MATLAB/MEX/FORTRAN TOOLBOX README

Alexander W. Richter* Nathaniel A. Throckmorton†

August 28, 2013

1 TOOLBOX STRUCTURE

This toolbox is used in Richter et al. (2013). To make use of the toolbox, the directory and its subfolders should be added to the MATLAB path:

1. Extract `MatlabMEXFortranToolbox.zip` and its subfolders.
2. Open MATLAB, click "File", and select "Set Path..." from the drop-down menu.
3. Click "Add with Subfolders...", select the parent folder of the extracted toolbox (i.e., the one that contains the folders Chebyshev, External, Internal, and Linear Interpolation), and click "OK".
4. Click "Save" and then "Close".

The following is an overview of the folders and their contents.

- Linear Interpolation: MATLAB, Fortran, and compiled MEX (Windows 32 and Windows 64) code used to approximate multi-dimensional functions using linear interpolation. We provide two types of Fortran/MEX code—one that uses C-style pointers and one that uses fortran-style pointers. C-style pointers are much faster, but cannot be used when an array is returned to MATLAB (i.e. when there are 3 or more stochastic components). These functions require evenly spaced grid points. Extrapolation is also linear and based on the nearest nodes on the edge of the state space. MATLAB and MEX function are interchangeable with the same inputs. The compiled MEX functions affix the letter F in front of the function name. See section 2 for details on compiling MEX functions. See section 3 for details on using the linear interpolation functions.
- Chebyshev: MATLAB, Fortran, and compiled MEX (Windows 32 and Windows 64) code used to approximate multi-dimensional functions using Chebyshev polynomials. MATLAB and MEX functions are interchangeable with the same inputs. The compiled MEX functions affix the letter F in front of the function name. See section 4 for details on usage.
- Internal: Commonly used MATLAB functions written by Richter and Throckmorton

*Department of Economics, Auburn University, arichter@auburn.edu
†Department of Economics, Indiana University, nathrock@indiana.edu

  - Gauss Hermite Quadrature: `ghquad` computes the weights required for numerical integration using Gauss-Hermite quadrature.
  - Chebyshev: `chebspace` outputs a discretized interval corresponding to the zeros of the Chebyshev polynomials. `chebpoly` outputs the coefficients, powers and zeros of the Chebyshev polynomials.
  - Plotting: `axeslim` adjusts the axes limits (box) on all subplots in a figure. A buffer is chosen between the curves and the box. `plotops` specifies figure properties we typically use in our papers.
  - `itinfo` displays iteration information such as iteration count, the distance between policy function updates, iteration time, and total algorithm time.

- External: Readily available MATLAB functions written by other authors
  - Chris Sims: Includes Sims' `gensys` algorithm for solving linear rational expectations models (requires `qzdiv` and `qzswitch`) and `csolve`, which is an excellent function for solving systems of equations.
  - David Terr: Code used to calculate the coefficients of Hermite (`HermitePoly`) and Chebyshev (`ChebyshevCoeff`) polymials.
  - `savex`: MATLAB function that saves all variables in the workspace *excect* those that the user specifies.

2 COMPILING MEX FUNCTIONS

The following instructions are for Intel Visual Fortran 11.1 (IVF), but other versions may be supported. Please consult the MATLAB version compatibility documentation. Since our Fortran code is in the free format (.f90), the MEX setup batch files need to be altered. Follow these steps:

1. Navigate to the mexopts folder in your MATLAB installation (e.g. . . .\MATLAB\R2011b\bin\win64 \mexopts)
2. Open the batch files corresponding to the installed IVF version and linker in a text editor.
3. Find all '/fixed' flags and delete them. Save the batch files.
4. Then use the command `mex -setup` in MATLAB and select IVF as the compiler. This step must be completed even if the compiler was previously setup.

Use the command `mex *.f90` to compile the f90 code as a MEX function replacing the asterisk with the filename. If successful, MATLAB will create a file with a `*.mex%` extension, where the asterisk is the same filename as the original function and the percent sign signifies the operating system (e.g., win32, win64).

If you have any problems setting up the compiler or compiling the Fortran code, first consult http://www.mathworks.com/matlabcentral/newsreader/. If you discover any bugs, please feel free to contact us directly with specific questions.

3 LINEAR INTERPOLATION

The functions in the linear interpolation folder are used to locally approximate multi-dimensional functions. The functions employ the nearest neighbor method by taking a weighted average of the slopes between the nearest neighbors in each dimension where the weights are the point's relative position. It is assumed that the discretized interval in each dimension is evenly spaced, which decreases computational burden. MATLAB, Fortran and Windows 32- and 64-bit MEX versions are available. The Fortran and Fortran-based MEX functions are prefixed with an `F`. The suffix `c` is added to indicate that the MEX gateway function exploits the speediness of c-style pointers, and also decreases the implementation burden by reducing the number of lines in the gateway function. c-style pointers are faster since it is not necessary to make the inputs allocatable, and the inputs are not subject to size limitations (for example, a vector is limited to 2^{19} elements).

3.1 ALLTERPXYZ

Linearly interpolates or extrapolates an `X`-dimensional function. Evaluates `Y` functions simultaneously. The outputs `o[1:Y]` have `Z` dimensions. For example, if `Z` is 1 then the outputs will be vectors with values corresponding to a vector of points evaluated for the last dimension. If `Z` is 2 then the outputs will be matrices with values over the last two dimensions.

```
Usage:
 [o1,...,oY] = allchebXYZ_T(x1,...,xX, ...
                            z1i,...,xXi, ...
                            pf1,...,pfY)
Inputs:
 x[1:X]    : Evenly-spaced discretized interval (vector)
 x[1:X-Z]i : Point to evaluate for respective variable (scalar)
 z[X-Z+1:X]: Points to evaluate for respective variable (vector)
 A[1:Y]    : Least squares estimates (square matrices)
 T         : Coefficients of Chebyshev polynomials (square matrix)
 P         : Powers of Chebyshev polynomials (square matrix)
Outputs:
 o[1:Y]    : Interpolated/extrapolated values
```

4 CHEBYSHEV INTERPOLATION

The functions in the Chebyshev folder are used to approximate multi-dimensional functions using Chebyshev polynomials. Functions are provided to discretize an interval with Chebyshev polynomial zeros; obtain the Chebyshev polynomial coefficients, powers, and zeros; approximate the least squares estimates on a tensor product Chebyshev polynomial basis; and evaluate the resulting approximating function. MATLAB, Fortran and Windows 32- and 64-bit MEX versions are available. The Fortran and Fortran-based MEX functions are prefixed with an `F`.

4.1 CHEBSPACE

Outputs a discretized interval corresponding to the zeros of the Chebyshev polynomials.

```
Usage:
 z_grid = chebspace(a,b,n)
```

```
Inputs:
 a     : Minimum of interval
 b     : Maximum of interval
 n     : Number of points
Output:
 z_grid: Vector of zeros in [a,b] domain
```

4.2 CHEBPOLY Outputs the coefficients, powers and zeros of the Chebyshev polynomials. Outputs T, P and X are a square matrices of dimension $\max\{\mathtt{M}\}+1$.

```
Usage:
 C = chebpoly(M)
Inputs:
 M    : Function dimensions (vector)
Output:
 C.T  : Chebyshev polynomial coefficients (matrix)
 C.P  : Chebyshev polynomial powers (matrix)
 C.X  : Chebyshev polynomial zeros (matrix)
```

4.3 CHEBWEIGHTSXY Obtains the least squares estimates of the coefficients of a regression of an X-dimensional function against a basis consisting of the tensor product of Chebyshev polynomials. Approximates Y functions simultaneously. The inputs T, P and X are first obtained with chebpoly.m. The inputs f1,...,fY should have the same dimensions.

```
Usage:
 [A1,...,AY] = chebweightsXY(n1,...,nX, ...
                             f1,...,fY, ...
                             T,P,X)
Inputs:
 n[1:X]: Maximum degree of Chebyshev polynomial in each dimension
 f[1:Y]: Function(s) to approximate
 T     : Chebyshev polynomial coefficients
 P     : Chebyshev polynomial powers
 X     : Chebyshev polynomial zeros
Outputs:
 A[1:Y]: Least squares coefficients in approximation
```

4.4 ALLCHEBXYZ_I Evaluates approximating function obtained with least squares regression with a tensor product Chebyshev polynomial basis (X variables, Y functions, Z stochastic components). The suffix I is either T (used in time iteration algorithms) or F (used in fixed point iteration algorithms). The difference is that allchebXYZ_T allows for some dimensions of the function to be evaluated for a single point, which reduces computation costs, whereas allchebXYZ_F evaluates the function(s) at multiple points. The inputs A(1:Y), T and P are obtained with chebweightsXY, and chebpoly.m.

```
Usage:
```

```
 [o1,...,oY] = allchebXYZ_I(z1bnd,...,zXbnd, ...
                            z1i,...,zXi, ...
                            A1,...,AY,T,P)
Inputs:
 z[1:X]bnd : Min and max of each variable (vector)
 z[1:X-Z]i : Point to evaluate for respective variable (scalar)
 z[X-Z+1:X]: Points to evaluate for respective variable (vector)
 A[1:Y]    : Least squares estimates (square matrices)
 T         : Coefficients of Chebyshev polynomials (square matrix)
 P         : Powers of Chebyshev polynomials (square matrix)
Outputs:
 o[1:Y]    : Values of approximating function
```

REFERENCES

RICHTER, A. W., N. A. THROCKMORTON, AND T. B. WALKER (2013): "Accuracy, Speed and Robustness of Policy Function Iteration," Manuscript, Indiana University.