Registering the Printing Data

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

These data consist of twenty replications of the printing of the three letters "fda" by a single individual. The position of the tip of the pen has been sampled 200 times per second. The data have already been pre-processed to orient the printed characters properly so that the X-coordinate is from left to right on the page, the Y-coordinate is up and down on the page, and Z-coordinate is vertical motion off the page. Moreover, a simple renormalization of the time scale for each record has also been carried out so that all records take 2.345 seconds, and the original data for each record have been linearly interpolated to set up exactly 470 pen positions. These values are stored as integers in a file called printfda.txt, and each integer gives pen position on the original writing surface in millimeters. The origin of the positions has been fixed by making the average of the positions along each coordinate for each record equal to zero. In the file, the positions for a specific replicate and for a specific coordinate are a single record of 470 values. Each record is preceded by a line indicating the replicate and the coordinate. The basic parameters of the data set are as follows.

```
herz = 200;
nsec =   5;
nrec =  20;
npts = herz*nsec;
```

Here is the code for inputting the data, and converting unit of measurement to meters.

```
fid  = fopen('printfda.dat','rt');
temp = fscanf(fid,'%f');
temp = reshape(temp,[npts+5,3,nrec]);
```

While we're at it, we'll also compute the coordinates of the mean of the pen position over the 20 replications.

```
penpos  = temp(3:(npts+2),:,:)./10000;
penmeanpos = zeros(npts,3);
for j=1:3
    penmeanpos(:,j) = mean(penpos(:,:,j),2);
```

```
end
```

We also need to define an array of coordinates for three fixed reference markers whose position does not change within a record, but which may change from record to record.

```
refpos = temp((npts+3):(npts+5),:,:)./10000;
```

Now we set up the time values.

```
pentime = linspace(0, nsec, npts)';
```

Transforming The Data to a Standard Orientation

The location and orientation of the coordinates in the raw data is relatively arbitrary; it is determined by the position of the camera and the location of the experiment in the room. Consequently, we need to translate and rotate the coordinates so that the handwriting is in the usual orientation that we assume on a page.

We first translate the data so that the origin is equal to the location of the first reference coordinates. Then we rotate the translated coordinates so that the third coordinate of the reference coordinates is zero. This makes the writing surface the X-Y or horizontal plane and the Z or vertical direction the direction in which the pen leaves the paper between strokes.

```
records = 1:nrec;
for i = records
  %  set the origin to the position of ref. IRED 1
  penpos(:,:,i) = penpos(:,:,i) - ones(npts,1)*refpos(1,:,i);
  refpos(:,:,i) = refpos(:,:,i) - ones(3   ,1)*refpos(1,:,i);
  %  compute the rotation taking the ref IRED's to an orientation
  %    = which the Z-coordinate will be 0.
  [u,s,v] = svd(refpos(:,:,i));
  if sum(v(:,3)) < 0, v(:,3) = -v(:,3);  end
  %  rotate the coordinates to this orientation
  refpos(:,:,i) = refpos(:,:,i) * v;
  penpos(:,:,i) = penpos(:,:,i) * v;
end
```

Removing Extraneous Leading and Trailing Parts

The raw data contain substantial pen movement before and after the three letters are formed. We will identify the time points at which the “f” begins and the “a” ends, respectively, by plotting the vertical Z coordinate, and using the `identify()` function to pick out by eye the appropriate points.

In order to see these left and right cut-points clearly, we need to plot the curve on a fairly fine scale, and so we set up two windows, one for the left and one for the right, that

contain the cut points with high probability. If we miss a cut-point for a record, we can always adjust the windows and re-plot.

```
leftind = 050:300;
rigtind = 400:950;
```

Now loop through the records, clicking on the left and right cutpoints, respectively, for each.

```
leftcut = zeros(20,1);
rigtcut = zeros(20,1);

for i = records
    %  plot the Z-coordinate for  this record = the left window
    subplot(2,1,1)
    plot(1:length(leftind), penpos(leftind,3,i), 'o')
    title(['Left Z for  replication',num2str(i)])
    [leftcut(i),y] = ginput(1);
    leftcut(i) = round(leftcut(i)) + leftind(1) - 1;
    %  plot the Z-coordinate for  this record = the right window
    subplot(2,1,2)
    plot(1:length(rigtind), penpos(rigtind,3,i), 'o')
    title(['Right Z for  replication ',num2str(i)])
    [rigtcut(i),y] = ginput(1);
    rigtcut(i) = round(rigtcut(i)) + rigtind(1) - 1;
end

cut = [leftcut,rigtcut];
```

Here we plot the X versus the Y coordinates inside the cutpointsas a way of checking that we have the portion of the script that we want to work with. Use any key-stroke to go from one replication to the next.

```
subplot(1,1,1)
for i = records
    index = leftcut(i):rigtcut(i);
    X = penpos(index,1,i) - mean(penpos(index,1,i));
    Y = penpos(index,2,i) - mean(penpos(index,2,i));
    %  compute the angle between the line joining reference
    %     IRED's 2 and 3 and the horizontal
    theta = atan2(refpos(3,1,i)-refpos(2,1,i), ...
                  refpos(3,2,i)-refpos(2,2,i));
    %  compute the rotation to the horizontal
    S = sin(theta);
    C = cos(theta);
    Tmat = [[C, S];[-S, C]];
    XY = [X,Y];
    XY = XY * Tmat;
    %  plot the X versus the Y coordinates
    plot(XY(:,2), XY(:,1), 'o')
    title(['Replication ',num2str(i)])
    pause;
```

```
end
```

Normalizing the Length of Each Record

We want to simplify our analysis by using a constant number of time points for each curve, even though in actuality the printing times varied across replications. The number 470 of time points is fairly average, and we will use this. We are going to ultimately use curve registration to registered each un-normalized record to a common mean record, but we need first to normalize the data first to get an estimate of this target mean record.

```
npts = 470;
```

Array `unipenpos` will now contain pen positions only within the script window. These pen positions are computed by linear interpolation to be at the npts standard points. Also, the meaning of dimensions is changed as follows:

- dimension 1 now corresponds to sampling points,
- dimension 2 now corresponds to replications,
- dimension 3 now corresponds to coordinates.

Array `unitime` contains the 470 standard time values.

```
unitime   = linspace(0, (npts-1)/200, npts)';
```

Now loop through the records, filling array `unipenpos` after rotating the first two coordinates to standard orientation.

```
unipenpos = zeros(npts,nrec,3);
for i = records
    index = leftcut(i):rigtcut(i);
    ni = length(index);
    pentimei = unitime(npts).*(index-index(1))./ ...
                 (length(index)-1);
    theta = atan2(refpos(3,1,i)-refpos(2,1,i), ...
                  refpos(3,2,i)-refpos(2,2,i));
    S = sin(theta);
    C = cos(theta);
    Tmat = [[C, S];[-S, C]];
    %  set up position functions
    shift  = median(penpos(index,:,i));
    for j = 1:3
        penpos(:,j,i) = penpos(:,j,i) - shift(j);
    end
    penpos(:,[2,1],i) = penpos(:,1:2,i) * Tmat;
    %  interpolate to npts standardized sampling times
    for j = 1:3
        unipenpos(:,i,j) = ...
           interp1(pentimei,penpos(index,j,i),unitime);
```

```
    end
end
```

Compute, too, the mean pen position over these normalized replications.

```
unipenmeanpos = zeros(npts,3);
for j=1:3, unipenmeanpos(:,j) = mean(unipenpos(:,:,j),2);  end
```

This code does a nice plot of each replicate along with the mean curve. Move from replicate to replicate with any key-stroke.

```
xrng = [min(min(unipenpos(:,:,1))), max(max(unipenpos(:,:,1)))];
yrng = [min(min(unipenpos(:,:,2))), max(max(unipenpos(:,:,2)))];
for i = records
    plot(unipenpos(:,i,1),unipenpos(:,i,2), '-', ...
         unipenmeanpos(:,1),unipenmeanpos(:,2), '--')
    xlabel('metres')
    ylabel('metres')
    title(['Record ',num2str(i)])
    axis([xrng,yrng])
    pause;
end
```

Registering Each Record to the Mean Record

Here is the code that does the registration. There are several things to note here. The curves being registered are the un-normalized original curves. The target curve to which they are registered is the mean of the normalized curves. The three coordinates, X-, Y-, and Z are being simultaneously registered; that is, the same time warping function *w(t)* is estimated for all three coordinate functions.

Moreover, it is not the curves themselves that are being registered, but their first derivatives. We use the derivatives for registration because they oscillate more than the curves themselves, and tend to oscillate about zero. After function `registerfd` is called, we use function `regy` to register the curves themselves using the warping function computed from registering their derivatives.

The smoothness of the warping function is also determined by the size of the smoothing parameter `lambda`, set here to 0.01.

The function `registerfd` involves a great deal of looping, and is therefore rather slow. On my 750 mherz Pentium III processor, the following analysis took about 20 minutes.

First we set up a three-dimensional array containing the mean function.

```
fd0mat = zeros(npts,1,3);
for j=1:3
    fd0mat(:,1,j) = mean(unipenpos(:,:,j),2);
end
```

Now we fix the various arguments of function `registerfd` Use the help command to see what these are.

```
Lfd      = 2;
lambda   = 1e-2;
conv     = 1e-3;
periodic = 0;
iterlim  = 10;
dbglev   = 1;
```

Now we loop through the records, registering each in turn.

```
uniregpenpos = unipenpos;
for i = records
    fprintf(['Record ',num2str(i)]);
    index      = leftcut(i):rigtcut(i);
    nptsi      = length(index);
    pentimei   = pentime(index) - pentime(index(1));
    unitimei   = linspace(0,pentimei(nptsi),npts)';
    fdmat      = zeros(nptsi,1,3);
    fdmat(:,1,:) = penpos(index,:,i);
    penbasisi  = create_bspline_basis([0,max(pentimei)], 47, 8);
    penfdi     = data2fd(fdmat,  pentimei, penbasisi);
    penfd0     = data2fd(fd0mat, unitimei, penbasisi);
    Dpenfdi    = deriv(penfdi,1);
    Dpenfd0    = deriv(penfd0,1);
    nbasisw    = 41;
    wbasis     = create_bspline_basis([0,max(pentimei)],nbasisw);
    Wfd0       = fd(zeros(nbasisw,1),wbasis);
    regstr     = registerfd(Dpenfd0, Dpenfdi, Wfd0, Lfd,...
                        lambda, periodic, iterlim, dbglev, conv);
    Wfdi       = regstr.Wfd;
    penregfdi  = regy(penfdi, Wfdi);
    uniregpenpos(:,i,:) = eval_fd(unitimei, penregfdi);
end
```

Some Interesting Plots

We can now compute the mean registered curve.

```
uniregpenmeanpos = zeros(npts,3);
for j=1:3
    uniregpenmeanpos(:,j) = mean(uniregpenpos(:,:,j),2);
end
```

Here we plot the mean curve, and delete portions of the curve that correspond to when the pen is off the paper. A threshold value of 0.5 is used for the Z-coordinate to define these intervals. We also plot dots every 0.5 seconds.

```
X = uniregpenmeanpos(:,1);
Y = uniregpenmeanpos(:,2);
Z = uniregpenmeanpos(:,3);
X(find(Z>0.0005)) = NaN;
Y(find(Z>0.0005)) = NaN;

index = linspace(1,npts,20);
plot(X, Y, '-', X(index), Y(index), 'o')
```

You can, if you wish, repeat the registration process using this new mean curve as the target. You may see some small improvements in the registered curves here are there. Repeating this process more than one is probably not worthwhile.

The magnitude of the acceleration vector, or the tangential acceleration, gives a striking picture of the energy in the handwriting. Here we plot all 20 of these tangential acceleration curves simultaneously. Note that second derivative estimates become very unstable near the boundaries of the records, and so to keep the plot clean, we cut off this unstable part.

```
penbasis = create_bspline_basis([0,max(unitime)],47,8);
uniregfd = data2fd(uniregpenpos(:,:,1:2), unitime, penbasis);
D2matreg = eval_fd(unitime, uniregfd, 2);
D2magreg = sqrt(D2matreg(:,:,1).^2 + D2matreg(:,:,2).^2);

index = find(unitime >= 0.01 & unitime <= 2.33);
plot(unitime(index), D2magreg(index,:))
xlabel('Normalized Time (sec)')
ylabel('Tangential Acceleration (m/sec/sec)')
```

You may want to use this code to plot the unregistered tangential accelerations to see how well the registration has worked.