The dynamics of handwriting printed characters

Annotated Analyses in Matlab

Before you begin, don't forget to add the path to the functional data analysis functions. On my system, this is achieved by the command

```
addpath('../fdaM')
```

The Data

The data to be analyzed in these notes are the registered X-, Y-, and Z-coordinates for the handwriting data described in Chapter 7, and in the notes on their analysis in the Web site for that chapter. It is assumed that these analyses have already been completed. These analyses leave us with the registered coordinate values at 470 equally-spaced time points in three-dimensional array `uniregpenpos`. The time values themselves are in vector `unitime`. If you have already done these analyses, and saved the results as .mat file `regprint`, than you can access these variables by this statement:

```
load regprint
```

Removing Linear Trend from the X-Coordinate

We want to develop a second-order linear differential equation for each of the coordinates. However, the X-coordinate differs from the other two in having an overall linear trend, since the hand moves in general from left to right. We first remove this trend, storing it away in matrix `Xcoef` to be re-introduced later after the differential equation has been estimated.

```
unipdapenpos = uniregpenpos;
Xcoef = zeros(20,2);
for i = records
     X = unipdapenpos(:,i,1);
     Xcoef(i,:) = X\[ones(npts,1),unitime];
     unipdapenpos(:,i,1) = X - Xcoef(i,1) - Xcoef(i,2)*unitime;
end
```

Now we make a functional data object out of the de-trended coordinates. We apply a small amount of additional smoothing as well, since we will need to work with third derivatives, and the higher the level of derivative needed, the smoother the curves must be. We penalize the size of the fifth derivative in order to control the curvature of the third.

```
penbasis0 = create_bspline_basis([0,max(unitime)], 47, 8);
unipenfd  = data2fd(unipdapenpos, unitime, penbasis0);
unilambda = 1e-14;
Lfd       = 5;
unipenfd  = smooth_fd(unipenfd, unilambda, Lfd);
```

Finally let's calculate the third derivative values, and their mean and standard deviation.

```
D3       = eval_fd(unitime, unipenfd, 3);
D3mean   = zeros(npts,3);
D3stddev = zeros(npts,3);
for j=1:3
    D3mean(:,j)   = mean(squeeze(D3(:,:,j)),2);
    D3stddev(:,j) = sqrt(var(squeeze(D3(:,:,j))'))';
end
```

The Principal Differential Analysis

Now we are ready to estimate the equation. First we have to set up the arguments required by function pdascalar. This function estimates the forcing function $f(t)$ and the weight functions $\beta_j(t)$ in equation (11.3) separately for each coordinate.

The order of the equation `pdaorder` is three since we want a second order linear equation in velocity. That is, technically we are using derivatives of order zero through three, but we will actually weight the coordinates themselves (derivatives of order zero) by zero. That is, $\beta_0(t) = 0$, but we need to estimate $\beta_1(t)$ and $\beta_2(t)$, and we must do this for each set of 20 coordinate functions. Here we set the order of the equation.

```
pdaorder = 3;
```

The scalar `festimate` has a logical value indicating whether we want to estimate the function function $f(t)$. The vector `westimate` contains logical values indicating which derivative weight functions $\beta_j(t)$ will be estimated and which will be fixed. Only the order zero weight function is fixed.

```
festimate = 1;
westimate = [0; 1; 1];
```

We also need to specify the amount of smoothing, if any, that will be applied to the estimates of these function. In this case we won't be doing any smoothing.

```
flambda = 0;
wlambda = zeros(3,1);
```

The intercept and weight functions require a basis specification since the function will return functional data objects for them. Here we use the B-spline basis of order four, and use 41 basis functions.

```
fbasis = create_bspline_basis([0,max(unitime)], 41);
wbasis = fbasis;
```

Finally, in our setting up of the analysis, we need to supply some default values for the forcing and weight functions. These values won't affect the functions being estimated, but these will supply the zero value for $_0(t)$.

```
ffd0 = fd(zeros(nbasisw,1,3), fbasis);
wfd0 = fd(zeros(nbasisw,4,3), wbasis);
```

Now we carry out the principal differential analysis.

```
[ffd, wfd, resfd] = pdascalar(unipenfd, pdaorder, ...
                              fbasis, flambda, ffd0, festimate, ...
                              wbasis, wlambda, wfd0, westimate);
```

Plotting the Results

Now we want to plot up our estimates, in `ffd` for the forcing functions and in `wfd` for the weight functions. Recall that there are a set of these for each coordinate direction. First, let's look at the forcing functions. This code plots these in three vertically arranged panels.

```
coordlabel = ['X', 'Y', 'Z']
for j=1:3
    subplot(3,1,j)
    plot(ffd); ylabel('meters'); title(coordlabel(j))
    if j==3, xlabel('seconds'); else xlabel(''); end
end
```

Now let's view the weight functions, looking only at those for velocity and acceleration, of course, since the weight on position is 0. This code plots these in a 3 by 2 array of panels.

```
m = 0;
for j=1:3, for i=2:3
        m = m + 1;
        subplot(3,2,m)
        plot(wfd(i,j))
        if i==2
            axis([0,2.34,-400,1200])
            if j==1, title('Velocity'); end
        end
        if i==3
            axis([0,2.34,-20,20])
            if j==1, title('Acceleration'); end
        end
        if j==3, xlabel('seconds'); else xlabel(''); end
        ylabel([coordlabel(j),' Weight'])
```

```
end, end
```

Finally, we can plot the residual functions, and these are returned in `resfd`. For comparison purposes, we will also plot the mean third derivative, which provides a reference function for how small these residuals are.

```
for j=1:3
    subplot(3,1,j)
    if j==1
        title('Residual Functions');
    else
        title('');
    end
    hold on
    plot(resfd(:,j))
    plot(unitime, D3mean(:,j), 'g--')
    xlabel(''), ylabel([coordlabel(j),' m/s^3'])
    axis([0,2.34,-150,150])
    hold off
    if j==3, xlabel('seconds');  end
end
```