The Nonlinear Systems Toolbox is a package of MATLAB .m files for the analysis and synthesis of nonlinear control systems described by polynomials. Some of these .m files are implemented by .mex files. Therefore the first step is to mex and link these .mex files.

There are eight .m files that are implemented by .mex files. They are:

- \([h, nh]=\text{cmp}(f, nf, df, g, ng, dg, d)\)
- \([h, nh]=\text{dd}(f, nf, df, g, ng, dg, d)\)
- \([h, nh]=\text{jcbn}(f, nf, df, nj, d)\)
- \([h, nh]=\text{mply}(f, nf, df, g, ng, dg, d)\)
- \(z=\text{mon}(x, n, d)\)
- \(c=\text{chuze}(n, k)\)
- \(m=\text{crd}(n, d)\)
- \(m=\text{crdsum}(n, [d0 \ d1])\) or \(m=\text{crdsum}(n, [d0 \ d1])\)

See the how2mex___ .m files in the ToBeMexed folder.

CMP, DD and MPLY accept complex fields, that is, polynomial fields with complex coefficients, but they require that \(F\) and \(G\) both be real or both be complex.
% Contents
% Set up Routine
%
% TAY_POLY Computes the Taylor series of a symbolic vector field
%
% Functions for controller design
%
% CH_CRDS Computes term by term changes of state and output coordinates and feedback on a system.
% DSP Computes term by term a dissipative feedback law.
% FBI Computes term by term the solution of the Francis-Byrnes-Isidori PDE of nonlinear regulation.
% FBK_LIN Computes term by term an approximately linearizing coordinate changes and feedback.
% FH2F_H_ Computes term by term a change of coordinates and feedback which approximately transforms one nonlinear system to another.
% HJB Computes the stationary solution of a parameterized nonlinear optimal control problem via the term by term solution of the Hamilton-Jacobi-Bellman PDE.
% HJI Computes the stationary solution of a parameterized nonlinear differential game via the term by term solution of the Hamilton-Jacobi-Isaacs PDE.
% INV_MFD Computes term by term an invariant manifold.
% MDL_MTCH Computes term by term an optimal model matching feedforward and feedback control law.
% ZBV Computes term by term a Lyapunov function.
% Utility Functions
%%

% CHUZE Computes binomial coefficients.
% CMP Composes polynomial vector fields.
% CMP_INV Computes the inverse of a polynomial vector field
% under composition.
% COLSUM Computes the column sums of a matrix.
% CRD Computes the number of homogeneous monomials of
% given degree.
% CRDSUM Computes the number of homogeneous monomials of
% lower degree to higher degree.
% DD Computes the directional derivative of one polynomial
% vector field by another.
% JCBN Computes the Jacobian of a polynomial vector field.
% MON Computes the value of the monomials of a vector.
% MPLY Computes the product of a polynomial matrix field and
% a polynomial vector or matrix field.
% PRT Extracts terms of desired degrees from
% a polynomial vector field.
% SHOWF Displays dynamics in a labeled format.
% SHOWWH Displays the output map in a labeled format.
The basic data type is a vector field which is polynomial of arbitrary degree in a vector which is composed of an arbitrary number of subvectors. Of course, memory and speed limitations will implicitly restrict the degrees of the vector fields and dimensions of vectors. NST also works with polynomial matrix fields, polynomial higher order tensor fields and complex fields.

Consider a nonlinear system of the form

\[
\begin{align*}
\dot{x} &= f(x,u) = f_0 + f_1(x,u) + f_2(x,u) + f_3(x,u) + \ldots + f_d(x,u) \\
y &= h(x,u) = h_0 + h_1(x,u) + h_2(x,u) + h_3(x,u) + \ldots + h_d(x,u)
\end{align*}
\]

where \( f_0 \) and \( h_0 \) are \( nx1 \) and \( px1 \) vectors, \( f_1(x,u) \) and \( h_1(x,u) \) are \( nx1 \) and \( px1 \) vector fields, linear in the \( nx1 \) state vector \( x \) and the \( mx1 \) input vector \( u \), \( f_2(x,u) \) and \( h_2(x,u) \) are quadratic in \( x \) and \( u \), and \( f_3(x,u) \) and \( h_3(x,u) \) are cubic in \( x \) and \( u \), and \( f_d(x,u) \) and \( h_d(x,u) \) are homogeneous polynomials of degree \( d \) in \( x \) and \( u \). The output vector \( y \) is \( px1 \).

The linear part of the system,

\[
\begin{align*}
f_1(x,u) &= a*x + b*u \\
h_1(x,u) &= c*x + d*u
\end{align*}
\]

is stored in the computer as the \( n \) by \( (n+m) \) and \( p \) by \( (n+m) \) matrices

\[
\begin{align*}
f_1 &= [a \ b] \\
h_1 &= [c \ d]
\end{align*}
\]
% A word about notation. On some machines, Matlab is not
% case-sensitive so this version of the Nonlinear_System_Toolbox
% uses only lower case identifiers in executable statements.
% In particular, to describe a linear system, we use a,b,c,d
% instead of the more standard A,B,C,D. Greek letters are
% abbreviated to the first two letters of their name, e.g.
% al for alpha, be for beta. The one exception is pi which is
% the constant in MATLAB. We use py instead.
% The Nonlinear_System_Toolbox also follows the standard Matlab
% convention of capitalizing identifiers when they appear in
% comments within script files, so in comments, A,B,C,D refer to
% the matrices a,b,c,d appearing in executable statements.
% However in this file which consists solely of comments, we
% henceforth adopt a hybrid convention, identifiers in displays
% will not be capitalized but will look exactly as they do in
% executable statements. Identifiers appearing in the body of the
% text will be capitalized as they are in comments.
The above nonlinear system consists of the two vector fields which are stored as the two matrices of coefficients of the monomials of \( X \) and \( U \), in order of increasing degree.

\[
\begin{align*}
\mathbf{f} &= [f_0 \ f_1 \ f_2 \ f_3 \ldots \ f_d] \\
\mathbf{h} &= [h_0 \ h_1 \ h_2 \ h_3 \ldots \ h_d]
\end{align*}
\]

The monomials are in block lexicographic order described below.

The number of distinct monomials of degree \( D \) in the \( N \times 1 \) vector \( X \) and the \( M \times 1 \) vector \( U \) is

\[
\binom{n+m+d-1}{d}
\]

This is readily computed by calling \( \text{crd}(n+m,d) \) or equivalently \( \text{chuze}(n+m+d-1,d) \).

The number of distinct monomials of degrees 0 thru \( D \) in \( X \) and \( U \) is the same as the number of distinct monomials of degree \( D \) in \( W \) where \( W \) is a scalar variable, the constant 1.

The number is

\[
\binom{n+m+d}{d}
\]

Therefore the matrix \( F \) is \( N \) by \( \text{CRD}(N+M+1,D) \) and \( H \) is \( P \) by \( \text{CRD}(N+M+1,D) \). Given the numerical values of the matrices \( F,H \) and the vectors \( X,U \), we obtain the values of the vector fields \( f(x,u) \) and \( h(x,u) \) by multiplying the matrices \( F \) and \( H \) by a column vector composed of all the distinct monomials of degrees 0 thru \( D \) in \( X \) and \( U \).
These monomials should be in block lexicographic order which we now describe.

The monomials of a single vector $X$ to degree three in lexicographic order are

$$[1; x; \{x\}^2; \{x\}^3]$$

The notation "\{ \}~" does not exist in Matlab.

For the $n$ vector

$$x=[x(1) \ x(2) \ldots x(n)]'$$

it is defined as follows

$$\{x\}^2= [x(1)*x(1) \ x(1)*x(2) \ldots x(1)*x(n) \ldots$$
$$x(2)*x(2) \ldots x(2)*x(3) \ldots x(n)*x(n)]'$$

$$\{x\}^3= [x(1)*x(1)*x(1) \ x(1)*x(1)*x(2) \ldots x(1)*x(1)*x(n) \ldots$$
$$x(1)*x(2)*x(2) \ x(1)*x(2)*x(3) \ldots x(1)*x(n)*x(n) \ldots$$
$$x(2)*x(2)*x(2) \ldots x(2)*x(2)*x(n) \ldots x(n)*x(n)*x(n)]'$$

Higher powers are defined similarly so that the indices to the right change faster than those to the left.

When $X$ is $N$ by 1 vector, the function

$$\text{mon}(x,n,[dl \ du])$$

returns the column vector

$$[\{x\}^dl; \ldots ; \{x\}^du]$$
The monomials of the composite vector \([X;U]\) to degree three in block lexicographic order are

\[
\begin{align*}
[1; x; u; \{x\}^2; \{x\}^*\{u\}; \{u\}^2; \ldots \\
\{x\}^3; \{x\}^*\{u\}; \{x\}^*\{u\}^2; \{u\}^3
\end{align*}
\]

The notation "\([ \}^\-\)" is as above and the notation "\([ \}^*\{ \}^\-\)"

is as follows for

\[
\begin{align*}
x &= [x_1 \ldots x_n]', \\
\{x\}^*\{u\} &= [x(1)*u(1) \ldots x(1)*u(m) \ x(2)*u(1) \ldots x(n)*u(m)]', \\
\{x\}^*\{u\}^2 &= [x(1)*u(1)*u(1) \ x(1)*u(1)*u(2) \ldots x(1)*u(m)*u(m)]. \\
\end{align*}
\]

Notice that the monomials are in order of increasing degree,

within each degree they are in blocks of the same degree in \(X\)

and the same degree in \(U\), and within the blocks the indices to

the right change faster than those to the left. We use the

notation \(\{X;U\}^D\) to denote the homogeneous monomials of degree

\(D\) in \([X;U]\) and the notation \(\{1;X;U\}^D\) to denote the monomials

in \([X;U]\) from degree 0 to \(D\). For example,

\[
\begin{align*}
\{x;u\} &= [x;u] \\
\{x;u\}^2 &= [\{x\}^2; \{x\}^*\{u\}; \{u\}^2] \\
\{x;u\}^3 &= [\{x\}^3; \{x\}^*\{u\}; \{x\}^*\{u\}^2; \{u\}^3] \\
\end{align*}
\]
% The monomials of the composite vector \([X;U;Y]\) to degree three
% in block lexicographic order are
% \[
{x;u;y}= [x;u;y]
\]
% \[
{x;u;y}^2 = [{x}^2; {x}*{u;y}; {u;y}^2]
\]
% \[
{x;u;y}^3 = [{x}^3; {x}^2*{u;y}; {x}*{u;y}^2; {u;y}^3]
\]
% \[
1; x; u; y}^3 = [{x}^3; {u;y}^3]
\]
% In general, if \(X_1, X_2, \ldots X_K\) are column vectors, the monomials
% of the composite vector \([X_1;X_2;\ldots;X_K]\) to degree three in block
% lexicographic order are defined recursively
% \[
{1;x_1;\ldots;x_k}^3 = [1; x_1; {x_2;\ldots;x_k}];\ldots
\]
% \[
{x_1}^2;\ldots
\]
% \[
{x_1}*{x_2;\ldots;x_k};\ldots
\]
% \[
{x_2;\ldots;x_k}^2;\ldots
\]
% \[
{x_1}^3;\ldots
\]
% \[
{x_1}^2*{x_2;\ldots;x_k};\ldots
\]
% \[
{x_1}*{x_2;\ldots;x_k}^2;\ldots
\]
% \[
{x_2;\ldots;x_k}^3]
\]
% The function
% \[
\text{mon}([x_1;x_2;\ldots;x_k],[n_1;\ldots;n_k],[d_1 d_2])
\]
% computes the monomials of degrees DL thru DU in block
% lexicographic order in \([X_1;X_2;\ldots;X_K]\), which are column vectors
% of dimensions, \([N_1;\ldots;N_K]\).
% For example
% \[
{1;x_1;\ldots;x_k}^3
\]
% is computed by
% \[
\text{mon}([x_1;x_2;\ldots;x_k],[n_1;\ldots;n_k],[0 3])
\]
% If the above nonlinear system is of degree 3, then the cubic
% vector field \( f(x,u) \) evaluated at \([X;U]\) is
% \[
% f*\text{mon}(\{x;u\},\{n;m\},[0 3])
% \]
% If \( f(x,u) \) has no constant term, \( F=[F_1 F_2 F_3] \), then the
% evaluation at \([X;U]\) is given by
% \[
% f*\text{mon}(\{x;u\},\{n;m\},[1 3])
% \]

% The matrix \( F \) can be thought of as representing a field \( f(x,u) \)
% which is polynomial of degrees 0 thru 3 in \([X;U]\) or can be
% thought of as representing a field \( f(wun,x,u) \) which is
% homogeneous of degree 3 in \([WUN;X;U]\). When \( WUN=1 \), the values
% are the same, i.e.
% \[
% f*\text{mon}(\{x;u\},\{n;m\},[0 3])=f*\text{mon}(\{1;x;u\},[1;n;m],[3 3])
% \]
% The submatrices of F that contain the coefficients of the
% homogeneous parts of the field and can be extracted by
% using the function PRT(F,NF,DF,D) where NF contains the
% dimensions of F, DF the degrees of F and D contains the degrees
% to be extracted.

% Associated to each vector field, F, is a pair of matrices,
% NF and DF, containing the dimensions and degrees of F.
% The degree matrix, DF, is 1x2,
%     df=[df(1,1) df(1,2)].
% The first entry, DF(1,1) is the degree of the lowest terms in F
% and the second, DF(1,2), is the degree of the highest terms in
% The matrix F must contain all the coefficients for all monomials
% from degree DF(1,1) to DF(1,2). This may require some zero fill
% If the above nonlinear system is of degree 3 then
%     df = [0 3]
% If F does not contain constant terms,
%     xdot = f(x,u) = f1(x,u) + f2(x,u) + f3(x,u)
% then
%     f = [f1 f2 f3]
%     df = [1 3].
% Notice that when homogenizing a vector field F whose lower
% degree is not 0, the coefficients for the missing degrees from
% 0 to DF(1,1)-1 must be added. These coefficients are all zero.
% The dimension matrix, NF, contains information about the
% dimensions of the value and argument of F. For vector fields
% NF has two columns, for matrix fields NF has 3 columns.
% The value of a vector field F is a column vector which can be
% composed of subvectors. The dimensions of these subvectors are
% stored in the first column of the dimension matrix, NF. The
% argument of a vector field F is a column vector which can be
% composed of subvectors. The dimensions of these subvectors are
% stored in the last column of NF. The rest of NF is zero
% filled. If F is a matrix field then the first two columns of
% NF contain the the row and column dimensions of sub-blocks of
% the value of F and the last column contains the dimensions of
% the subvectors in the argument of F.
%
% The information in NF is needed by the routines that
% compose two vector fields, CMP. that differentiate one vector
% field in the direction of another, DD, that multiply a matrix
% field and a vector field, MLPY, and that compute the Jacobian
% of a vector field, JCBN. See below
% For the above nonlinear system
% nf = [n n 0 m]
% nh = [p n 0 m]
% If we viewed [f(x,u); h(x,u)] as a single vector field
% represented by the matrix
% fh = [f; h]
% then
% nfh = [n n p m]
%
% Occasionally it is desirable to view the subvectors in the
% value and the argument of a vector field as subvectors of a
% single supervector. We denote this through the dimension
% matrix. For example, we can consider the vector
% field [f(x,u); h(x,u)] as a mapping from [x; u; y] to
% [x; u; y] defined by [x; u; y] → [f(x,u); 0; h(x,u)].
% This mapping is represented by the matrix
% fh = [f; h]
% with dimensions
% nfh = [n 0 p; n m 0]’
% The 0 in the first column opposite the M in the second column
% indicates the M missing rows of fh. The 0 in the second
% column opposite the P in the first indicates many missing
% columns of FH, the coefficients of those monomials in [x; u; y]
% containing a factor from y. As needed, FH will be filled with
% rows and/or columns of zeros based on the information in nfh.
% Notice in contrast with missing intermediate degree terms which
% need to be accounted for by filling with zero coefficients,
% missing subvectors in the value and argument do not require
% filling in with zero coefficients but rather are denoted
% through the dimension matrix.
We can consider the vector field \( f(x,u) \) as a mapping from \([x;u]\) to \([x;u]\) (more precisely, to \([xdot;udot]\)) defined by \([x;u] \rightarrow [f(x,u);0]\). If we wished to differentiate \( h(x,u) \) in the direction \( f(x,u) \), we should consider the argument and value of \( f \) and the argument of \( h \) as parts of a supervector, \([x;u]\). So

\[
\begin{align*}
\text{nf} &= [n 0; n m]' \\
\text{nh} &= [p 0; n m]'
\end{align*}
\]

Since the value of \( h \) need not be part of the supervector \([x;u]\), we could also let

\[
\text{nh} = [0 p; n m]'
\]

but since the value of \( f \) is the first subvector of \([x;u]\), we could not let

\[
\text{nf} = [0 n; n m]'
\]

We can also let the supervector be \([x;u;y]\) in which case

\[
\begin{align*}
\text{nf} &= [n 0 0; n m 0]' \\
\text{nh} &= [0 0 p; n m 0]'
\end{align*}
\]

The directional derivative of \( h(x,u) \) in the direction \( f(x,u) \) is computed by

\[
\text{dd}(h, nh, dh, f, nf, df, d)
\]

The last parameter \( D = [D(1,1) \ D(1,2)] \) contains the lower and upper degrees of the directional derivative to be computed.
Occasionally, we consider vector fields depending on external parameters such as
\[ f(x,u,x_,u_) = f_0 + f_1(x,u,x_,u_) + \ldots + f_d(x,u,x_,u_) \]
where \( X_ \) and \( U_ \) are \( N_ \) by 1 and \( M_ \) by 1 vectors.

The matrix of coefficients
\[ f = [ f_0 \ f_1 \ f_2 \ f_3 \ldots \ f_d ] \]
is an \( N \) by \( \text{CRD}(N+M+N_+M_+1,D) \) matrix. The column order corresponds to that of the monomials in \([X;U;X_;U_]\) in block lexographic order. The dimensions and degrees of \( F \) are
\[ n_f = [ n \ 0 \ 0 \ 0; n \ m \ n_ \ m_ \ ]' \]
\[ d_f = [ 0 \ d ] \]

Suppose \( H \) is an \( N \) by \( \text{CRD}(N+M+1,D) \) matrix representing
\[ h(x,u) = h_0 + h_1(x,u) + h_2(x,u) + h_3(x,u) + \ldots + h_d(x,u) \]
\[ h = [ h_0 \ h_1 \ h_2 \ h_3 \ldots \ h_d ] \]
If we wished to compute the directional derivative of \( h(x,u) \)
in the direction \( f(x,u,x_,u_) \), we would call
\[ \text{dd}(h,nh,dh,f,nf,df,d) \]
with
\[ nh = [ p \ 0 \ 0 \ 0; n \ m \ 0 \ 0 ]' \]
\[ dh = [ 0 \ d ] \]
In summary when calling \( \text{DD}(H,NH,DH,F,NF,DF,D) \)
the number of rows in \( NH \) and \( NF \) must be the same and for each row, the nonzero entries in the last column of \( NH \) and the first and second columns of \( NF \) must agree.

For more information, type HELP DD.
% If we wished to form the composition of $h(x,u)$ following
% $f(x,u,x_-,u_-)$ (more precisely, $[f(x,u,x_-,u_-);0]$) we call
% 
% $\text{cmp}(h,nh,dh,f,nf,df,d)$
% 
% For CMP the value of $F$ and the argument of $H$ must be the same.
% This means that the number of rows in NH and NF must be the
% same and for each row, the nonzero entries in the last column
% of NH and the first column of NF must agree.
% For more information, type HELP CMP.
% If we wished to form the Jacobian of h(x,u) with respect to x
% we call
%    jcbn(h,nh,dh,nj,d)
% with
%    nh= [p 0;n m]'
%    nj= [n 0]'
% The number of rows in NH and NJ must be the same and for each
% row, the nonzero entries in the last column of NH and the
% column NJ must agree. The Jacobian is computed with respect to
% those subvectors with nonzero entries in NJ. To compute the
% Jacobian with respect to both X and U, let
%    nj= [n m]'
% If we called
%    jcbn(h,nh,dh,nj,d)
% with
%    nh= [p 0 0 0;n m 0 0]'
%    nj= [n 0 n_ 0]'
% then H would be zero-filled to represent the vector field
% h(x,u,x_) and the Jacobian with respect X and X_ is computed.
% Because both NH(4,2) and NJ(4,1) are zero, H is not zero-filled
% to represent the vector field h(x,u,x_,u_).
% The Jacobian of a vector field is a matrix field which is
% row-stacked and transposed to make a long vector field. Row
% stacking is used because of our standing convention of ordering
% multi-indexed quantities so that the indices on the left move
% slower than those on the right.
% For more information, type HELP JCBN.
Vector fields are distinguished from matrix fields by their dimension matrix. As we have discussed, the dimension matrix of a vector field has two columns, the dimensions of the subvectors of the value are in the first column and the dimensions of the subvectors of the argument are in the second column.

The dimension matrix of a matrix field has three columns. The value of a matrix field can be composed of submatrices. The row dimensions of the submatrices are in the first column of the dimension matrix and the column dimensions of the submatrices are in the the second column of the dimension matrix. The third column contains the dimensions of the argument of the matrix field.

The function MPLY multiplies a polynomial matrix field with another polynomial matrix field or a polynomial vector field.
% A tensor field can be treated as a long vector field by reducing
% the multiple index associated with the value of the field to a
% single index through lexicographic ordering. The standard
% convention is to let the indices to the right change faster
% than those to the left. This is a generalizes the way
% tensor fields of order two, i.e., matrix fields are treated.

% The basic functions DD, CMP, JCBN accept tensor fields
% wherever they make mathematical sense.
% In particular, JCBN takes a
% tensor field of order k to one of order k+1.
% The dimension matrix of a tensor field of order k has k+1
% columns. The value of a tensor field of order k is indexed by
% a k-tuple of indices. The first index of the k-tuple ranges
% over the components of subvectors whose dimensions are in the
% first column of the dimension matrix, the second index of
% the k-tuple ranges over the components of subvectors whose
% dimensions are in the second column of the dimension matrix,
% and the k-th index of the k-tuple ranges over the components of
% subvectors whose dimensions are in the k-th column of the
% dimension matrix. The last column of the dimension matrix
% contains the dimensions of the argument of the tensor field.
This file describes how to create MEX files on a MAC for the following functions:

- \( [h, nh] = \text{cmp}(f, nf, df, g, ng, dg, d) \)
- \( [h, nh] = \text{dd}(f, nf, df, g, ng, dg, d) \)
- \( [h, nh] = jcbn(f, nf, df, nj, d) \)
- \( [h, nh] = mply(f, nf, df, g, ng, dg, d) \)
- \( c = \text{chuze}(n, k) \)
- \( m = \text{crd}(n, d) \)
- \( m = \text{crdsum}(n, d0, d1) \) or \( m = \text{crdsum}(n, [d0 d1]) \)
- \( z = \text{mon}(x, n, d) \)
% First create a directory for the files that will be needed.
% You can give this directory any name that you want; in this
% description we’ll assume that it’s called "ToBeMexed".
% Start MATLAB 7.04 and make it your working directory.
% Make sure that the following source files are in the directory
% Makefile  addprod.c  choose.c  chuze.c
%    cmp.c    compose.c    crd.c    crdsum.c
%    dd.c    jacobian.c    jcbn.c    mexdefin
%    mply.c    sqz.ext.c    walkmonom.c    walkredm
%
% At the MATLAB prompt type the following commands following each
% carriage return
% >> mex chuze.c choose.c
% >> mex crd.c choose.c
% >> mex crdsum.c choose.c
% >> mex cmp.c sqz.ext.c compose.c choose.c walkmonom.c addprod.
% >> mex dd.c sqz.ext.c addprod.c addprodzplx.c jacobian.c choose
% >> mex jcbn.c jacobian.c choose.c walkmonom.c sqz.ext.c walkre
% >> mex mon.c choose.c walkredmonom.c walkmonom.c
% >> mex mply.c addprod.c choose.c walkmonom.c walkredmonom.c sq

% This causes MATLAB to compile and link the c programs,
% creating the MEX files. Afterwards you should find in the dir
% ToBeMexed in addition to the source files listed above,
% there should be 8 new files, whose names are something like
%  
% chuze.mexmac    cmp.mexmac    crd.mexmac    crd
% dd.mexmac       jcbn.mexmac   mply.mexmac
% mon.mexmac

% As a test, try typing "chuze(6,2)". MATLAB should give the ans
function [ka,fk,py,lk]=hjb(f,l,n,m,d,f_,n_)

% [KA,FK,PY,LK]=HJB(F,L,N,M,D,F_,N_)
% computes the stationary solution of a
% parametrized nonlinear optimal control problem via the term b
% term solution of the Hamilton-Jacobi-Bellman (HJB) equation u
% to feedback terms of degree D.
% E. G. Al’brekht presented the term by term method for solving
% the unparametrized Hamilton-Jacobi-Bellman (HJB) equation in
% This generalization is due to A. J. Krener.
The input parameters are as follows.

F is the state dynamics,

\[ X' = F(X, U, X_\_). \]

This is an N vector field, polynomial of degrees 1 thru D in the vector \([X; U; X_\_]\) whose subvectors are the state X, the control U and the dynamic parameter \(X_\_), which are of dimensions N, M and N_ respectively.

The coefficients of F corresponding to the monomials of degree zero in X and U must be zero, i.e., \(F(X, U, X_\_)\) is \(O(X, U)\).

L is the Lagrangian (running cost),

\[ L(X, U, X_\_). \]

This is a scalar field, polynomial of degrees 2 thru D+1 in the vector \([X; U; X_\_]\). The coefficients of L corresponding to the monomials of degrees zero and one in X and U must be zero.

In other words, \(L(X, U, X_\_)\) is \(O(X, U)^2\).

D is the desired degree of the optimal feedback.
\% F_ is the dynamics of the parameters,
\% \quad X_\' = F_ (X_).
\% This is an N_ vector field, polynomial of degrees 1 thru D in
\% the vector [X_].
\% If the system does not depend on parameters, then
\% F_ and N_ can be ommitted in the call of HJB.
\% Then they are set to the default values F_ = [], N_ = 0.
The output parameters are as follows.

KA, the optimal feedback,

\[ U = KA(X, X_\_), \]

degrees 1 thru D

\[ KA(X X_\_) \text{ is } O(X). \]

FK, the optimal closed loop system, degrees 1 thru D

\[ X' = FK(X, X_\_) = F(X, KA(X X_\_), X_\_). \]

PY, the optimal cost

\[ PY(X, X_\_), \text{ degrees 2 thru D+1,} \]

\[ PY(X, X_\_) \text{ is } O(X)^2. \]

LK, the optimal running cost, degrees 2 thru D+1,

\[ LK(X, X_\_) = L(X, KA(X X_\_), X_\_) \]

\[ LK(X, X_\_) \text{ is } O(X)^2. \]
if nargin==5
    f_= [];
    n_= 0;
end

% First we compute the quadratic cost PY2(X) and linear feedback
% U=K*X by using LQR2 from the Control Systems Toolbox.

disp(sprintf('Computing the HJB feedback of degree %-2.0f',1))

a= f(:,1:n);
b= f(:,n+1:n+m);
ii=0;
q=zeros(n,n);
for i=1:n
    ii=ii+1;
    q(i,i)=2*l(1,ii);
    for j=i+1:n
        ii=ii+1;
        q(i,j)=l(1,ii);
        q(j,i)=l(1,ii);
    end %j
end % i
for i=1:n
    for j=1:m
        ii=crd(n,2)+(i-1)*(m+n_)+j;
        s(i,j)=l(1,ii);
    end
end
ii=crd(n,2)+n*(m+n_);
r=zeros(m,m);
for i=1:m
  ii=ii+1;
  r(i,i)=2*l(1,ii);
  for j=i+1:m
    ii=ii+1;
    r(i,j)=l(1,ii);
    r(j,i)=l(1,ii);
  end %j
end %i

[k,p] = lqr2(a,b,q,r,s);
k= -k;
ka = [k zeros(m,n_)];
nka = [0 m 0; n 0 n_]’;
inka = [1 1];
py=zeros(1,crd(n+n_,2));
ii=0;
for i=1:n
    ii=ii+1;
    py(1,ii)=p(i,i)/2;
    for j=i+1:n
        ii=ii+1;
        py(1,ii)=p(i,j);
    end %j
end %i
npy = [1,n;0,0;0,n_];
dpy = [2 2];
ri= inv(r);
ak= a+b*k;
ae=[ak zeros(n,n_); zeros(n_,n) f_(:,1:n_)];
% We compute the Jacobians DFDU and DLDU of F and L with respect
nf = [n 0 0; n m n_]’;
df = [1 d];
nl = [1 0 0; n m n_]’;
dl = [2 d+1];
nj = [0 m 0]’;
[dfdu,ndfdu] = jcbn(f,nf,df,nj,[0 d-1]);
[dldu,ndldu] = jcbn(1,nl,dl,nj,[1 d]);
We modify the F, L, DFDU and DLDU by the linear feedback and obtain functions of \([X;X_;U_].\)

\[
\begin{align*}
\text{ph} &= \begin{bmatrix} \text{eye}(n) & \text{zeros}(n,n_); & \text{ka} & \text{zeros}(n_,n) & \text{eye}(n_) \end{bmatrix}; \\
n\text{ph} &= \begin{bmatrix} n & m & n_; & n & 0 & n_ \end{bmatrix}'; \\
d\text{ph} &= \begin{bmatrix} 1 & 1 \end{bmatrix}; \\
d\text{fk} &= \begin{bmatrix} 1 & 2 \end{bmatrix}; \\
[fk,nfk] &= \text{cmp}(f,nf,df,ph,nph,dph,dfk); \\
d\text{lk} &= \begin{bmatrix} 2 & 3 \end{bmatrix}; \\
dlk &= \text{cmp}(l,nl,dl,ph,nph,dph,dlk); \\
dfduk &= \text{cmp}(dfdu,ndfdu,[0 \ d-1],ph,nph,dph,[0 \ 1]); \\
ddlduk &= \text{cmp}(dldu,ndldu,[1 \ d],ph,nph,dph,[1 \ 2]); \\
ddfduk &= \begin{bmatrix} 0 & 1 \end{bmatrix};
\end{align*}
\]
% We convert $F_-$ from a function of $X_-$ into a function of $[X;X_-]$ using the standard projection $TH$.

```
th = [zeros(n_, n) eye(n_)];
nth = [0 0 n_; n 0 n_]';
[f_, nf_] = cmp(f_, [0 0 n_; 0 0 n_]', df, th, nth, 1, df);
nfkf_ = [n 0 n_; n 0 n_]';
```

% We compute the degree $J+1$ cost $PYJ1(X,X_-)$ and degree $J$ feedback $KAJ(X,X_-)$

```
for j = 2:d
  disp(sprintf('Computing the HJB feedback of degree %2.0f', j))
i2 = crd(n_, j+1) + n*crd(n_, j);
i1 = crd(n+n_, j+1) - i2;
temp = [eye(i1) zeros(i1, i2)];
mat = dd(temp, [i1 0 0; n 0 n_]', j+1, ae, [n 0 n_; n 0 n_]', 1, j+1);
mat = mat(:, 1:i1);
fkf_ = [fk; prt(f_, nf_, df, dfk)];
temp = dd(py, npy, dpy, fkf_, nfkf_, dfk, j+1)....
    + prt(1k, nlk, dlk, j+1);
temp = temp(:, 1:i1);
pyj1 = -temp / mat;
clear mat
pyj1 = [pyj1 zeros(1, crd(n_, j+1) + n*crd(n_, j))];
py = [py pyj1];
dpy = [2 j+1];
[temp, ntemp] = jcbn(py, npy, dpy, [n 0 0]', [1 j]);
temp = mply(temp, ntemp, [1 j], dfduk, ndfduk, ddfduk, j)....
    + prt(dlduk, ndlduk, dfk, j);
ka = -rinv*temp;
ka = [ka kaj];
```
% We modify the F, L, DFDU and DLDU by the feedback to degree J
% and obtain functions of [X;X_;U_].

j1= crd(n+n_,j);
ph= [ph [zeros(n,j1);kaj;zeros(n_,j1)]];
dph= [1 j];
dfk= [1 min(j+1,d)];
[fk,nfk]= cmp(f,nf,df,ph,nph,dph,dfk);
dlk= dfk+[1 1];
[lk,nlk]= cmp(l,nl,dl,ph,nph,dph,dlk);
if j<d,
    ddfduk= dfk-[1 1];
    [ddfduk,ndfduk]= cmp(ddfduk,ndfduk,[0 d-1],ph,nph,dph,ddfduk);
    [dlduk,ndlduk]= cmp(1,dl,d,ph,nph,dph,dlk);
end
end

% All rights reserved.

% rinv2 error corrected 4/6/07.

% Cross term in lagrangian added to call of lqr2.m 10/2/13
% Corrected definition of s
% U_ option dropped 3/31/15.
function [th,la,thx_u_,ff,hh]=fbi(f,h,n,m,p,deg,f_,h_,n_,m_,th1,l

% [TH,LA,TH_X_U,FF,HH]=FBI(F,H,N,M,P,DEG,F_,H_,N_,M_,TH1,LA1)
% computes the term by term solution of the
% generalized Francis-Byrnes-Isidori (FBI) PDE for nonlinear
% regulation, model matching. This generalization can be found
% Krener, A. J., Optimal model matching controllers for linear
% and nonlinear systems, Nonlinear Control Systems Design 1992,
% It is based on earlier work of Francis, Isidori-Byrnes
% and Huang-Rugh. See above for citations. This routine is al
% used by FL to compute an approximate feedback linearization.
%
The input parameters are as follows.

F is the plant dynamics,

\[ X' = F(X, U, X_-, U_-) \]

This is an N vector field, polynomial of degrees 1 thru DEG in the vector \([X; U; X_-; U_-]\) whose subvectors are the plant state \(X\), the plant input \(U\), the model state \(X_-\), and the model input \(U_-\) which are of dimensions \(N, M, N_-\) and \(M_-\), respectively.

H is the plant output map,

\[ Y = H(X, U, X_-, U_-) \]

where \(Y\) is P dimensional.

H is an P vector field, polynomial of degrees 1 thru DEG in the vector \([X; U; X_-; U_-]\).
% F_ is the model dynamics,
% X'_-=F_(X_,U_).
% This is an N_ vector field, polynomial of degrees 1 thru DEG
% the vector [X_;U_].
% H_ is the model output map,
% Y_-=H_(X_,U_) where Y_ is also P
% dimensional. H_ is an P vector field, polynomial of degrees 1
% thru DEG in the vector [X_;U_].
% Notice the model dynamics does not depend on X,U.
% DEG is the degree of the data and the desired solution.
% TH1 and LA1 are the linear parts of the solution, which
% can be prespecified. If they are absent, FBI will compute the
% TH1, LA1 are NxN_, Mx(N_+M_) respectively.
FBI computes, to degree DEG, a submanifold of $X, X_\_ \text{ space}$

and a feedforward

such that the submanifold is invariant under the combined pla
and model dynamics and the error $E = Y - Y_\_$ is zero on the
submanifold. This requires that $TH$ and $LA$ be a solution of th
generalized Francis-Byrnes-Isidori (FBI) PDE

Following Huang-Rugh, these equations are solved term by term
up to degree DEG

At each degree, these reduce to linear equations for next
coefficients of $TH$ and $LA$ which depend on the previously foun
coefficients of lower degrees. Generally, the number of
equations and the number of unknowns are not the same.
They are the same if $M = P$ and $M_\_ = 0$.

If the equations are not solvable or if there are several
solutions, a least squares solution is found using MATLAB’s
pseudoinverse function PINV. A warning is displayed and the
least squares solution is used in later calculations as if it
were the true solution.
THX_U_ is TH zero-filled to be a function of [X_; U_].
FBI also computes, to degree DEG, the result of the changes of coordinates

\[
    Z = X - \text{TH}(X_-)
\]
\[
    E = Y - Y_-
\]
and feedforward
\[
    V = U - \text{LA}(X_-, U_-)
\]
on F and H and returns them as FF and HH, which are Nx1 and Px1 fields in [X; U; X_; U_].
a = f(:,1:n);
b = f(:,n+1:n+m);
aa_ = f(:,n+m+1:n+m+n_);
bb_ = f(:,n+m+n_-1:n+m+n_-m_-);
a_ = f_(:,1:n_);
b_ = f_(:,n_-1:n_-m_-);
if p==0
    c = [];
d = [];
    cc_ = [];
    dd_ = [];
c_ = [];
d_ = [];
else
    c = h(:,1:n);
d = h(:,n+1:n+m);
    cc_ = h(:,n+m+1:n+m+n_-);
    dd_ = h(:,n+m+n_-1:n+m+n_-m_-);
c_ = h(:,1:n_-);
d_ = h(:,n_-1:n_-m_-);
end %if
if nargin==10 % compute the linear submanifold X = TH1*X_
    % and the linear feedforward U= LA1*[X_;U_].
    disp('Solving the FBI equations of degree 1')
    mat= zeros(n*n_+m*n_+m*m_,(n+p)*(n+m_));
    mat(1:n*n_+m*n_,1:(n+p)*n_)= kron([a b;c d]',eye(n_));
    -kron([eye(n) zeros(n,m);zeros(p,n+m)'],a_);
    mat(:,(n+p)*n_+1:(n+p)*(n_+m_))= ....
    [-kron([eye(n) zeros(n,m);zeros(p,n+m)'],b_); ....
    kron([b;d]',eye(m_))];
    z= [reshape([-aa_;c_-cc_]',1,(n+p)*n_)...
        reshape([-bb_;d_-dd_]',1,(n+p)*m_)];
    thla1rs= z*pinv(mat);
    res= z- thla1rs*mat;
    if max(max(abs(res)))>0.000001
        disp('Warning! Cannot exactly solve the FBI eqns at degree 1')
        disp('will use the least squares solution, residue of size...')
        disp(max(max(abs(res))))
    end %if;
    th1= reshape(thla1rs(1,1:n*n_),n_,n)';
    la1= reshape(thla1rs(1,n*n_+1:(n+m_)*n_),n_,m)';
    la1= [la1 reshape(thla1rs(1,(n+m_)*n_+1:(n+m_)*n_+m*m_),m_,m)'];
    clear mat z thla1rs
end

th= th1;
thx_u_= [th1 zeros(n,m_)];
la= la1;
if deg==1 return; end
% We compute the degree 2 parts of F(TH1(X_), LA1(X_, U_), X_, U_)
% and H(TH1(X_), LA1(X_, U_), X_, U_).
nf = [n 0 0 0; n m n_ m_]';
nh = [p 0 0 0; n m n_ m_]';
df = [1 deg];
ph = [thx_u_; la; eye(n_+m_)];

% We multiply TH1 by the degree two part of F_ and extract the
% degree two part of H_.
nf_ = [0 0 n_ 0; 0 0 n_ m_]';
nh_ = [p 0 0 0; 0 0 n_ m_]';

% We compute the degree 2 parts of the resulting expressions.

for j=2:deg
    % We compute the degree J terms of the solution to FBI equations
    disp(sprintf('Solving the FBI equations of degree %-2.0f',j))
    n_j= crd(n_,j);
    n_m_j= crd(n_+m_,j);
    temp= eye(n_j);
    temp1= dd(temp,[n_j n_],j,a_,[n_ n_],1,j);
    mat= kron(a',eye(n_j))-kron(eye(n),temp1);
    mat= [mat zeros(n*n_j,n*(n_m_j-n_j))];
    mat= [mat; kron(b',eye(n_m_j))];
    mat= [mat [kron(c',eye(n_j)) zeros(n*n_j,p*(n_m_j-n_j));
                 kron(d',eye(n_m_j))]];
    temp= thf_j - ftlj;
    temp1= h_j - htlj;
    z= [reshape(temp',1,n*n_m_j) reshape(temp1',1,p*n_m_j)];
    thlajrs= z*pinv(mat);
    res= z- thlajrs*mat;
    if max(max(abs(res)))>0.000001
        disp(sprintf('Warning! Cannot exactly solve the FBI eqns at degree %-2.0f',j))
        disp('will use the least squares solution, residue of size...')
        disp(max(max(abs(res))))
    end %if;
    thj= reshape(thlajrs(1,1:n*n_j),n_j,n);
    laj= reshape(thlajrs(1,n*n_j+1:n*n_j+m*n_m_j),n_m_j,m);
    th= [th thj];
    thx_u_= [thx_u_ thj zeros(n,n_m_j - n_j)];
    la= [la laj];
clear mat z thlajrs temp temp1 thj laj
if \( j < \text{deg}, \)
\[
\begin{align*}
\text{ph} &= [\text{thx}_u; \ la; \ \text{eye}(n_+m_+), \ \text{zeros}(n_+m_, \text{crd}(n_+m_+1, j)-(n_+m_+1))]; \\
n\text{ph} &= [n \ m \ n_+m_; \ 0 \ 0 \ n_+m_]'; \\
d\text{ph} &= [1 \ j]; \\
\text{ftlj} &= \text{cmp}(f, n\text{f}, df, \text{ph}, n\text{ph}, d\text{ph}, j+1); \\
\text{htlj} &= \text{cmp}(h, nh, df, ph, nph, dph, j+1); \\
n\text{thf}_j &= \text{dd}(\text{th}, nth, dph, f_, n\text{f}_-, df, j+1); \\
\text{h}_j &= \text{prt}(h_, nh_, df, j+1);
\end{align*}
\]
end
end
k = crd(n+m+n_+m_+1,deg)-(n+m+n_+m_+1);
ph = [eye(n+m+n_+m_) zeros(n+m+n_+m_,k)];
thal = cmp(thx_u_,[n 0 0;0 0 n_ m_]',df,...
    [zeros(n_+m_,n+m) eye(n_+m_)],[0 0 n_ m_;n m n_ m_]',1,df);
ph(1:n,:) = ph(1:n,:) + thall;
laall = cmp(la,[0 m 0;0 0 n_ m_]',df,...
    [zeros(n_+m_,n+m) eye(n_+m_)],[0 0 n_ m_;n m n_ m_]',1,df);
ph(n+1:n+m,:) = ph(n+1:n+m,:) + laall;
nph = [n m n_ m_;n m n_ m_]';
ff = cmp(f,nf,df,ph,nph,df,df);
hh = cmp(h,nh,df,ph,nph,df,df);
temp = dd(th,nth,df,f_,nf_,df,df);
ff = ff - cmp(temp,[n 0 0;0 0 n_ m_]',df,...
    [zeros(n_+m_,n+m) eye(n_+m_)],[0 0 n_ m_;n m n_ m_]',1,df);
hh = hh - cmp(h_,[0 0 0;0 0 n_ m_]',df,...
    [zeros(n_+m_,n+m) eye(n_+m_)],[0 0 n_ m_;n m n_ m_]',1,df);

% Copyright (c) 1995, 2005 by A. J. Krener.
% All rights reserved.
function [h,nh]=dd(f,nf,df,g,ng,dg,d)

% DD Directional derivative of one polynomial vector field by another.
%
% [H,NH] = DD(F,NF,DF,G,NG,DG,D)
% computes the directional derivative of a vector field F, of
% dimensions NF and polynomial of degrees DF(1,1) to DF(1,2) by
% vector field G, of dimensions NG and polynomial of degrees
% DG(1,1) to DG(1,2). The directional derivative is polynomial
% degrees MAX(DF(1,1)-1,0)+DG(1,1) to MAX(DF(1,2)-1,0)+DG(1,2).
% Only the terms from degree D(1,1) to degree D(1,2) are computed
% and returned in H. If these differ from the above then part or
% all of H may be zero-filled to make H polynomial of degrees
% D(1,1) to D(1,2). The dimensions of H are in NH as explained
% below. The degrees DF, DG and D can 1x2 matrices or scalars.
% scalar D is equivalent to [D D]. There is no restriction on
% degrees but DD gets much slower and requires much
% more memory as they increase.
The value of $F$ is a column vector composed of subvectors. The dimensions of these subvectors are in the first column of $N F$. The argument of $F$ is a column vector composed of subvectors. The dimensions of these subvectors are in the second column of $N F$. The dimensions of $G$ and $H$ are in $N G$ and $N H$ and these are the same size as $N F$. The rest of $N F$, $N G$ and $N H$ are filled with zeros.

DD assumes that the argument of $F$, the argument of $G$ and the value of $G$ are all the same vector and, in particular, the component subvectors are in the same order. If not, take as the common vector, a supervector of all three and modify $N F$ and $N G$ as follows. If a particular subvector is absent in the argument of $F$, place a zero instead of its dimension in the second column of $N F$. Do not zero fill $F$. Similarly, if a particular subvector is absent in the value or argument of $G$, place a zero instead of its dimension in the first or second column of $N G$.

The value of $H$ is the same as the value of $F$ while the argument of $H$ is the supervector of the argument of $F$, the argument of $G$ and the value of $G$. The dimensions in $N H$ will reflect this. $F$ can also be a matrix or tensor field.

DD is implemented by DD.MEX
\%
\% The vector fields F and G can be complex but if one is complex then the other must be too. Otherwise MATLAB will crash.
\% If F and G are complex then the output argument H is complex.

\% Copyright (c) 1996, 2005 by A. J. Krener.
\% All rights reserved.
function [h,nh]=cmp(f,nf,df,g,ng,dg,d)

%CMP Composition of polynomial vector fields.
%
% [H,NH]=CMP(F,NF,DF,G,NG,DG,D)
% computes the composition of a vector field F, of
% dimensions NF and polynomial of degrees DF(1,1) to DF(1,2) by
% vector field G, of dimensions NG and polynomial of degrees
% DG(1,1) to DG(1,2). The result is a vector field H, of
% dimensions NG and polynomial of degrees DF(1,1)*DG(1,1)
% to DF(1,2)*DG(1,2).
% Only the terms from degree D(1,1) to degree D(1,2) are comput
% and returned in H. If these differ from the above then part o
% all of H may be zero-filled to make H polynomial of degrees
% D(1,1) to D(1,2). The dimensions of H are in NH as explained
% below. The degrees DF, DG and D can 1x2 matrices or scalars.
% scalar D is equivalent to [D D].
The degrees DF, DG and D can be any nonnegative integers but gets much slower and requires much more memory as they increase.

The value of F is a column vector composed of subvectors. The dimensions of these subvectors are in the first column of NF.

The argument of F is a column vector composed of subvectors. The dimensions of these subvectors are in the second column of NF.

The dimensions of G and H are in NG and NH and these are the same size as NF. The rest of NF, NG and NH are filled with zeros.

CMP assumes that the argument of F and the value of G are the same vector and, in particular, the component subvectors are in the same order. If not, take as the common vector, a supervector of both and modify NF and NG as follows. If a particular subvector is absent in the argument of F, place a zero instead of its dimension in the second column of NF.

Do not zero fill F. Similarly, if a particular subvector is absent in the value of G, place a zero instead of its dimension in the first column of NG. The value of H is the same as the value of F while the argument of H is the argument of G. The dimensions in NH will reflect this.

F can also be a matrix or tensor field.

CMP.M is implemented by CMP.MEX.
The vector fields $F$ and $G$ can be complex but if one is complex then the other must be too. Otherwise MATLAB will crash. If $F$ and $G$ are complex then the output argument $H$ is complex.

Copyright (c) 1996, 2005 by A. J. Krener.
All rights reserved.