{X} f(X, c_1, ..., c_K),$$

$$\text{s.t.} \quad AX \leq B, \quad CX = D, \quad X1 \leq X \leq X2,$$

where X is an $M \times 1$ vector of variables, A and B are $P \times M$ and $P \times 1$ matrices that determine P inequality constraints, C and D are $Q \times M$ and $Q \times 1$ matrices that determine Q equality constraints, and $X1$ and $X2$ are $M \times 1$ vectors that determine interval constraints for each variable.

The general syntax under the default options is

$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0, A, B, C, D, X1, X2)\,;$$

where $X0$ is an $M \times 1$ vector of initial values.

Parameters for unneeded constraints are either omitted or replaced with empty matrices, depending on their position in the heirarchy. For example, if the constraints $CX = D$ and $X1 \leq X \leq X2$ are unneeded, then the paramaters C, D, $X1$ and $X2$ are omitted:

$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \ \texttt{myfun}(X, c1, ..., cK),\ \ X0, A, B)\,;$$

On the other hand, if the constraints $AX \leq B$ and $CX = D$ are unneeded, then the parameters A, B, C and D are replaced by empty matrices:

$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \ \texttt{myfun}(X, c1, ..., cK),\ \ X0, [\,], [\,], [\,], [\,], X1, X2)\,;$$

The variable X can be an $M \times N$ matrix, but for computation `fmincon` handles it using 1-D indexing; i.e., X is implicitly converted to $X(:)$, and `fmincon` handles the expressions AX and CX as $A * X(:)$ and $C * X(:)$, respectively. Thus, A and C must have MN columns that are conformable with $X(:)$.

The algorithms in `fmincon` are well-suited for smooth functions.

d Multivariate minimization with nonlinear constraints `fmincon` can also be used for problems with nonlinear constraints:

$$\min_X f(X, c_1, ..., c_K),$$

$$\text{s.t.} \quad g(X, c_1, ..., c_K) \leq 0, \quad h(X, c_1, ..., c_K) = 0,$$

where g and h are vector-valued functions that determine the constraints.

To use this feature, first express g and h as a user-defined m-file:

```
function   [G, H] = myconstr(X, c1, ..., cK)
      G = g(X, c1, ..., cK);
      H = h(X, c1, ..., cK);
```

To compute a solution, the constraint function is input to `fmincon` as a parameter:

$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \ \texttt{myfun}(X, c1, ..., cK),\ \ X0, [\,], [\,], [\,], [\,], [\,], [\,],\ \ldots$$
$$@\,(X)\ \ \texttt{myconstr}(X, c1, ..., cK))\,;$$

This command suppresses the inputs for the linear constraints, and calculates a local minimum subject to the nonlinear constraints $G \leq 0$ and $H = 0$, where $[G,\ H] = \texttt{myconstr}(X, c1, ..., cK)$.

Linear and nonlinear constraints may be combined. For example, the following command combines the constraints $CX = D$ with the inequality constraints:

$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0, [\,], [\,], C, D, [\,], [\,],\ \ldots$$
$$@\,(X)\ \texttt{myconstr}(X, c1, ..., cK))\,;$$

3 Zero-finding

a Single equation The solver `fzero` computes a solution to the problem

$$f(x, c_1, ..., c_K) = 0,$$

where the variable x is a real scalar.

The solution with default options is computed by

$$[x\_sol,\ f\_val] = \texttt{fzero}\,(@\,(x)\ \texttt{myfun}(x, c1, ..., cK),\ x0)\,;$$

If $x0$ is a scalar, then `fzero` uses it as an initial point. If $x0$ is a vector of length 2 such that $f(x0(1), c_1, ..., c_K)$ and $f(x0(2), c_1, ..., c_K)$ differ in sign, then `fzero` searches for a zero within the interval having $x0(1)$ and $x0(2)$ as endpoints. `fzero` returns NaN if a solution cannot be found.

b Systems of equations The solver `fsolve` computes a solution to the problem

$$f_1(x_1, ...x_M, c_1, ..., c_K) = 0,$$
$$\vdots$$
$$f_N(x_1, ...x_M, c_1, ..., c_K) = 0,$$

where the variables $x_1, ..., x_M$ are real scalars.

Let X be a vector or matrix that represents the M variables. The system may be defined as a vector- or matrix-valued function; e.g.,

$$
\begin{aligned}
&\texttt{function} \quad F = \texttt{myfun}(X, c1, ..., cK) \\
&\qquad F = [f_1(X, c1, ..., cK); \\
&\qquad\qquad\qquad \vdots \\
&\qquad\qquad f_N(X, c1, ..., cK)];
\end{aligned}
$$

The following command computes a solution using the default options:

$$[X\_sol,\ F\_val] = \texttt{fsolve}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0)\,;$$

where the initial condition $X0$ is a vector or matrix that is conformable with X.

4 Options

a Stopping criteria The optimization and zero-finding solvers carry out calculations recursively from a given initial point $X_0 = X0$ according to

$$X_i = \Psi(X_{i-1} : f(\cdot), c_1, ..., c_K),$$

where Ψ is the formula used by the solver. The recursions are stopped, and the current values $X\_sol = X_i$ and $f\_val = f(X_i, c_1, ..., c_K)$ are output, when a stopping criterion is met. Stopping criteria are governed by the options `TolX`, `TolFun`, `MaxIter` and `MaxFunEvals`.

`TolX` sets a stopping criterion based on the change in X_i at each iteration:

$$\|X_i - X_{i-1}\| < \texttt{TolX}\ \ (1 + \|X_{i-1}\|)\,.$$

where $\|\cdot\|$ is the Euclidean norm. `TolFun` sets an additional stopping criterion based on the change in $f_i = f(X_i, c_1, ..., c_K)$ at each iteration:

$$\|f_i - f_{i-1}\| < \texttt{TolFun}\ \ (1 + \|f_{i-1}\|)\,.$$

`TolX` and `TolFun` may be set to any positive scalar.

`MaxIter` stops the recursion at iteration $i =$ `MaxIter`. `MaxFunEvals` stops the recursion after `MaxFunEvals` function evaluations have been performed. For example, if each iteration entails n function evaluations, then the recursion is stopped at iteration $i =$ `MaxFunEvals`$/n$. `MaxIter` and `MaxFunEvals` may be set to any positive integer.

b Matlab Toolbox solvers The solvers `fminsearch`, `fminbnd` and `fzero` are part of the Matlab Toolbox, and their options are controlled by the function `optimset`. In addition to the options listed in the preceding subsection, the following options are available:

`Display` - Controls display of solver output.

- `'off'` - no output displayed.
- `'iter'` - display output at each iteration (not available for `lsqnonneg`).
- `'final'` - display final output only.
- `'notify'` - display output only if the solver does not converge.

`FunValCheck` - Checks whether the values of f are valid.

- `'on'` - display error when f returns a valye that is complex or `NaN`.
- `'off'` - display no error.

`OutputFcn` - User-defined function that is called at each iteration.

- `@myfun`

`PlotFcn` - User-defined or built-in plot function called at each iteration.

- `@myfun`
- `@optimplotx` - plots the current point.
- `@optimplotfval` - plots the current function value.

`@optimplotfunccount` - plots the current function count (not available for `fzero`).

The following table lists the available options for the three solvers:

fminsearch	fminbnd	fzero
TolX	TolX	TolX
TolFun	MaxIter	Display
MaxIter	MaxFunEvals	FunValCheck
MaxFunEvals	Display	OutputFcn
Display	FunValCheck	PlotFcn
FunValCheck	OutputFcn	
OutputFcn	PlotFcn	
PlotFcn		

`fminsearch` uses `TolX` and `TolFun` as a joint stopping stopping criterion; i.e., it stops the recursion when both criteria are satisfied.

User-defined option settings are created using `optimset`. The syntax is:

$$\texttt{myopt} = \texttt{optimset}\,('option1', value1, ..., 'optionJ', valueJ)\,;$$

This command creates a Matlab structure called `myopt` that sets the options *option*1, ... *optionJ* to the values *value*1, ...*valueJ*, leaving the other options at their default values. Inputting `myopt` causes the Matlab function to use the specified settings:

$$[X\_sol,\ f\_val] = \texttt{solver}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ parameters,\ \texttt{myopt})\,;$$

For example, the following commands restrict `fminbnd` to a maximum of n iterations:

$$\texttt{myopt} = \texttt{optimset}\,('\texttt{MaxIter}', n)\,;$$
$$[x\_sol,\ f\_mn] = \texttt{fminbnd}\,(@\,(x)\ \texttt{myfun}(x, c1, ..., cK),\ x1, x2,\ \texttt{myopt})\,;$$

Default options for the three functions may be viewed using `optimset`. The syntax is:

$$\texttt{optimset}\,(\texttt{'solver'})\,;\quad \text{or} \quad \texttt{optimset}\,(@\texttt{solver})\,;$$

Options having the value [ ] are either not set as defaults, or are unavailable for the solver considered.

c Optimization Toolbox solvers The solvers `fminunc`, `fmincon` and `fsolve` are part of the Optimization Toolbox, and their options are controlled by the function `optimoptions`. The eight previously discussed options are available for these three solvers, and many more options are also available. In particular, the solvers allow for a choice of numerical algorithm.

The syntax for creating user-defined options settings for these solvers is

$$\texttt{myopt} = \texttt{optimoptions}\,(\texttt{'solver'},\ \textit{'option1'},\ \textit{value}1, ...\textit{'optionJ'},\ \textit{valueJ})\,;$$

or

$$\texttt{myopt} = \texttt{optimoptions}\,(@\texttt{solver},\ \textit{'option1'},\ \textit{value}1, ...,\textit{'optionJ'},\ \textit{valueJ})\,;$$

For example, the following commands set $\texttt{TolX} = q$ in `fmincon`:

$$\texttt{myopt} = \texttt{optimoptions}\,(\texttt{'fmincon'},\ \texttt{'TolX'},\ q)\,;$$
$$[X\_sol,\ f\_val] = \texttt{fmincon}\,(@\,(X)\ \texttt{myfun}(X, c1, ..., cK),\ X0, A, B, [\,], [\,], [\,], [\,], [\,],\ \texttt{myopt})\,;$$

Default options for the three functions may be viewed using

$$\texttt{optimoptions}\,(\texttt{'solver'})\,;\quad \text{or} \quad \texttt{optimoptions}\,(@\texttt{solver})\,;$$

For a list of functions in the Optimization Toolbox together with links to the available options for each function, type `doc optimoptions` in the Command Window and scroll down to "Name-Value Pair Arguments"..

5 Production economy example

a Problem specification Consider a one-period economy in which goods 1 and 2 are produced using capital and labor. The economy is endowed with $\bar{k}$ and $\bar{l}$ units of capital and labor, respectively. Production functions for the two goods are given by

$$f_i(k_i, l_i) = A_i k_i^{\alpha_i} l_i^{1-\alpha_i}, \quad i = 1, 2,$$

and the social welfare function is

$$W(y_1, y_2) = \ln y_1 + \gamma \ln y_2.$$

The optimal allocation is the solution to the following problem:

$$\begin{aligned} &\max_{y_1, y_2, k_1, k_2, l_1, l_2} W(y_1, y_2), \\ \text{s.t.} \quad & y_i, k_i, l_i \geq 0, \quad y_i \leq A_i k_i^{\alpha_i} l_i^{1-\alpha_i}, \quad i = 1, 2, \\ & k_1 + k_2 \leq \bar{k}, \quad l_1 + l_2 \leq \bar{l}. \end{aligned} \tag{1}$$

b Nonlinear programming solution with inequality constraints

Let the variables be represented as a 2×3 matrix:

$$X = \begin{bmatrix} y_1 & k_1 & l_1 \\ y_2 & k_2 & l_2 \end{bmatrix}.$$

The nonnegativity and feasibility constraints can be expressed as

$$\texttt{eye}(6) * X(:) \geq \texttt{zeros}(6, 1),$$

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0; \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} * X(:) \leq \begin{bmatrix} k\_\texttt{bar}; \\ l\_\texttt{bar} \end{bmatrix}.$$

`fmincon` correctly implements these constraints for the parameters

$$A = \begin{bmatrix} & & -\texttt{eye}(6); & & & \\ 0 & 0 & 1 & 1 & 0 & 0; \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \texttt{zeros}(6, 1); \\ k\_\texttt{bar}; \\ l\_\texttt{bar} \end{bmatrix}.$$

Next define the function

$$\texttt{function}\quad [G,\ H] = \texttt{prodconstr}(X, A1, A2, \texttt{alph1}, \texttt{alph2})$$
$$G = [X(1,1) - A1 * X(1,2)\hat{}\texttt{alph1} * X(1,3)\hat{}(1 - \texttt{alph1});$$
$$X(2,1) - A2 * X(2,2)\hat{}\texttt{alph2} * X(2,3)\hat{}(1 - \texttt{alph2})];$$
$$H = 0;$$

After assigning values for the parameters $A1$, $A2$,`alph1`,`alph2` and `gam`, the solution can be computed using the command

$$X\_sol = \texttt{fmincon}\,(@\,(X)\ \ -(\texttt{log}(X(1,1)) + \texttt{gam} * \texttt{log}(X(2,1))),\ X0, A, B,\ [\,],\ [\,],\ \ldots$$
$$[\,],\ [\,], @\,(X)\ \texttt{prodconstr}(X, A1, A2, \texttt{alph1}, \texttt{alph2}))\,;$$

c Nonlinear programming solution with equality constraints

Alternatively, the production and resource constraints can be treated as equality constraints. In this case, define the m-file

$$\texttt{function}\quad [G,\ H] = \texttt{prodconstr\_eq}(X, A1, A2, \texttt{alph1}, \texttt{alph2})$$
$$G = 0;$$
$$H = [A1 * X(1,2)\hat{}\texttt{alph1} * X(1,3)\hat{}(1 - \texttt{alph1}) - X(1,1);$$
$$A2 * X(2,2)\hat{}\texttt{alph2} * X(2,3)\hat{}(1 - \texttt{alph2}) - X(2,1)];$$

and the parameters

$$C = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0; \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},\quad D = \begin{bmatrix} k\_\texttt{bar}; \\ l\_\texttt{bar} \end{bmatrix},$$
$$LB = \texttt{zeros}(2,3).$$

The solution is computed by

$$X\_sol = \texttt{fmincon}\,(@\,(X)\ \ -(\texttt{log}(X(1,1)) + \texttt{gam} * \texttt{log}(X(2,1))),\ X0,\ [\,],\ [\,], C, D,\ \ldots$$
$$LB,\ [\,], @\,(X)\ \texttt{prodconstr\_eq}(X, A1, A2, \texttt{alph1}, \texttt{alph2}))\,;$$

d Solution using first-order conditions

The Lagrangian for problem (1) is

$$\mathcal{L} = \ln y_1 + \gamma \ln y_2 + \lambda_1 \left(A_1 k_1^{\alpha_1} l_1^{1-\alpha_1} - y_1 \right)$$
$$+\lambda_2 \left(A_2 k_2^{\alpha_2} l_2^{1-\alpha_2} - y_2 \right) + \mu_k(\bar{k} - k_1 - k_2) + \mu_l(\bar{l} - l_1 - l_2).$$

First-order necessary conditions for a solution include the equations

$$\frac{1}{y_1} = \lambda_1, \quad \frac{\gamma}{y_2} = \lambda_2,$$
$$\lambda_1 \alpha_1 A_1 k_1^{\alpha_1 - 1} l_1^{1-\alpha_1} = \mu_k = \lambda_2 \alpha_2 A_2 k_2^{\alpha_2 - 1} l_2^{1-\alpha_2},$$
$$\lambda_1 (1-\alpha_1) A_1 k_1^{\alpha_1} l_1^{-\alpha_1} = \mu_l = \lambda_2 (1-\alpha_2) A_2 k_2^{\alpha_2} l_2^{-\alpha_2},$$

which may be rearranged as follows:

$$\frac{\alpha_1}{\alpha_2} \frac{k_2}{k_1} = \frac{(1-\alpha_1)}{(1-\alpha_2)} \frac{l_2}{l_1} = \gamma, \tag{2}$$
$$y_1 = A_1 k_1^{\alpha_1} l_1^{1-\alpha_1}, \quad y_2 = A_2 k_2^{\alpha_2} l_2^{1-\alpha_2}. \tag{3}$$

The solution is determined by (2), (3) and the resource constraints:

$$k_1 + k_2 = \bar{k}, \quad l_1 + l_2 = \bar{l}.$$

For this case, define the m-file

```
function   F = FOC(X, A1, A2, alph1, alph2, gam)
    F = [(alph1 * X(2,2))/(alph2 * X(1,2)) - gam;
        ((1 - alph1) * X(2,3))/((1 - alph2) * X(1,3)) - gam;
        X(1,1) - A1 * X(1,2)^alph1 * X(1,3)^(1 - alph);
        X(2,1) - A2 * X(2,2)^alph2 * X(2,3)^(1 - alph2);
        X(1,2) + X(2,2) - k_bar;
        X(1,3) + X(2,3) - 1_bar]
```

The solution is computed by

```
X_sol = fsolve(@(x) FOC(X, A1, A2, alph1, alph2, gam), X0);
```

4 Maximum likelihood estimation example

Consider the following linear model:

$$y_i = \alpha + \sum_{j=1}^{J} \beta_j x_{ji} + \varepsilon_i, \quad i = 1, ..., N,$$

where $y_i, x_{i1}, ..., x_{iJ}$ are observations, $\alpha, \beta_1, ..., \beta_J$ are parameters, and ε_i are mutually uncorrelated normal random variables with zero mean and variance σ^2.

a Estimation using `fminsearch`

The likelihood function of $\alpha, \beta_1, ..., \beta_J, \sigma^2$ of the model given the observations is

$$\mathcal{L} = \left(\frac{1}{2\pi\sigma^2}\right)^{N/2} \exp\left(-\frac{1}{2\sigma}\sum_{i=1}^{N} \varepsilon_i^2\right)$$

$$= \left(\frac{1}{2\pi\sigma^2}\right)^{N/2} \exp\left(-\sum_{i=1}^{N}\left(y_i - \alpha - \sum_{j=1}^{J} \beta_j x_{ji}\right)^2\right)^{1/2\sigma}.$$

The maximum likelihood estimates $\hat{\alpha}, \hat{\beta}_1, ..., \hat{\beta}_J$ are the solutions to the following problem:

$$\max_{\alpha,\beta_1,...,\beta_p} \exp\left(-\sum_{i=1}^{N}\left(y_i - \alpha - \sum_{j=1}^{J} \beta_j x_{ji}\right)^2\right). \tag{4}$$

Let the observations and parameters be expressed as

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{J1} \\ 1 & x_{12} & \cdots & x_{J2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1N} & \cdots & x_{JN} \end{bmatrix},$$

$$\texttt{bet} = [\ \alpha \quad \beta_1 \quad \cdots \quad \beta_J$$

Create the m-file:

```
function  x = lik_fn(bet, Y, X)
x = exp(-sum(Y - X * bet).^2);                :
```

For given data matrices Y and X of size $N \times 1$ and $N \times J$, respectively, the solution is calculated by

$$\texttt{bet\_hat} = \texttt{fminsearch}\,(@\,(x) \quad -\texttt{lik\_fn}(x, Y, X), \;\; \texttt{zeros}(J+1, 1))\,;$$

Alternatively, the objective function can be defined within `fminsearch`:

$$\texttt{bet\_hat} = \texttt{fminsearch}\,(@\,(x) \;\; \texttt{exp}(-\texttt{sum}(Y - X * x).\hat{}2), \;\; \texttt{zeros}(J+1, 1))\,;$$

b Estimation using OLS

Problem (4) is equivalent to

$$\min_{\alpha, \beta_1, \ldots, \beta_p} \left(y_i - \alpha - \sum\nolimits_{j=1}^{J} \beta_j x_{ji} \right)^2 .$$

The solution is given by the standard OLS estimator, which can be calculated by

$$\texttt{bet\_hat} = (X' * X) \backslash (X' * Y);$$