rows in the **Data Viewer**, starting with `a` and ending with `b`. The formula `a;b` creates a list starting with `a` and incrementing by `b` for each row. For example, replace `t` with `0:6.283` and then replace `V` with `sin(t)`. Plot `V` versus t and you will see a nice sine wave! Pretty lame, huh?

One trick is to read in data from a file and manipulate it with the data browser and use *XPPAUT* as a fancy graphing program.

9.5 Oscillators, phase models, and averaging

One of the main areas of my own research concerns the behavior of coupled nonlinear oscillators. In particular, I have been interested in the behavior of weakly coupled oscillators. Before describing the features of *XPPAUT* that make such analyses easy, I will discuss a bit of the theory. Consider an autonomous differential equation

$$X' = F(X)$$

which admits as a solution an asymptotically stable periodic solution $X_0(t) = X_0(t+P)$, where P is the period. Now suppose that we couple two such oscillators, say, X_1 and X_2:

$$X_1' = F(X_1) + \epsilon G_1(X_2, X_1),$$

$$X_2' = F(X_2) + \epsilon G_2(X_1, X_2),$$

where G_1, G_2 are possibly different coupling functions and ϵ is a small positive number. One can apply successive changes of variables and use the method of averaging to show that, for ϵ sufficiently small,

$$X_j(t) = X_0(\theta_j) + O(\epsilon),$$

with

$$\theta_1' = 1 + \epsilon H_1(\theta_2 - \theta_1), \qquad \theta_2' = 1 + \epsilon H_2(\theta_1 - \theta_2),$$

and H_j are P-periodic functions of their arguments. This type of model is called a **phase model** and has been the subject of a great deal of research. One of the key questions is, What is the form of the functions H_j and how can one compute them? This is where *XPPAUT* can be used. The formula for H_j is

$$H_j(\phi) \equiv \frac{1}{P}\int_0^P X^*(t)\cdot G_j[X_0(t+\phi), X_0(t)]\,dt \tag{9.1}$$

which is the average of the coupling with a certain P-periodic function X^* called the **adjoint** solution. This equation satisfies the linear differential equation and normalization:

$$\frac{dX^*(t)}{dt} = -[D_X F(X_0(t))]^T X^*(t), \quad X^*(t)\cdot X_0'(t) = 1.$$

Here, $D_X F$ is the derivative matrix of F with respect to X and A^T is the transpose of the matrix A.

Thus, to compute the functions H_j, the adjoint must be computed along with the integral. *XPPAUT* implements a numerical method invented by Graham Bowtell for computing the adjoint $X^*(t)$. The method only works for oscillations which are asymptotically stable. Here is the idea. Let $B(t) = (D_X F(X_0(t)))$. Since the limit cycle is asymptotically stable, if we integrate the equation

$$Y' = B(t)Y$$

forward in time, it will converge to a periodic orbit proportional to $X_0'(t)$ as a consequence of the stability of the limit cycle and the translation invariance. Similarly, the solution to

$$Z' = -B(t)^T Z$$

will converge to a periodic orbit if we integrate it *backward* in time. As you can see, this is the desired adjoint solution and is how *XPPAUT* computes it. Once $X^*(t)$ is computed, it is trivial to compute the integral and thus compute the interaction function.

9.5.1 Computing a limit cycle and the adjoint

To use *XPPAUT* to compute the adjoint, as a necessary first step to computing the interaction function H we have to compute exactly one full cycle of the oscillation. Let's use as our example the Morris–Lecar equation:

$$v' = I + g_l(e_l - v) + g_k w(e_k - w) + g_{ca} m_\infty(v)(e_{ca} - v),$$

$$w' = (w_\infty(v) - w)\lambda_w(v).$$

We will look at

$$v_1' = f(v_1, w_1) + \epsilon(v_2 - v_1),$$

$$w_1' = g(v_1, w_1),$$

$$v_2' = f(v_2, w_2) + \epsilon(v_1 - v_2),$$

$$w_2' = g(v_2, w_2).$$

Since the method of averaging depends on the two oscillators being identical, except possibly for the coupling, all you need to do is integrate the isolated oscillator. Use the file `ml.ode`. Change the parameters `I=.09, phi=0.5`. Integrate the equations and click on `Initialconds Last` a few times to make sure you have gotten rid of all transients. Now you are pretty much on the limit cycle. To compute the adjoint, you need one full period. In many neural systems, coupling between oscillators occurs only through the voltage and thus the integral (9.1) involves only one component of the adjoint, the voltage component. For whatever reason, the numerical algorithm for computing the adjoint for a given component converges best if you start the oscillation at the *peak* of that component. Thus, since we will only couple through the potential in this example, we should start the oscillation at the peak of the voltage. Here is a good **trick** for finding that maximum. In the **Data Viewer** click on `Home` which makes sure that the first entry of the data is at the

top of the **Data Viewer**. Click on `Find` and, in the dialog box, choose the variable `v` and, for a value, choose `1000` and then click `Ok`. *XPPAUT* will find the closest value of `v` to `1000` which is obviously the maximum value of `v`. Now click on `Get` in the **Data Viewer** which grabs this as initial conditions. In the **Main Window**, click on `Initialconds Go` to get a new solution. Plot the voltage versus time. Use the mouse to find the time of the next peak. (With the mouse in the graphics window, hold down the left button and move the mouse around. At the bottom of the windows, read off the values.) The next peak is at about 22.2. In the **Datad Viewer** scroll down to this time and find exactly where `v` reaches its next maximum. Note that, as we suspected, it is at $t = 22.2$. In the **Main Window**, click on the `nUmerics` menu and then `Total` to set the total integration time. Choose `22.21` (it is always best to go a little bit over—but just a little). Escape to the main menu and click on `Initialconds Go`. Now we have one full cycle of the oscillation! The rest is easy.

Computing the adjoint

Click on `nUmerics Averaging New adjoint` and after a brief moment, *XPPAUT* will beep. (Sometimes, when computing the adjoint, you will encounter the `Out of Bounds` message. In this case, just increase the bounds and it recomputes the adjoint.) Click on `Escape` and plot v versus time. This time, the adjoint of the voltage is in the **Data Viewer** under the voltage component. (Note that the adjoint is almost strictly positive and looks nearly like $1 + \cos t$. This is no accident and has been explained theoretically.)

9.5.2 Averaging

Now we can compute the average. Recall that the integral depends on the adjoint, the original limit cycles, and a phase-shifted version of the limit cycle:

$$X^*(t) \cdot G(X_0(t+\phi), X_0(t)).$$

In *XPPAUT*, you will be asked for each component of the function G. For unshifted parts, use the original variable names, e.g., `x, y, z` and, for the shifted parts, use primed versions, `x', y', z'`. The coupling vector in our example is

$$(V(t+\phi) - V(t), 0).$$

Thus, for our model, the two components for the coupling are `(v'-v, 0)`. This says that we take the phase-shifted version of the variable `v`, called `v'` by *XPPAUT*, and subtract from it the unshifted version of `v`. With these preliminaries, it is a snap to compute the average. Click on `nUmerics Averaging Make H`. Then type in the first component of the coupling, `v'-v` and the second `0`. Then let it rip. In a few moments, the calculation will be done. If your system has more than two columns in addition to time `t` (as this example does with - `v, w, ica, ik`), then the first column contains the average function $H(\phi)$, the second column contains the odd part of the interaction function, and the third column contains the even part. Plot the function H by escaping back to the main menu and plotting `v` versus time—remember `v` is the first column in the browser after the `t` column. This is a periodic function. For later purposes, we want to approximate this periodic function. Click

on nUmerics stocHastic Fourier and choose v as the column to transform. Look at the data browser and observe the first three cosine terms (column 1 after the t column) and the first three sine terms (column 2). They are (3.34, −3.05, −.29) and (0,3.61,−0.33), respectively. Thus to this approximation,

$$H(\phi) = 3.34 - 3.05\cos\phi - 0.29\cos 2\phi + 3.61\sin\phi - 0.33\sin 2\phi. \qquad (9.2)$$

To get the interaction function, adjoint, or original orbit back into the data browser, click on Averaging Hfun, etc. from the nUmerics menu.

Summary

Here is how to average weakly coupled oscillators:

1. Compute exactly one period of the oscillation—you may want to phase-shift it to the peak of the dominant coupling component.
2. Compute the adjoint from the nUmeric Averaging menu.
3. Compute the interaction function using the original variable names for the unshifted terms and primed names for the shifted terms.

9.5.3 Phase response curves

One of the most useful techniques available for the study of nonlinear oscillators is the **phase response curve** (PRC). The PRC is readily computed experimentally in real biological and physical systems and is defined as follows. Suppose that there is a stable oscillation with period T. Suppose that at $t = 0$ one of the variables reaches its peak. (This can always be done by translating time.) At a time τ after the peak, one of the state variables is given a brief perturbation that takes it off the limit cycle. This will generally cause the next peak to occur at a time $T'(\tau)$ that is different from the time it would have peaked in absence of the perturbation. The PRC $\Delta(\tau)$ is defined as

$$\Delta(\tau) \equiv 1 - T'(\tau)/T.$$

If $\Delta(\tau) > 0$ for some τ, we say that the perturbation advances the phase since the time of the next maximum is shortened by the stimulus. Delaying the phase occurs when $\Delta(\tau) < 0$. Experimental biologists have computed PRCs for a variety of systems such as cardiac cells, neurons, and even the flashing of fireflies. We will use *XPPAUT* to compute the PRC for the van der Pol oscillator

$$\ddot{x} = -x + \dot{x}(1 - x^2)$$

subjected to a square-wave pulse of width σ and amplitude a.

Here is how we will set up the problem. We will let τ be a variable rather than a parameter so that we can range through it and keep a record—it is like a bifurcation parameter. We will define a parameter T_0 which is the unperturbed period. We will use the ability of *XPPAUT* to find the maxima of solutions to ODEs and have the computation stop when the maximum of x is reached. The time at which this occurs is the time $T'(\tau)$ and so we will also track an auxiliary variable $1 - t/T_0$ which is the PRC. Here is the ODE file:

```
# vdpprc.ode
# PRC of the van der Pol oscillator
init x=2
x'=y
y'=-x+y*(1-x^2)+a*pulse(t-tau)
tau'=0
pulse(t)=heav(t)*heav(sigma-t)
par sigma=.2,a=0
par t0=6.65
aux prc=1-t/t0
@ dt=.01
done
```

Note that the function `pulse(t)` produces a small square-wave pulse. To compute the PRC, you want to integrate the equations for a bit to get a good limit cycle with no perturbation ($a = 0$). Then start at the maximum value of x and integrate until the next maximum. This will tell you the base period. Then you want to apply the perturbation at different times and compute the time of the next maximum and then from this get the PRC. Fire up *XPPAUT* with this file. Follow these simple steps:

1. Integrate it and then integrate again using the `Initialconds Last` (**I L**) command to be sure you have integrated out transients.

2. Now find the variable of interest, in this case, x. To do this, in the **Data Viewer** click on `Home` to go to the top of the data window. Click on `Find` and type in `x` for the variable and `100` for the value. *XPPAUT* will try to find the value of `x` closest to 100. This will be the maximum. Click on `Get` to load this as an initial condition.

3. Figure out the unperturbed period. One way is to integrate the equations and estimate the period. A better way is to let *XPPAUT* do it for you. In the **Main Window**, click on `nUmerics Poincare map Max/Min` and fill in the dialog box as follows:

Variable: x
Section: 0
Direction: 1
Stop on sect: Y

and then click on `Ok`. You have told *XPPAUT* to integrate, plotting out only the maxima (Direction=1) of x. The `Stop on section` ends the calculation when the section is crossed. Click on `Transient` and choose 4 for the value. This is so we don't stop at the initial maximum. `Transient` allows the integrator to proceed for a while before looking at and storing values. Now exit the numerics menu by clicking `Esc`. Click on `Initialconds Go` and the program will integrate until x reaches a maximum. In the **Data Viewer**, click on `Home` and you should see that `Time` has a value of around 6.6647. This is the unperturbed period. Set the parameter `t0` to the value in the **Data Viewer**.

4. Now we are all set to compute the PRC. Change the perturbation amplitude from 0 to 3. Click on `Initialconds Range` and fill in the dialog box as follows:

`Range over: tau`
`Steps: 100`
`Start: 0`
`End: 6.6647`
`Reset storage: N`

and click `Ok`. It should take a second or two.

5. Plot the auxiliary quantity `PRC` against the variable `tau`. (Click on `Viewaxes 2D` and put `tau` on the `X-axis` and `PRC` on the `Y-axis`. Click `OK` and then `Window Fit` to let *XPPAUT* figure out the window.) You should see something like the top curve in Figure 9.2.

Freeze this curve and try using a different value of the amplitude `a`.

Exercise. Compute the PRC for the Morris–Lecar oscillator by adding the required parts to the file `ml.ode`. Define a square-wave pulse just like above with a width of 0.25 and a magnitude of 0.1. If you are stuck, download the file `mlprc.ode` which sets up the ODE for you. You should get something like the bottom plot of Figure 9.2.

9.5.4 Phase models

In the previous section we saw how to reduce a pair of coupled oscillators to a pair of scalar models in which each variable lies on a circle. *XPPAUT* has a way of dealing with flows on systems of scalar variables each of which lie on a circle. The phase-space of a system of, say, two such variables, is the two-torus. Let's start with the simplest phase model:

$$\theta_1' = \omega_1 + a\sin(\theta_2 - \theta_1),$$

$$\theta_2' = \omega_2 + a\sin(\theta_1 - \theta_2)$$

representing a pair of sinusoidally coupled oscillators with different uncoupled frequencies, ω_1, ω_2 and coupling strength, a. The ODE file for this is straightforward:

```
# phase2.ode
# phase model for two coupled oscillators
th1'=w1+a*sin(th2-th1)
th2'=w2+a*sin(th1-th2)
par w1=1,w2=1.2,a=.15
@ total=100
done
```

Fire this up and integrate the equations. You will get the `Out of bounds` message at $t = 90$ or so. That is because the variables θ_j are actually defined only modulo 2π and

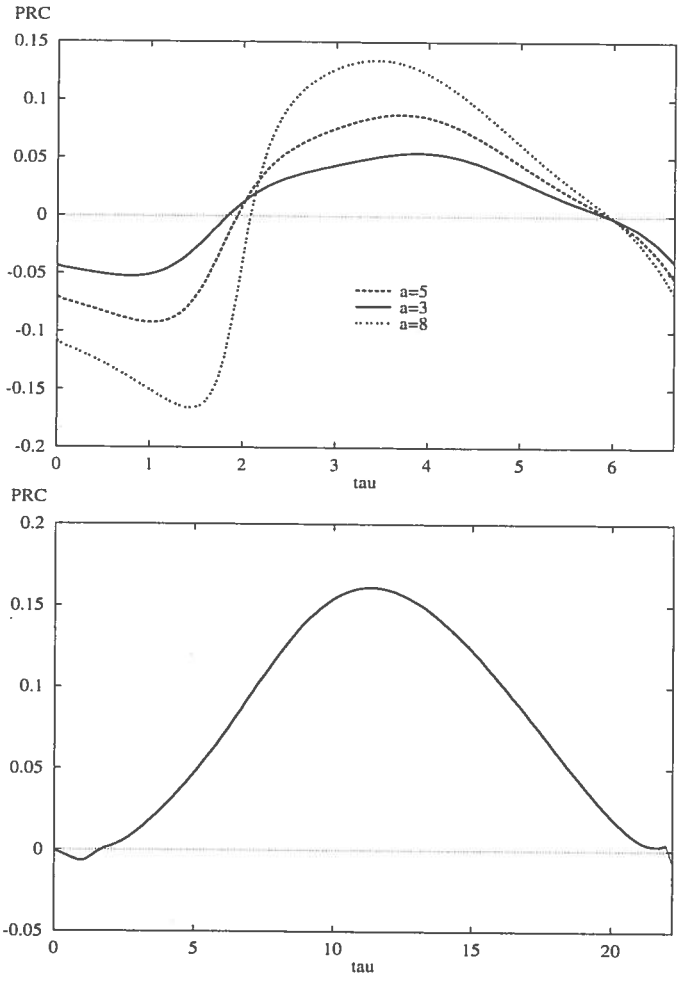

Figure 9.2. *Top: The PRC for the van der Pol oscillator with different amplitudes. Bottom: The PRC for the Morris–Lecar model.*

are not really going out of bounds but instead just wrapping around the circle. Thus, it is more proper to look at θ_j modulo 2π. If you define an auxiliary variable which is $\theta \bmod 2\pi$ and then plot it, you will get a series of ugly lines that cross from one part of the screen to the other. This is because *XPPAUT* does not know that this particular variable is defined on the circle. There is a simple way to tell *XPPAUT* which of the state variables lie on the circle. Click on `phAsespace Choose` (**A C**) and when prompted for the period, choose the default which is approximately 2π. A little window will appear with all your variables included. Move the cursor to the left of each of them and click the mouse to see a little X next to the variable. Put this mark next to both variables and click on `done`. This tells *XPPAUT* that these are both considered "folded" variables and they will be folded mod 2π. Now reintegrate the equations. This time you get no such out of bounds message—instead

you will see that the plots are all modulo 2π. Switch the view to the phase-plane for the two variables, θ_1, θ_2. (In the **Initial Data Window** click on the box to the left of each variable and then on the `xvsy` button.) You will see that the trajectories seem to converge on a diagonal line which is shifted slightly upward. Try integrating using the `Initialconds mIce` (**I I**) command to choose a variety of initial conditions. They should converge to the same line. This is an example of a phase-locked solution—an invariant circle on the torus. Change a from 0.15 to 0.08 and integrate the equations again. Notice how the trajectories do not converge to an attractor. Instead, the whole torus will gradually fill up. Phase-locking no longer occurs.

A derived example

Now let's turn to the example that we derived in the previous section and see if a pair of oscillators will phase-lock when we use the interaction function defined in (9.2), `phase_app.ode`:

```
# phase_app.ode
# phase model for two coupled oscillators
# using numerically computed H
h(x)=3.34-3.05*cos(x)-.29*cos(2*x)+3.61*sin(x)-.33*sin(2*x)
th1'=w1+a*h(th2-th1)
th2'=w2+a*h(th1-th2)
par w1=1,w2=1,a=.1
@ total=100
@ fold=th1,fold=th2
@ xlo=0,ylo=0,xhi=6.3,yhi=6.3,xp=th1,yp=th2
done
```

I have added a few new @ commands. The `fold=th1` directive tells *XPPAUT* to make a circle out of the variable `th1`, that is, to mod it out. All variables that you want to mod out can be set by the `fold=name` directive. This automatically tells *XPPAUT* to look for modded variables. The default period is 2π so we don't have to change it. If you want to change the period to, say, 3, set it with the command `@ tor_period=3`. Run *XPPAUT* on this, using the mouse to set some initial conditions. Note how all initial data synchronize along the diagonal, implying $\theta_1 = \theta_2$. Change the intrinsic frequency `w2` and see how big you can make it before phase-locking is lost.

Pulsatile coupling

Winfree [42] introduced a version for coupling oscillators using the PRC of the oscillator, assuming that the interaction between oscillators occurred only through the phase and took the form of a product. Ermentrout and Kopell [11] proved that this was a reasonable model when certain assumptions were made concerning the attractivity of the limit cycle. Here is a pair of pulse-coupled phase models:

$$\frac{d\theta_1}{dt} = \omega_1 + P(\theta_2)R(\theta_1),$$