Registering the Juggling 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

We recorded the three coordinates of the tip of the forefinger ten sequences of ten seconds each. The number of juggling cycles varied from sequence to sequence, ranging between 11 and 13. The data that you have available have already been processed through a number of steps. The original coordinates have been rotated and translated so that the coordinate directions are as described in the Chapter. That is, X is position measured in meters in the frontal plane with positive to the juggler's right. Y is position in the fore-and-aft or sagittal plane, with positive being outwards. Z is vertical. Moreover, each sequence has been already trimmed so that the sequence begins and ends at a fixed identifiable point in the juggling cycle. In other words, there are exactly an integer number of juggling cycles in each sequence, and all ten sequences begin and end at a fixed common stage in the juggling cycle. There are 123 cycles in total.

We need to load the data from text files, and the files that we need are

- `seqn`: a vector containing the 10 numbers of cycles per sequence.
- `ni`: a vector containing the number of coordinate values in each sequence.
- `Xcoord`: a matrix with a column for each sequence, and a row for each coordinate value in the longest of the ten sequences. It contains the X coordinates.
- `Ycoord`: a matrix with a column for each sequence, and a row for each coordinate value in the longest of the ten sequences. It contains the Y coordinates.
- `Zcoord`: a matrix with a column for each sequence, and a row for each coordinate value in the longest of the ten sequences. It contains the Z coordinates.

These commands load the data that we need.

```
load seqn.txt
load ni.txt
load Xcoord.txt
load Ycoord.txt
load Zcoord.txt
```

We now set up the times at which each coordinate is observed for each sequence, also calculate the mean duration for each cycle, which works out to be 0.7129 seconds.

```
timei = (ni-1)/200;
timepercycle = mean(timei./seqn');
```

Here we bundle together the data from the three coordinates into a single three-dimensional array.

```
coord = zeros(size(Xcoord,1),3,10);
coord(:,1,:) = Xcoord;
coord(:,2,:) = Ycoord;
coord(:,3,:) = Zcoord;
```

Setting up a Target Sequence for Each Sequence

Our strategy here is to regard each of the ten sequences an observation, rather than working with individual cycles. We will keep the sequences intact, and only look at the structure of an individual cycle after each sequence has been registered to an image of itself.

We will need to register each of the ten sequences separately, since they vary in length. Our goal is to register each sequence to a target for which the length of each juggling cycle is the same, namely the average of the cycle durations within the sequence.

Consequently, for sequence we estimate an image of the sequence for which the length of each juggle cycle is the same. We do this by expanding each sequence in terms of a Fourier series basis with period equal to this fixed duration. This expansion has enough power to pick up the variation within a typical cycle, but will not reflect any variation from one cycle to another. It is, in effect, an estimate of the average cycle simply replicated as many times as there is cycles.

First, specify the number of terms in the Fourier series, and set up the matrix that will contain these coefficients for each sequence.

```
nbasis0 = 21;
coef0 = zeros([nbasis0,3,10]);
```

Now we can loop through the ten sequences and calculate this Fourier series expansion for each.

```
for i=1:10
    period = timei(i)/seqn(i);  %  average period for this cycle
    rng    = [0,timei(i)];
    basis0 = create_fourier_basis(rng, nbasis0, period);
    time   = linspace(0,timei(i),ni(i));
    curve  = squeeze(coord(1:ni(i),:,i));
    fd0    = data2fd(curve,time,basis0);
```

```
    coef0(:,:,i) = getcoef(fd0);
end
```

The last step in defining the target for the registration phase is to take the average across sequences of these coefficients of the Fourier series expansion.

```
coef0mean = zeros(nbasis0,3);
for j=1:3
    coef0mean(:,j) = mean(squeeze(coef0(:,j,:))')';
end
```

These average coordinates now give us a mean juggling cycle, which we can of course replicate as many times as we wish to match the length of each sequence.

Registering the Sequences to the Target Juggling Cycle

Our first step in registering the sequences is to define some parameters for the function `registerfd`.

```
lambda   = 0;    %  the roughness penalty parameter
Lfd      = 2;    %  the derivative to be penalized
periodic = 0;    %  the data are considered nonperiodic
iterlim  = 10;   %  maximum number of iterations
dbglev   = 1;    %  amount of information displayed by function
conv     = 1e-6; %  convergence criterion
```

The number of coefficients in the Fourier series expansion of sequence *i* and of the registration function $W_i(t)$ will vary. Here we calculate the maximum numbers of coefficients required. Note that, since we will be using a Fourier basis, the number of coefficients will have to be odd. We ask for 11 coefficients per cycle plus 1 to expand the sequence, and 2 coefficients per cycle plut 3 for the registration function.

```
nmax      = max(seqn);
ncoefmax  = 1+11*nmax; % for the sequence
ncoefmaxw = 3+ 2*nmax; % for the registration function
if ncoefmax == round(ncoefmax/2)*2
     ncoefmax  = ncoefmax  + 1;
end
if ncoefmaxw == round(ncoefmaxw/2)*2
     ncoefmaxw = ncoefmaxw + 1;
end
```

Now set up the matrices that will store the coefficients for the two expansions.

```
coefreg = zeros([ncoefmax,3,10]);
coefWfd = zeros(ncoefmaxw,10);
```

A number of calculations are required for each sequence, and we want to discuss each of these in turn. Rather than displaying all these calculations at the same time within a loop over sequences, we will assume that the following code is contained within a for loop

```
for i=1:10
…
end
```

In the following code, then, the variable `i` indicates a specific sequence.

First we repeat the calculation of the number of basis functions needed for the two expansions, but this time just for the *i*th sequence.

```
nbasis  = 1 + 11*seqn(i);
nbasisw = 3 +  2*seqn(i);
if nbasis  == round(nbasis /2)*2, nbasis  = nbasis  + 1;  end
if nbasisw == round(nbasisw/2)*2, nbasisw = nbasisw + 1;  end
```

The duration of the sequence must be calculated as well as the duration of the target sequence. The times of observations are also specified here.

```
period  = timei(i);
period0 = timepercycle*seqn(i);
time    = linspace(0,period, ni(i));
time0   = linspace(0,period0,ni(i));
```

Set up two vectors for the range of the observed and target sequence.

```
rng  = [0,period];
rng0 = [0,period0];
```

Now define three Fourier bases, (1) one for the observed sequence, (2) one for the target sequence, and (3) one for the registration function $W_i(t)$. Note that the period for the observed sequence and for the registration function is the entirely length of the sequence, whereas that for the target function is only for the duration of the standard average cycle.

```
basisi = create_fourier_basis(rng,  nbasis,  period);
basis0 = create_fourier_basis(rng0, nbasis0, timepercycle);
wbasis = create_bspline_basis(rng,  nbasisw, period);
```

Now we're ready to set up the functional data objects that will be input into function `registerfd`. This function `fdi` is the data to be registered. The first command below extracts the data for this sequence, and the second command converts the data into a functional data object.

```
curve = reshape(coord(1:ni(i),:,i),[ni(i),1,3]);
fdi   = data2fd(curve,time,basisi);
```

This function `fd0i` is the target to which the data are registered. The first command extracts the coefficients for this target, and the second sets up the functional data object.

```
coef0i = reshape(coef0(:,:,i),[nbasis0,1,3]);
fd0i   = fd(coef0i,basis0);
```

This function `Wfd0i` is the registration function that defines the warping function.

```
Wfd0 = fd(zeros(nbasisw,1),wbasis);
```

Here we go! Be prepared for this registration to take a few minutes since the sequences are long. The results are returned to a struct object `regstr`.

```
regstr = registerfd(fd0i, fdi, Wfd0, Lfd, lambda, periodic, ...
                    iterlim, dbglev, conv);
```

The last step in the loop is to store the two sets of coefficients that we will need later: (1) those for the registered sequence, and (2) those for the registration function.

```
coefreg(1:nbasis,:,i) = reshape(getcoef(regstr.regfd),nbasis,3);
coefWfd(1:nbasisw,i)  = getcoef(regstr.Wfd);
```

Finally, here is the code again, but for the entire loop.

```
for i = 1:10
    nbasis  = 1 + 11*seqn(i);
    nbasisw = 3 +  2*seqn(i);
    if nbasis  == round(nbasis /2)*2, nbasis  = nbasis  + 1;  end
    if nbasisw == round(nbasisw/2)*2, nbasisw = nbasisw + 1;  end
    period  = timei(i);
    period0 = timepercycle*seqn(i);
    time    = linspace(0,period, ni(i));
    time0   = linspace(0,period0,ni(i));
    rng  = [0,period];
    rng0 = [0,period0];
    basisi = create_fourier_basis(rng,  nbasis,  period);
    basis0 = create_fourier_basis(rng0, nbasis0, timepercycle);
    wbasis = create_bspline_basis(rng,  nbasisw, period);
    curve = reshape(coord(1:ni(i),:,i),[ni(i),1,3]);
    fdi   = data2fd(curve,time,basisi);
    coef0i = reshape(coef0(:,:,i),[nbasis0,1,3]);
    fd0i   = fd(coef0i,basis0);
    Wfd0 = fd(zeros(nbasisw,1),wbasis);
    regstr = registerfd(fd0i, fdi, Wfd0, Lfd, lambda, ...
                        periodic, iterlim, dbglev, conv);
    coefreg(1:nbasis,:,i) = ...
                    reshape(getcoef(regstr.regfd),nbasis,3);
    coefWfd(1:nbasisw,i)  = getcoef(regstr.Wfd);
end
```

Some Interesting Plots

We would like to get an impression of how much the registration process had changed each sequence. We can view this in terms of the departure of the warping function *h(t)* from *t*, or the deformation function *d(t) = h(t) – t*. We can also view this in terms of the curves themselves, plotting the unregistered and registered curves, and possibly the target function as well.

In the following code, let us assume that we are still inside the loop through sequences, and still working with sequence *i*. Moreover, let us assume that all of the quantities that we have already calculated in the registration step are still available. In other words, let's assume that the loop is not yet closed by an `end` statement.

First, let's look at the deformation function. We have to use the coefficients of the Fourier series expansion to compute the registration function *W(t)* from regstr, and then use this to calculate the warping function values.

```
Wfd     = fd(coefWfd(1:nbasisw,i), wbasis);
warpvec = monfn(time, Wfd);
warpvec = warpvec*rng(2)/max(warpvec);
```

Now we plot the deformation function, adding a `pause` statement to allow us to view the plot before moving on to the next plot.

```
subplot(1,1,1)
plot(time, warpvec-time, rng, [0,0], '--')
title(['Deformation for Sequence ',num2str(i)])
pause
```

Now let us compare the unregistered and registered sequences, along with a view of the constant-period target sequence. First we set up the registered function for the sequence.

```
coefi  = reshape(coefreg(1:nbasis,:,i),[nbasis,1,3]);
fdregi = fd(coefi, basisi);
```

Next we evaluate all three functions.

```
fdmat    = squeeze(eval_fd(fdi,    time));
fdregmat = squeeze(eval_fd(fdregi, time));
fd0mat   = squeeze(eval_fd(fd0i,   time0));
```

Of course, we will need a plot for each coordinate, so we set up three panels, and plot the sequences.

```
for j=1:3
```

```
    subplot(3,1,j)
    plot(time0, fdregmat(:,j), '-', ...
         time0, fdmat(:,j),    '--', ...
         time0, fd0mat(:,j),   ':')
    title(['Sequence ',num2str(i),' Coordinate ',num2str(j)])
end
pause
```

It is also interesting to examine tangential velocity and acceleration, which are measures taken across the three coordinates. Here we compute tangential velocity for both the unregistered and registered sequences.

```
Dfdimat   = squeeze(eval_fd(fdi,    time, 1));
Dfdregmat = squeeze(eval_fd(fdregi, time, 1));
TVi    = sqrt(sum((Dfdimat.^2),2));
TVregi = sqrt(sum((Dfdregmat.^2),2));
```

Now we plot the tangential velocities, along with lines demarcating each cycle.

```
subplot(1,1,1)
plot(time0, TVi, '-', time0, TVregi, '--')
axis([0,period0,0,3])
title(['Tangential Velocity for Sequence ',num2str(i)])
hold on
tseq = 0.5:timepercycle:seqn(i)*timepercycle;
for j=1:seqn(i)
    plot([tseq(j),tseq(j)], [0,3], 'r:')
end
hold off
pause
```

Saving the Results

These analyses are required to proceed to the estimation of a differential equation for the data. Therefore, you should now save the results as a .mat file using the command

```
save juggle
```

Plots of the Mean Cycle

Now we want to look more closely at the mean juggling cycle. But first we have to provide an estimate of that mean cycle. We do this by first expanding each sequence in terms of a Fourier basis with a period equal to the length of a standard cycle. This expansion can capture the detail within cycles, but nothing of the variation from cycle to cycle. The following code does this.

```
nbasis0 = 21;
coef0 = zeros(nbasis0,10,3);
for i=1:10
    nbasis  = 1+11*seqn(i);
    if nbasis == round(nbasis/2)*2, nbasis = nbasis + 1;  end
    periodi = timepercycle*seqn(i);
    rngi    = [0,periodi];
    timei   = linspace(0,periodi,ni(i));
    coefi   = reshape(coefreg(1:nbasis,:,i),[nbasis,1,3]);
    basisi  = create_fourier_basis(rngi, nbasis, periodi);
    fdregi  = fd(coefi, basisi);
    fdmati  = eval_fd(fdregi, timei);
    basis0  = create_fourier_basis(rngi, nbasis0, timepercycle);
    fd0i    = data2fd(fdmati, timei, basis0);
    coef0(:,i,:) = squeeze(getcoef(fd0i));
end
```

Now we can compute the mean of these coefficients across the ten sequences, and then make a functional data object using the resulting mean coefficients. This gives an estimate of the average cycle. Note, however, that we use the expansion over two cycles rather than one when we set up the basis because we want some flexibility in terms of where we will begin and end the plot of a typical cycle.

```
coef0mean = zeros(nbasis0,3);
for j=1:3, coefmean(:,j) = mean(coef0(:,:,j),2);  end
basis0   = create_fourier_basis([0,2*timepercycle], nbasis0,
timepercycle);
juggmeanfd = fd(coef0mean, basis0);
```

Now we can evaluate the mean cycle over a time period that is one cycle in length, but begin as 0.505 seconds, when the tangential velocity is at a minimum. We will also evaluate the first two derivatives.

```
timecycle = linspace(0,timepercycle,101) + .505;
D0juggmeanmat = eval_fd(juggmeanfd,timecycle,0);
D1juggmeanmat = eval_fd(juggmeanfd,timecycle,1);
D2juggmeanmat = eval_fd(juggmeanfd,timecycle,2);
```

Here is a three-dimensional plot of the mean cycle.

```
plot3(D0juggmeanmat(:,1),D0juggmeanmat(:,2),D0juggmeanmat(:,3))
axis([-.2,.2,-.2,.2,-.2,.2])
grid on
```

This code produces Figure 12.1 in the chapter.

```
timeseq = 0:0.05:timepercycle;
nseq = length(timeseq)-1;
subplot(1,1,1)
plot(D0juggmeanmat(:,1),D0juggmeanmat(:,3), '-')
```

```
axis([-.15,.2,-.15,.2])
xlabel('\fontsize{12} Meters')
ylabel('\fontsize{12} Meters')
hold on
for i=1:nseq
    sqrdist = (timecycle-timeseq(i)).^2;
    j = find(sqrdist==min(sqrdist));
    plot(D0juggmeanmat(j,1),     D0juggmeanmat(j,3),'.')
    text(D0juggmeanmat(j,1)+.005, ...
         D0juggmeanmat(j,3),      ...
         ['.',num2str(timeseq(i)*100)])
end
sqrdist = (timecycle-.278).^2;
j = find(sqrdist==min(sqrdist));
plot(D0juggmeanmat(j,1),     D0juggmeanmat(j,3),'o')
text(D0juggmeanmat(j,1)+.005,D0juggmeanmat(j,3),'Throw')
sqrdist = (timecycle-.414).^2;
j = find(sqrdist==min(sqrdist));
plot(D0juggmeanmat(j,1),     D0juggmeanmat(j,3),'o')
text(D0juggmeanmat(j,1)-.035,D0juggmeanmat(j,3),'Catch')
sqrdist = (timecycle-.542).^2;
hold off
```

We also want to look at tangential velocity and acceleration, computed with this code.

```
juggmeanTV = sqrt(sum((D1juggmeanmat.^2),2));
juggmeanTA = sqrt(sum((D2juggmeanmat.^2),2));
```

Here we plot the tangential velocity as it appears in the upper left panel of Figure 12.3.

```
subplot(2,2,1)
plot(timecycle, juggmeanTV)
axis([0,timepercycle,0,2.2])
axis('square')
ylabel('\fontsize{12} Tangent. Vel. (m/s)')
hold on
plot(timecycle(40), juggmeanTV(40), 'o')
text(timecycle(40), juggmeanTV(40)-.1, '.28')
plot(timecycle(68), juggmeanTV(68), 'o')
text(timecycle(68), juggmeanTV(68)-.1, '.48')
hold off
```

Here we plot the tangential acceleration as it appears in the upper right panel of Figure 12.3.

```
subplot(2,2,2)
plot(timecycle, juggmeanTA)
axis([0,timepercycle,0,35])
axis('square')
ylabel('\fontsize{12} Tangent. Accel. (m/s^2)')
```

```
hold on
plot(timecycle(59),      juggmeanTA(59), 'o')
text(timecycle(59),      juggmeanTA(59)-2, '.42')
plot(timecycle(68),      juggmeanTA(68), 'o')
text(timecycle(68)+0.005,juggmeanTA(68)+2, '.48')
plot(timecycle(77),      juggmeanTA(77), 'o')
text(timecycle(77),      juggmeanTA(77)-2, '.54')
plot(timecycle(89),      juggmeanTA(89), 'o')
text(timecycle(89),      juggmeanTA(89)+2, '.63')
hold off
```