Software documentation Stephen P. Ellner, Yodit Seifu, and Robert H. Smith. Fitting population dynamic models to time series data by gradient matching.

GRADfuncs.R contains **R** functions for the procedures described in the paper to fit a delay-differential equation population model by gradient matching, plus some utility functions for fitting penalized regression splines. Two standard **R** libraries are required, Mass and quadprog. This document explains how to use the main functions, assuming that the reader already knows how to program in **R**. General introductions to **R** are available from the **R** home page (cran.r-project.org); we recommend the tutorial in the **jumpstart** library. These functions were ported to **R** by SPE and tested in **R** version 1.2.2.

This is not an **R** library, so you need to `source("GRADfuncs.R")` to make the functions available in your current **R** session; this will also load the necessary libraries. **TestGRADfuncs.R** includes some examples of using the functions.

```
bcgrad(tvals, x, hder=NA)
```

Description: Computes a bias-corrected gradient estimate from time series data

Arguments:

`tvals`	observation times (vector)
`x`	values of the time series at the observation times (vector)
`hder`	bandwidth for local polynomial regression used to smooth the time series and obtain the raw gradient estimate (scalar)

Value: returns a list with components

`smooth`	the local polynomial smooth of the time series `x(tvals)`
`raw`	uncorrected estimate of the gradient
unbiased	bias-corrected estimate of the gradient

Details: An intial gradient estimate is obtained by smoothing the time series using local quartic polynomial regression with Gaussian kernel $K(d)=\exp(-d^2/\text{hder}^2)$; if a value of `hder` is not supplied, then `hder=tvals[2]-tvals[1]` is used. The `smooth` and `raw` components of the returned list are the constant and linear coefficients of the local regressions at each observation time. The bias towards 0 in the estimated gradient is then estimated by applying the same procedure to the smoothed values, at times halfway between each successive pair of observation times for the time series. This produces a set of times at which exact and estimated gradients are known. Using these as data, a monotone regression spline for `(true gradient) = f(estimated gradient)` is fitted by GCV (function gcv.rssM) and the spline is then applied to the estimated gradient values at the observation times to obtain an approximately bias-corrected set of gradient estimates.

```
gcv.rss(xvals, yvals, hmin, hmax, nknots=NA, Boundary.knots=NA,
intercept=T, penalty=1, tol=0.01, draw=T)
```

Description: Fits a one-dimensional penalized regression spline $y=f(x)$ to data x=`xvals`, y=`yvals` with smoothing parameter selected by the GCV criterion with adjustable cost per model *df*.

Arguments:

`xvals`	values of the independent variable (scalar)
`yvals`	values of the dependent variable (vector)
`hmin,hmax`	bounds on log10 of the optimal smoothing parameter (scalars)
`nknots`	number of internal knots for the spline (scalar)
`Boundary.knots`	interval-end knots for the spline (vector of length 2)
`intercept`	logical: does model include an intercept at Boundary.knots[1]?

`penalty`	cost per model degree of freedom (trace of the "hat" matrix).
`tol`	tolerance (on log10 scale) for optimizing the smoothing parameter
`draw`	if draw=T, a plot is drawn showing GCV versus smoothing parameter, and the fitted curve.

Value: a list including the following components

`hopt`	log10(optimal smoothing parameter)
`fitted.values`	values of the fitted regression at `xvals`
`gcv`	GCV score of the fitted model
`model`	The fitted model, as returned by the utility function rssfit.

Details: Golden section search is used to find the smoothing parameter (on log10 scale) which optimizes the GCV score for a fitted penalized regression spline. All values of the independent variable must lie between `Boundary.knots[1]` and `Boundary.knots[2];` the `nknots` internal knots for the spline are evenly spaced between these same limits.

Bracketing bounds on the optimal smoothing parameter must be supplied; use `draw=T` to make sure you have a bracket, otherwise the returned fit is meaningless. The cost parameter `penalty` has exactly the same meaning as in `smooth.spline();` the default is `penalty=1` which gives the standard GCV criterion. Setting `penalty=2` is recommended for removing spurious wiggles, with only a minor increase in bias. The returned model is itself a list, which can be used without modification as an argument in `predict.rss.`

```
gcv.rssM(xvals, yvals, hmin, hmax, nknots=NA, Boundary.knots=NA,
intercept=T, penalty=1, ncongrid=NA, tol=0.01, draw=T)
```

Description: Fits a one-dimensional penalized regression spline $y=f(x)$ to data x=`xvals`, y=`yvals` with smoothing parameter selected by the GCV criterion, subject to the constraint that the fitted values at a series of grid points covering the range of the data must be monotonic non-decreasing.

Arguments: same as `gcv.rss` with the addition of

`ncongrid`	number of grid points used in the monotonicity constraint

Value: same as `gcv.rss`.

Details: The constraining grid points are evenly spaced between the `Boundary.knots.` If a value for the number of constraining grid points (`ncongrid`) is not supplied, `ncongrid=50` is used. The constraint of monotonicity at a grid of points is a linear constraint on the model coefficients, and is applied using quadratic programming (`solve.QP` from the `quadprog` library); the GCV score is adjusted to take the active constraints into account (see Wood 1997, cited in the paper).

```
predict.rss(model,newx)
```

Description: generate predicted values from a penalized regression spline model.

Arguments:

`newx`	Values of the independent variable at which predicted values are desired
`model`	Penalized regression spline model as returned by `rssfit,rssfitM, gcv.rss, or gcv.rssM.`

Value: a list with components

`x`	same as `newx`
`y`	the corresponding predicted values

Details: information in `model` is used to compute the appropriate spline basis functions at the values in `newx` (via a call to the utility function `rssbasis`) , and the coefficient matrix of `model` is then used to calculate predicted values at `newx`.

```
gcv.rssadd2(x, y, nknots1=20, nknots2=20, Boundary.knots1=NA,
Boundary.knots2=NA, intercept1=T, intercept2=T, penalty=1,
signs=c(1,1), par=c(0,0))
```

Description: Fits an additive bivariate penalized regression spline $y = f_1(x[,1]) + f_2(x[,2]) + error$ with smoothing parameter selected by the GCV criterion, subject to sign constraints on the values of the ridge functions at the knots.

Arguments: similar to gcv.rss with *i*=1,2 indicating which ridge function.

`x,y`	values of the independent (`x`:matrix) and dependent (`y`:vector) variables
`nknots`	numbers of internal knots for the ridge functions
`Boundary.knots`	interval-end knots for the splines (vectors of length 2)
`intercept`	logical: should the model include an intercept term?
`penalty`	cost per model degree of freedom.
`signs`	Sign contstraint: ridge function *i* is constrained to be nonnegative if `signs[i]`>0, nonpositive if `signs[i]`<0, at each knot value.
`par`	initial guess for log10(optimal smoothing parameter) (vector, length2)

Value: a list including the following components.

`hopt`	log10(optimal smoothing parameters)
`fitted.values`	values of the fitted ridge functions at `x` (matrix)
`gcv`	GCV score of the fitted model
`model`	The fitted model, as returned by the utility function `rssadd2`

Details: If you don't want sign constraints, then use `gam()` in Simon Wood's **mgcv** library. An additive regression spline again has the form of linear regression onto a set of basis functions, and the sign constraints are applied with `solve.QP` as in `gcv.rssM`. Optimization of the GCV score is performed by `optim` with `method="Nelder-Mead";` if in doubt about convergence you can try re-starting from the returned "optimal" smoothing parameters, or multiple fits from several different initial guesses.

```
predict.rssadd2(model,newx)
```

Description: generate predicted values from a additive bivariate penalized regression spline model.

Arguments:

`newx`	Values of the independent variable at which predicted values are desired
`model`	Penalized regression spline model as returned by `gcv.ssadd2` or `rssadd2`

Value: a list with components

`x`	same as `newx`
`y`	the corresponding predicted values of the two ridge functions (matrix)

Details: information in `model` is used to compute the appropriate spline basis functions at the values in `newx` (via a call to the utility function `rssbasis`) , and the coefficient matrix of `model` is then used to calculate predicted values at `newx`. Note that the two ridge function values are returned separately as the two columns of `y`.