Matlab programs for the Integral Projection Model

Fitting and numerical solution of the integral projection model are easiest in a matrix language such as Matlab. The routines distributed here are extensions of the ones used for the simulation trials and case studies in Easterling (1998) and Easterling, Ellner, and Dixon (2000). They are not completely general, but they include a variety of options for fitting the model that the user can select in arguments to the main routine `lambda`.

Installing the software

The simplest method is to extract all the files into a single directory, such as /integral on a UNIX system or c:\integral in Windows. If you're on a UNIX system, just cd into /integral and start Matlab from there, and Matlab will automatically find all the files that it needs. On a Windows system, use the Matlab path command to make sure that Matlab will have C:\integral in its search path for files.

Data format

The fitting routines require your data to be in a particular format. All the data for one individual at a particular census time goes across a row. A few lines from a typical data file might look as follows:

0.80	0.95	3	0.10	0.15	0.08
0.43	0	0	0.00	0.00	0.00
0.56	0.67	2	0.21	0.09	0.00

The first two columns are the sizes of the individual at two consecutive census times. A zero in the second column represents an individual that died. The third column is the number of offspring each individual produced. The last three columns are the sizes of the offspring. Note that the data file must have the form of a rectangular matrix in order to keep Matlab happy. Columns are therefore added to accommodate the largest number of offspring in the data set, and the lines for individuals with fewer offspring are padded with zeros at the end (for example, if the data are recorded in a spreadsheet, by adding zero to all entries in the data range).

Fitting options

The fitting methods involve least-squares or (logistic) binary regression fitting of 6 functions of an individual's current size, that specify the model:

- Survival probability
- Mean new size (conditional on survival)
- Variance of new size
- Mean number of offspring
- Mean offspring size
- Variance of offspring size

For each of these, the user selects whether the function is fitted by polynomial regression or by a cubic smoothing spline. *NOTE: the spline option currently requires that you have the Matlab Spline Toolbox installed. We plan on removing this requirement in a future release. If you need this, please contact us.* If polynomial regression is selected, then the user also selects the order of the regression (1=linear, 2=quadratic, etc.)

While fitting the integral model, the software also fits a matrix model for comparison, and the user selects the number of discrete size-classes for the matrix model. Size-class boundaries are set at evenly-spaced percentiles of the size distribution in the data set.

The user can set these fitting options in the input to the function `lambda` as described below. Options not specified by the user (other than the name of the data matrix) revert to a default of fitting a 5×5 matrix model, fitting survival probability by a cubic spline, and fitting all other components by linear regression. The option of fitting survival probability by regression is included but **not recommended**, since the regression fits are not constrained to lie between 0 and 1, and the fitting criterion is least-squares.

Running the software.

The sample data files `testdata.txt` and `monkdata.txt` can be used for a trial run of the software. After starting Matlab, the first step is to load the data by typing the command

```
>> load testdata.txt
```

The matrix `testdata` should now be loaded into Matlab's memory. You can check if this has happened using the matlab command

```
>> who
```

and checking that testdata appears in the list of currently defined variables.

To fit the integral projection model and compute the asymptotic growth rate λ using the default values of the fitting options, use the command

```
>> [growrates, kmat, mshpts] = lambda(testdata)
```

(if you happen to forget this while sitting at the keyboard, type

```
>> help lambda
```

and a reminder will appear on the screen). The variable growrates is a 3-entry vector containing

- the asymptotic growth rate from the integral model

- the one-step growth rate computed directly from the data file (i.e. the number of survivors and new offspring in year 2, divided by the total number of individuals in year 1)
- the asymptotic growth rate from the matrix model with user-defined number of classes.

The variable `mshpts` is a vector of unevenly-spaced size values $(x_1, x_2, \ldots x_n)$ spanning the range of the data, and the variable `kmat` is a matrix whose $(i,j)^{th}$ entry is the value of the fitted kernel at (x_i,x_j).

The full call for lambda , in which the user specifies all options, is

```
[grow_rates, kmat, mshpts] =
   lambda(data,numclass,fit_methods,reg_order,makepics,mshsize);
```

The first argument "data" must be given. `lambda(testdata)` uses all the default settings

The second argument "numclass" is the number of classes in the matrix model used for comparison. It has no effect on the integral projection model.

`lambda(testdata,6)`

sets the number of classes for the matrix fit to 6 and uses default values for all the others

`lambda(testdata,6,[1 1 1 2 2 2], [2 2 2 2 2 2], 2, 175)`

a) has 6 matrix classes
b) fits splines (1) for survival, growth, and growth variance and fits regression (2) for the 3 parts of fecundity (mean number of offspring,
c) fits 2nd order (quadratic) regression when regression is used
d) does not print plots
e) uses a mesh of size 175

`lambda(testdata,[],[],[],2,200)`

uses the default values except for not printing plots and 200 for mesh size

Please remember that `kmat` **is not** a matrix projection model corresponding to the integral model; it is a matrix of kernel values for the integral projection model. However, it has the following useful properties:

- the right dominant eigenvector of `kmat` gives the values of the stable size distribution at the sizes $(x_1, x_2, \ldots x_n)$ that were returned in `mshpts`.
- the left dominant eigenvector of `kmat` (which is also the right dominant eigenvector of the transpose of `kmat`) gives the reproductive values of individuals of sizes $(x_1, x_2, \ldots x_n)$.

These eigenvectors can be obtained using the Matlab function `eig`. For example

```
>> [vec, val]=eig(kmat);    % eigenvectors & eigenvalues
>> dvec=vec(:,1);           % extract dominant eigenvector
>> plot(mshpts,dvec/sum(dvec))
```

produces a plot of the stable age distribution (given by `dvec/sum(dvec)`) . And

```
>> [vec, val]=eig(kmat');  % eigenvectors & eigenvalues
>> dvec=vec(:,1);          % extract dominant eigenvector
>> plot(mshpts,dvec/dvec(1))
```

produces a plot of relative reproductive values.

The computed growth rate, stable size distrubution, and reproductive values are numerical approximations whose accuracy increases with the number of mesh points at which the kernel is computed. This is determined by the variable `mshsize` that is set in lambda.m. There is no way to know in advance how many mesh points are needed to get numerically accurate results for a given data set. Users are therefore advised to start with a small value of `mshsize` (50 or so), and increase `mshsize` (75, 100, 150, …) until the computed growth rate no longer changes if more mesh points are added.

Sensitivity and Elasticity calculations

The current release includes a separate function for automatically doing elasticity and sensitivity analysis. The call is

```
[sens_kernel, elas_kernel] = sens(kmat,mshpts)
```

where `kmat,mshpts` are the values returned from `lambda` (which therefore must be called first. Sorry, we don't know who's going to be using this). The returned values are matrices whose i,j^{th} entries are the relative sensitivity and elasticities at (x_i, x_j) where $\{x_i\}$ are the mesh points. A call to `sens` automatically results in these surfaces being plotted, along with the stable size distribution and reproductive value. The plotting code is grouped together at the end of `sens.m` and can be commented out to suppress plotting.

Please do not forget that kmat is not a matrix projection model. The Splus/R code in the neighboring archive produces a (fictitious) matrix projection model as a way of getting numerical solutions to the integral model. This code works differently, and treating `kmat` from `lambda` as if it were a projection matrix will generally give incorrect results.