Chapter 11
Functional Models and Dynamics

This chapter brings us to the study of continuous time dynamics, where functional data analysis has, perhaps, its greatest utility by providing direct access to relationships between derivatives that could otherwise be studied only indirectly. Although dynamic systems are the subject of a large mathematical literature, they are relatively uncommon in statistics. We have therefore devoted the first section of this chapter to reviewing them and their properties. Then we address how "principal differential analysis (PDA)" can contribute to their study from an empirical perspective.

11.1 Introduction to Dynamics

Functional data offer access to estimated derivatives, which reflect *rates of change*. We have already seen the interpretative advantage of looking at velocity and acceleration in the Berkeley growth data. The field of *dynamics* is the study of systems that are characterized by relationships among derivatives. Newton's Second Law,

$$F = ma$$

which we can rewrite in functional data analysis terms as

$$D^2x(t) = \frac{1}{m}F(t), \tag{11.1}$$

is probably the most famous dynamic model. It is, in fact, a concurrent functional linear model with a constant coefficient function where force F is the functional covariate predicting acceleration D^2x. Dynamic models such as this are developed in the physical sciences from first principles and are often proposed as approximations to data derived from complex systems of all kinds.

J.O. Ramsay et al., *Functional Data Analysis with R and MATLAB*, Use R,
DOI: 10.1007/978-0-387-98185-7_11,

11.1.1 An Example of First-Order Dynamics

Consider a straight-sided bucket of water with a leak at the bottom. Water will leak out from the hole at a rate proportional to the amount of pressure on the bottom of the bucket and the size of the hole (ignoring second-order effects like surface tension and flow turbulence). Since the pressure is proportional to the height $x(t)$ of the water in the bucket at time t, the flow rate $Dx(t)$ can be described as follows:

$$Dx(t) = -\beta x(t). \tag{11.2}$$

The negative sign is introduced here because water flowing *out* of the bucket reduces the height.

Equations with this structure have the solution

$$x(t) = Ce^{-\beta t}.$$

Since $C = x(0)$, it is called the *initial condition or state* of this system. Since in our example $\beta > 0$, the height of the water exhibits exponential decay.

If a hose adds water to the bucket at a rate $g(t)$, Equation (11.2) becomes

$$Dx(t) = -\beta x(t) + \alpha g(t). \tag{11.3}$$

The coefficient α is required to match the units of the two terms. The input function $g(t)$ is called a *forcing function*, changing the unforced behavior of the system.

Of course most buckets change their diameter with height, and there will be additional loss from evaporation, splashing and so forth. Effects such as these would require the coefficients to change with time:

$$Dx(t) = -\beta(t)x(t) + \alpha(t)g(t). \tag{11.4}$$

This is a concurrent functional linear model predicting the instantaneous rate of change $Dx(t)$ on the basis of two covariates: the *state* $x(t)$ and the external input or forcing $g(t)$.

Not all systems can be interpreted or developed so readily. Nonetheless, exploring the relationships between derivatives in a system can provide a useful guide to understanding its behavior. One of the useful aspects of first- and second-order linear dynamics is that explicit solutions can be given for systems like (11.3) that enable us to understand the instantaneous nature of the system's behavior. These are explored further in the next section.

Even fairly simple dynamic systems can produce highly complex behavior. This is one of the reasons they are such powerful mathematical tools. Analyzing such systems is an active area of research in applied mathematics with a large body of literature. However, for linear systems, it is fairly easy to write down explicit solutions.

When a forcing function is included the system, (11.3) leads to

$$x(t) = Ce^{-\beta t} + e^{-\beta t} \int_0^t \alpha e^{\beta \tau} g(\tau) \mathrm{d}\tau.$$

To make this equation more interpretable, let us consider the situation where $g(t) = g$ is constant:

$$x(t) = Ce^{-\beta t} + \frac{\alpha g}{\beta}.$$

The height of water tends to a level $\alpha g/\beta$ that balances inflow of water with outflow leaving the bucket. Moreover, it tends to that level at an exponential rate. As a rule of thumb, the exponential term implies that $x(t)$ will move approximately two thirds of the distance to $\alpha g/\beta$ in $1/\beta$ time units.

11.1.2 Interpreting Second-Order Linear Dynamics

Of course, relationships between x and Dx may not capture all the important information about how a system evolves. *Linear second-order dynamics* are expressed as

$$D^2x(t) = -\beta_0 x(t) - \beta_1 Dx(t) + \alpha g(t). \tag{11.5}$$

A good way to understand (11.5) is to think of a physical system described by Newton's Second Law: Each of the terms represents a different "force" on the system. The first term represents position-dependent forces like a spring for which the "stress" (force) is proportional to the "strain" (deformation). The second term is proportional to the speed at which the system moves, and can be thought of in terms of friction or viscosity, especially when β_1 is positive. As before, $g(t)$ again represents external inputs into a system that modify its behavior, like Newton's Second Law, expression (11.1) above.

Let us first suppose that $\beta_1 = \alpha = 0$ so that

$$D^2x(t) = -\beta_0 x(t).$$

If $\beta_0 \geq 0$, the solutions are of the form

$$x(t) = c_1 \sin(\sqrt{\beta_0}t) + c_2 \cos(\sqrt{\beta_0}t),$$

which are periodic with a period $1/\sqrt{\beta_0}$.

More generally, if $\beta_1 \neq 0$, we examine the *discriminant*

$$d = \beta_1^2/4 - \beta_0.$$

Direct differentiation shows that solutions are given by linear combinations of exponential functions

$$x(t) = c_1 \exp(\gamma_1 t) + c_2 \exp(\gamma_2 t)$$

with

$$\gamma_1 = \frac{-\beta_1}{2} + \sqrt{d}, \quad \gamma_2 = \frac{-\beta_1}{2} - \sqrt{d}.$$

These solutions will decay exponentially if $\gamma_1 < 0$ (since $\gamma_2 <= \gamma_1$). If $d < 0$, γ_1 and γ_2 are complex conjugates, and

$$x(t) = \exp(-\beta_1 t/2)[d_1 \sin(t\sqrt{-d}) + d_2 \cos(t\sqrt{-d})]. \tag{11.6}$$

This yields oscillations that increase or decrease according to the sign of β_1. Moreover, the oscillations have period $2\pi/\sqrt{-d}$.

Using these observations, we can divide the (β_0, β_1) space into regions of different qualitative behavior: oscillatory and nonoscillatory, exponential growth and exponential decay. This division is depicted in Figure 11.1.

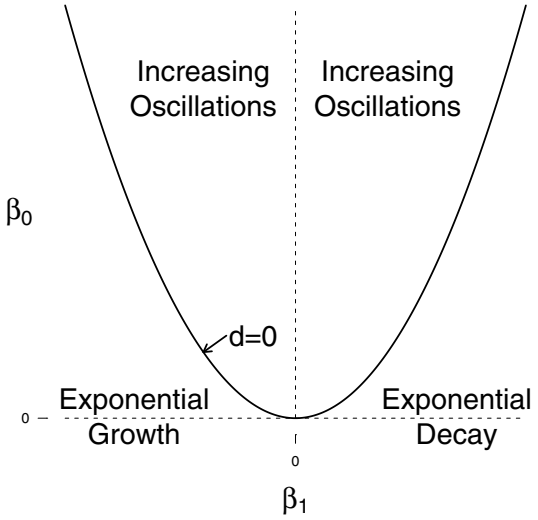

Fig. 11.1 A diagram of the various dynamic regimes for a second-order differential equation for different values of β_0 and β_1.

For many purposes, we may want to generalize expression (11.5) further to consider time-varying coefficients:

$$D^2 x(t) = -\beta_1(t) Dx(t) - \beta_0(t) x(t) \tag{11.7}$$

in terms of the instantaneous values of the functional discriminant

$$d(t) = \beta_1(t)^2/4 - \beta_0(t).$$

These diagnostics should be understood with some caution: If the coefficient functions in (11.7) vary rapidly, their instantaneous changes may not translate into substantive changes in the overall behavior of $x(t)$. For example, if $d(t)$ becomes negative only briefly, there may not be enough time for any meaningful oscillation to occur. Rapidly changing coefficient functions or strong relationships between the coefficient functions of $x(t)$ and its derivatives may provide a good indication that a more complex, nonlinear system could be considered.

We draw from all of this that a relatively simple dynamic equation like (11.5) can define a wide variety of behaviors. We also see the important issue of system *stability*. There are multiple definitions of stability, but in general if the coefficient of velocity, $\beta_1(t)$, is positive, the system will be stable, exhibiting something like exponential decay possibly with a damped oscillation; if $\beta_1(t)$ is negative, the system will exhibit something like exponential growth or a similarly growing oscillation.

11.1.3 Higher-Dimensional Linear Systems

Dynamic models can include more than one state variable. The second-order system discussed in Section 11.1.2 can be cast as a first-order system with a vector state, with components representing the location and velocity. In this context it is less easy to produce analytic expressions with which to analyze the stability properties of a system. However, the rules are not very different. A multidimensional linear system involving a k-dimensional state can be written as

$$D\mathbf{x}(t) = -\mathbf{B}(t)\mathbf{x}(t), \tag{11.8}$$

where $\mathbf{B}(t)$ is now a $k \times k$ matrix. It is clear that for this system, $\mathbf{x}(t) \equiv 0$ results in a stable solution.

We can specialize this system to constant coefficients and add a forcing function to get the following:

$$D\mathbf{x}(t) = -\mathbf{B}\mathbf{x}(t) + \mathbf{u}(t).$$

If $\mathbf{u}$ is constant, this system has a *fixed point* at $\mathbf{x} = -\mathbf{B}^{-1}\mathbf{u}$ at which the solution does not change. We can understand the stability of this solution in terms of the eigenvalues $d_1, \ldots, d_k$ of $-\mathbf{B}$. Letting

$$\xi_j(t) = e^{d_j t}, \ j = 1, \ldots, k,$$

the solution to (11.8) is given in terms of linear combinations of the $\xi_j(t)$. For a general matrix $\mathbf{B}$, some of the eigenvalues may be complex. For real-valued matrices $\mathbf{B}$, any complex eigenvalue will be paired with its complex conjugate. Moreover, the imaginary parts describe the oscillations that we observed for the second-order system, with the period of oscillation being 2π over the (positive) imaginary part of each complex conjugate pair. Moreover, any eigenvalue with a positive real part will explode exponentially; a complex conjugate pair of eigenvalues with a positive real

part will exhibit an exponentially increasing oscillation. A forcing term may shift the behavior but will not change the stability properties unless the forcing term is a function of the state vector in a way that in essence modifies the state transition matrix $\mathbf{B}$.

How are we to deal with higher-order multivariate dynamics? In Section 11.4 we use a model of the form

$$D^2\mathbf{x}(t) = -\mathbf{B}_0(t)\mathbf{x}(t) - \mathbf{B}_1(t)D\mathbf{x}(t) + \mathbf{u}(t).$$

In order to examine the stability of this system, we expand it by creating a new variable $\mathbf{y}(t) = D\mathbf{x}(t)$. This system can be written down as

$$\begin{pmatrix} D\mathbf{y}(t) \\ D\mathbf{x}(t) \end{pmatrix} = \begin{pmatrix} -\mathbf{B}_1(t) & -\mathbf{B}_0(t) \\ \mathbf{I} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{y}(t) \\ \mathbf{x}(t) \end{pmatrix} + \begin{pmatrix} \mathbf{u}(t) \\ 0 \end{pmatrix}.$$

That is, we treat $D\mathbf{x}(t)$ as extra dynamic variables to provide a first-order $2k \times 2k$ system. The same analysis may be made of the eigenvalues of the expanded matrix above. As for one-dimensional systems, we can only interpret the local behavior of a system if the parameters in the system change at a much slower rate than the system itself.

11.2 Principal Differential Analysis for Linear Dynamics

We have seen how linear models describing relationships between derivatives results in a system whose behavior can be qualitatively characterized. We would now like to use this theory to characterize the behavior of a system from which we have data.

How can we fit linear dynamic models to functional data? One approach is to solve a differential equation like (11.3) for some value of the parameters and fit this to observed data by least squares. This procedure is computationally expensive, however, and such models rarely fit observed data well since they do not account for unobserved external influences on a system.

Instead, we use the fact that functional data analysis already gives us derivative information. Given repeated measurements of the same process, we can model

$$Dx_i(t) = -\beta(t)x_i(t) + \alpha(t)u_i(t) + \varepsilon_i(t), \tag{11.9}$$

where the $\varepsilon_i(t)$ are error terms to allow for variation between different curves. This expression represents a functional linear regression and could be fit with `fRegress`.

However, we can view the model in a different light: when $u_i(t) \equiv 0$ functional linear regression estimates $\beta(t)$ to minimize

$$\text{PDASSE}(\beta) = \sum_{i=1}^{N} \int [Dx_i(t) + \beta(t)x_i(t)]^2 \, \mathrm{d}t = \sum_{i=1}^{N} \int \left[L_\beta x_i(t)\right]^2 \mathrm{d}t. \tag{11.10}$$

That is, the model looks for a linear differential operator to represent covariation between x and Dx. This method has been labeled *principal differential analysis* (PDA) because of its similarity to principal components analysis:

- Functional PCA looked for linear operators defined by $\beta(t)$ to explain variation between curves.
- PDA looks for linear operators to explain variation between derivatives but within curves.

Naturally, we can extend the same ideas to multivariate functions and to higher derivatives; these are all accommodated in the `fda` package.

When we also wish to consider inputs into a dynamic system, the PDA objective criterion is the difference between the effective input and the linear differential operator:

$$\text{PDASSE}_u(\beta) = \sum_{i=1}^{N} \int \left[L_\beta x_i(t) - \alpha(t)u(t)\right]^2 \mathrm{d}t. \tag{11.11}$$

Both β and α here are functional objects to be estimated. This creates an *input-output* system which responds to changes in $u(t)$. Our examples below do not use forcing functions, but we provide a description of how to incorporate them into the code.

11.3 Principal Differential Analysis of the Lip Data

We illustrate the use of PDA with data on the movement of lips during speech production. Figure 11.2 presents the position of the lower lip when saying the word "Bob" 20 times. As is clear from the data, there are distinct opening and shutting phases of the mouth surrounding a fairly linear trend that corresponds to the vocalization of the vowel. Muscle tissue behaves in many ways like a spring. This observations suggests that we consider fitting a second-order equation to these data.

The function `pda.fd` is the basic tool for this analysis. In a break from our naming conventions, the equivalent Matlab function is `pdacell`. The arguments to this function are similar to `fRegress`. We need to give it the functional data object to be analyzed along with a list of functional parameter objects containing bases and penalties for the β and α coefficient functions. The following code attempts to derive a second-order homogeneous differential equation like expression (11.7) for `lipfd` obtained from smoothing the lip data with no smoothing in the coefficients $\beta_0(t)$ and $\beta_1(t)$:

```
lipfd    = smooth.basisPar(liptime, lip, 6,
             Lfdobj=int2Lfd(4), lambda=1e-12)$fd
names(lipfd$fdnames) = c("time(seconds)",
          "replications", "mm")
lipbasis = lipfd$basis
bwtlist  = list(fdPar(lipbasis,2,0),
```

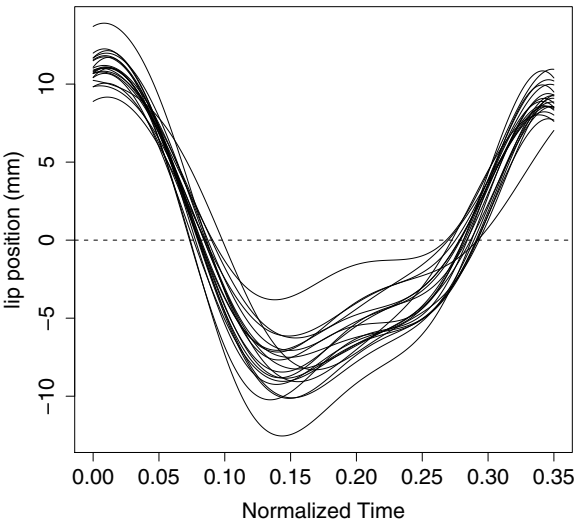

Fig. 11.2 The lip data. These give the position of the lower lip relative to the upper during 20 enunciations of the word "Bob" by the same subject.

```
                    fdPar(lipbasis,2,0))
pdaList  = pda.fd(lipfd,bwtlist)
```

The definition of `pda.fd` provides for arguments `awtlist` and `ufdlist`, whose absence here indicates that the forcing function $\alpha g(t)$ in (11.5) is zero.

We now need to analyze the result. The function

```
plot.pda.fd(pdaList,whichdim=3)
```

will plot the first two panels in Figure 11.3. In higher dimensional systems these coefficient functions can be grouped by dimension, equation, or observation. For the third panel we have plotted the discriminant function:

```
dfd         = (0.25*pdaList$bwtlist[[2]]$fd^2
                 - pdaList$bwtlist[[1]]$fd )
dfd$fdnames= list('time','rep','discriminant')
```

From this we see that there is an initial explosive motion as the lips, previously sealed, are opened. This is followed by a period where the motion of the lips is largely oscillatory with a period of about 30-40 ms. This corresponds approximately to the spring constant of flaccid muscle tissue. During the "o" phase of the word, there is a period of damped behavior when the lips are kept open in order to enunciate the vowel.

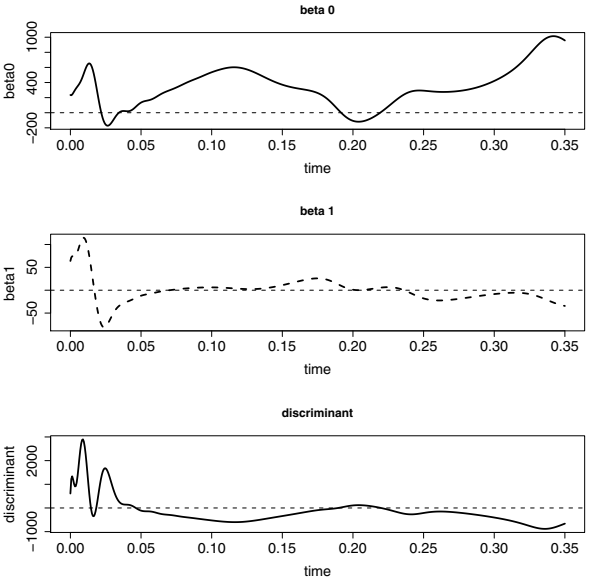

Fig. 11.3 Results of performing principal differential analysis on the lip data. Top two panels represent the estimated $\beta_0(t)$ and $\beta_1(t)$ functional coefficients. The bottom panel shows the discriminant function. This reveals in initial explosive motion as the lips part followed by oscillatory motion, modulated for the production of the "o".

We can also overlay our β coefficients on the bifurcation diagram in Figure 11.1. The following code produces Figure 11.4:

```
pda.overlay(pdaList)
```

This tells much the same story. The initial impulse corresponds to explosive growth, followed by largely stable oscillatory motion.

11.4 PDA of the Handwriting Data

PDA could be effectively carried out using `fRegress`. However `pda.fd` also works for for multivariate functional observations, which cannot be handled by the current version of `fRegress`. Here we examine the handwriting data in Figure 1.8 where PDA provides informative results. As for the lip data, since this is a physical system, we model the second derivative of the data.

For multidimensional systems, a PDA will have three levels of weight functions. Each function is indexed according to the equation in which it appears (i), the variable it multiplies (j), and the derivative of that variable (k):

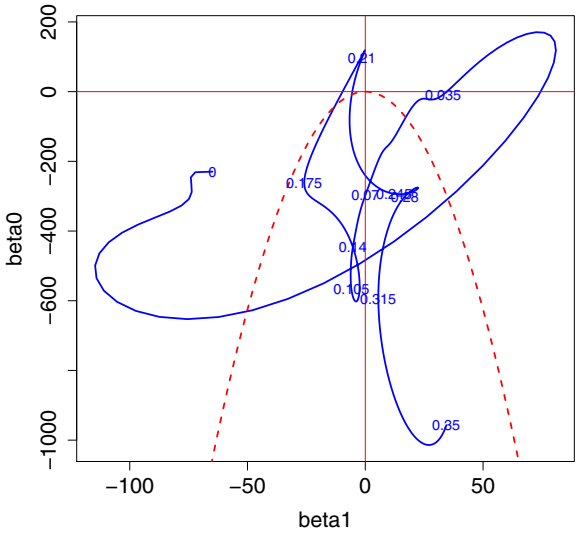

Fig. 11.4 An overlay of the PDA parameters estimated for the lip data on a bifurcation diagram of a second-order linear system. Points are marked as time points from 0 to 0.35 seconds. The initial explosive growth is followed by a period close to undamped oscillations.

$$Lx_{qi}(t) = D^m x_{qi}(t) + \sum_{k=0}^{m-1} \sum_{j=1}^{d} \beta_{ijk}(t) D^k x_{qj}(t). \tag{11.12}$$

the subscript q is used here to represent repeated observations of a multivariate functional process.

In order to account for these levels of functions, for multidimensional systems `pda.fd` uses lists in R and cell arrays in Matlab for both functional data and functional parameter objects. To begin with, we take x to be given by a list of functional data objects, one for each dimension. This allows the various dimensions of x to be defined with respect to different bases. In the handwriting data example we split the `fdafd` object into each of its dimensions:

```
xfdlist = list(fdafd[,1],fdafd[,2])
```

If we want to construct a second-order analysis, we need a three-dimensional array of functional parameter objects. In Matlab, this is simply a three-dimensional cell array, the dimensions being in the order (i, j, k) in (11.12). In R, we achieve the same result by nesting lists within lists: Once we have created the `bwtlist` object, we should have that `bwtlist[[i]][[j]][[k]]` contains the functional parameter object needed to define $\beta_{ijk}(t)$. The following code sets up the analysis of the handwriting data:

```
pdaPar = fdPar(fdabasis,2,1)
pdaParlist = list(pdaPar, pdaPar)
bwtlist = list( list(pdaParlist,pdaParlist),
                list(pdaParlist,pdaParlist) )
pdaList = pda.fd(xfdlist, bwtlist)
```

For higher-dimensional systems, the analysis presented in Figure 11.4 is no longer feasible. Instead, we consider the pointwise eigenvalues of the system that we described in Section 11.1.3. These can be plotted as functional quantities. Nonzero eigenvalues sometimes come in conjugate pairs. Therefore, a plot of the imaginary part of the eigenvalues may be symmetrical. With nonzero imaginary parts, the system oscillates. When the real part of any eigenvalue is positive, the system experiences exponential growth or a growing oscillation. Otherwise it is stable or decaying.

The function `eigen.pda(pdaList)` takes the result of `pda.fd` and produces the stability analysis given in Figure 11.5. As can be see, there is a strong and stable periodic solution in the data, with the real parts of the eigenvalues staying close to zero, indicating that the writing is dominated by ellipsoidal motion.

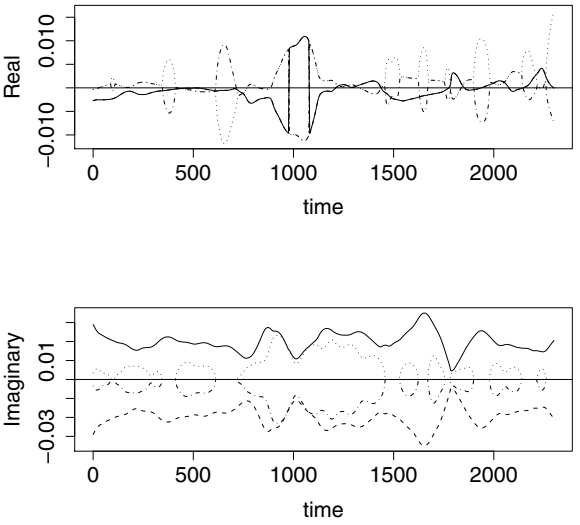

Fig. 11.5 Stability analysis of a principal differential analysis (PDA) of the handwriting data. The nearly constant imaginary eigenvalue indicates a constant cycle modulated locally.

11.5 Registration and PDA

How do we reconcile registration and the analysis of dynamics? These appear to be competing demands. The appearance of features in data such as the pubertal growth spurt makes an *a priori* case for registration: the dynamics of growth are likely to be markedly different during puberty than at other times. We therefore would like to align individuals so that puberty occurs at the same time and the dynamics should therefore be comparable across individuals. However, it is fairly easy to see that registration can have a strong effect on the derivatives of functional data:

$$Dx(h(t)) = D[h](t)D[x](h(t))$$

Here, $D[x](h(t))$ indicates that this is the derivative of x with respect to its argument rather than with respect to t. This will be stronger for second derivatives. We may thus lose the comparison that registration was designed to achieve.

One way around this is to register derivatives individually. That is, we first take derivatives, register each of them individually with the same registration curve, and then conduct a regression analysis. The `register.newfd` function is designed to do this. The following code carries this out for the lip data. We first perform landmark registration, then register each of the first and second derivatives with the resulting registration function. Doing this creates a separate functional data object for each derivative, and these objects no longer represent exact derivatives and anti-derivatives of each other. The function `pda.fd` is designed to analyze the dynamic properties of a single functional data object from which derivatives can be extracted. It will therefore not be usable here. We instead use `fRegress`:

```
lipreglist = landmarkreg(lipfd, as.matrix(lipmarks),
                         lipmeanmarks, WfdPar)
Dlipregfd  = register.newfd(deriv.fd(lipfd,1),
                           lipreglist$warpfd)
D2lipregfd = register.newfd(deriv.fd(lipfd,2),
                           lipreglist$warpfd)
xfdlist    = list(-Dlipregfd,-lipreglist$regfd)
lipregpda  = fRegress( D2lipregfd, xfdlist, bwtlist)
```

The results of this have been plotted with the original PDA results in Figure 11.6: The registered coefficient functions are smoother. The period near the end in which $\beta_0(t)$ is close to zero also suggests a pure frictional force when the mouth is closing.

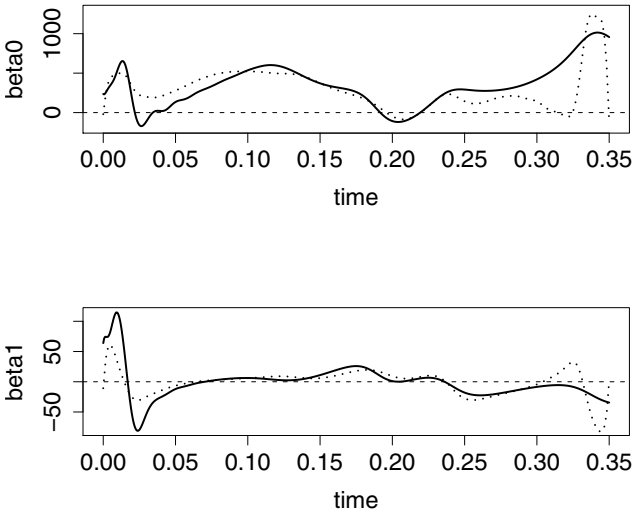

Fig. 11.6 A comparison of PDA results for the unregistered lip data (solid) and the lip data with each derivative first calculated and then registered with the same warping function (dashed).

11.6 Details for `pda.fd`, `eigen.pda`, `pda.overlay` and `register.newfd`

11.6.1 Function `pda.fd`

The `pda.fd` function is like `fRegress` except that since both the response and covariates are given by derivatives of the same function, only one functional data object needs to be specified. It also handles multivariate functional data, but insists that each dimension be given in a separate element of a list. This allows each dimension to be defined with respect to different bases. The standard call is

```
pda.fd(xfdlist, bwtlist,awtlist, ufdlist,nfine)
```

We divide the description of the arguments into two cases:

$x(t)$ is univariate:

`xfdlist` A functional data object defining $x(t)$.

`bwtlist` A list of functional parameter objects, the jth element of each should specify the basis and smoothing penalty for $\beta_j(t)$.

`awtlist` A list of functional parameter objects defining the coefficient functions for the inputs, this should be the same length as `ufdlist` and may be `NULL`.
`ufdlist` A list of functional data objects that act as external influences on the system.

$x(t)$ is multivariate:

`xfdlist` A list of functional data objects, the ith entry defining $x_i(t)$.
`bwtlist` A list of lists of lists of functional parameter objects. `bwtlist[[i]][[j]][[k]]` should define the basis and smoothing penalty for $\beta_{ijk}(t)$.
`awtlist` A list of lists functional parameter objects. `awtlist[[i]][[j]]` represents the coefficient of `ufdlist[[i]][[j]]` in equation i.
`ufdlist` A two-level list of functional data objects, `ufdlist[[i]]` represents the list of input functions that affect equation i.

Both `awtlist` and `ufdlist` default to `NULL`, in which case they are ignored. Individual elements of `bwtlist`, `awtlist` and `ufdlist` can be set to `NULL`, in which case the corresponding coefficient functions are forced to be zero. The `nfine` component gives the number of evaluation points at which to perform a linear regression. It defaults to 501.

The function returns:

`bwtlist` A list array of the same dimensions as the corresponding argument, containing the estimated or fixed weight functions defining the system of linear differential equations.
`resfdlist` A list of length equal to the number of variables or equations. Each member is a functional data object giving the residual functions or forcing functions defined as the left side of the equation (the derivative of order m of a variable) minus the linear fit on the right side. The number of replicates for each residual functional data object is the same as that for the variables.
`awtlist` A list of the same dimensions as the corresponding argument. Each member is an estimated or fixed weighting function for a forcing function.

11.6.2 Function `eigen.pda`

This function calculates the pointwise eigenvalues of the system and produces a plot of the same format as Figure 11.5. If `awtlist` is present, the fixed point of the system is also calculated at each time and plotted. Its arguments are

`pdaList` A list object returned by `pda.fd`.
`plotresult` Should the result be plotted? Default is `TRUE`.
`npts` Number of points to use for plotting.

`...` Other arguments for `plot`.

In addition to producing the plot the function returns a list with elements

`argvals` The evaluation points of the coefficient functions.
`eigvals` The corresponding eigenvalues of the system at each point.
`limvals` The pointwise fixed-point of the system.

11.6.3 Function `pda.overlay`

For a second-order univariate principal differential analysis, this function plots $\beta_0(t)$ against $\beta_1(t)$ and overlays a bifurcation diagram. It requires

`pdaList` A list object returned by `pda.fd`.
`nfine` Number of points to use for plotting.
`ncoarse` Number of points use as time markers along the plot.

11.6.4 Function `register.newfd`

This function will register a given functional data object with a specified warping function. It requires

`yfd` A multivariate functional data object defining the functions to be registered with `Wfd`.
`Wfd` A functional data object defining the registration functions to be used to register `yfd`. This can be the result of either `landmarkreg` or `register.fd`.
`type` Indicates the type of registration function.

- `direct` Assumes `Wfd` is a direct definition of the registration functions. This is produced by `landmarkreg`.
- `monotone` Assumes that `Wfd` defines a monotone functional data objected, up to shifting and scaling to make end points agree. This is produced by `register.fd`.
- `periodic` Does shift registration for periodic functions. This is output from `register.fd` if `periodic=TRUE`.

It outputs a functional data object containing the registered curves.

11.7 Some Things to Try

1. Instead of the time-varying principal differential analysis given for the handwriting data, try a constant-coefficient principal differential analysis, but include a

constant forcing term. Does your interpretation differ markedly? What does the fixed point of the system tell you?

2. Try a PDA of the Chinese handwriting data. Do the dynamics of this system appear to be very different from the cursive script?
3. PDA can be performed on a single time series as well, but we have to borrow strength across times instead of across replicates. One easy way to do this is to insist that all the $\beta_{ijk}(t)$ be constant. Try this with data on the incidence of melanoma over a 30-year period. These data are available in the `melanoma` object.

   a. Smooth the data, choosing the optimal λ by `gcv` and plot both data and the smooth. We observe that there are two distinct dynamics: a linear trend and a cycle with a period of about 10 years.
   b. These observations suggest that

   $$D^4x(t) + \alpha D^2(x) \approx 0.$$

   We would like to get a handle on α. To do this, conduct a PDA using your estimated smooth and representing α as a constant functional data object.
   c. The `pda.fd` function produces an `Lfd` object. Try resmoothing the data using this object to define a penalty. How does the optimal value of λ change? How do the degrees of freedom change?

11.8 More to Read

The study of dynamics has a long history in applied mathematics. Borrelli and Coleman (2004) provide a good introductory overview. While linear differential equations with constant coefficients are relatively easily studied, nonlinear systems are harder to analyze. Generalizing (11.8) a system is described by a vector of states, and its evolution is given in terms of

$$D\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \theta), \tag{11.13}$$

where $\mathbf{f}$ is a vector-valued nonlinear function. Unlike (11.8), however, it is not usually possible to write down solutions to (11.13) analytically. Instead, we must rely on numerical methods to approximate them. Despite these challenges, nonlinear dynamic systems have proved enormously versatile in producing different forms of qualitative behavior that mimic real-world processes, from bursts of neural firing through epidemic processes and chemical reactions. We can examine the behavior of these systems by extending the analysis of the stability of linear systems that we described above and examining how the stability of fixed points and cycles changes with elements of the parameter vector θ. This is a large field and the reader is directed to the relevant literature such as Kuznetsov (2004) to learn more.

Despite the usefulness of such models, there is relatively little literature on assessing their agreement with data or on estimating and performing inference for θ. This is partly due to the numerical difficulties involved in finding solutions to (11.13) and partly due to the idealization involved in assuming that a system evolves deterministically. One way of reducing the numerical challenges in fitting these data is a nonlinear version of PDA; if all the components of $\mathbf{x}$ are measured, we can smooth the data to create $\hat{\mathbf{x}}$ and estimate θ to minimize

$$\text{SISE}(\theta) = \int (D\hat{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \theta))^2 \, \mathrm{d}t.$$

The idea has been rediscovered numerous times (see Bellman and Roth, 1971; Varah, 1982; Pascual and Ellner, 2000; Chen and Wu, 2008). The statistical properties of the resulting estimates have recently been examined (Brunel, 2008). This technique can only be used, however, when there are enough data to smooth each component of $\mathbf{x}$. More recent work has focused on using (11.13) as a smoothing penalty and iteratively refining θ to match the data (Ramsay et al., 2007). The use of functional data analysis in statistical inference for nonlinear systems remains an important research area.