```
$ mc 1000000000 1 12345
```

and confirm you get exactly the same answer. Your program should print the approximate value of the integral and the error compared to the exact solution. Your program should also print the time required to complete the calculation.

Determine the number of samples required to achieve an error tolerance of 10^{-4} with 99% confidence. Verify your approximation is within tolerance by comparing to the exact solution

$$\int_0^\infty e^{-\lambda x}\cos(x)\,dx = \frac{\lambda}{1+\lambda^2}.$$

Calculate the time required to compute the solution for $N = 10^p$ for $p = 3, 4, \ldots, 10$ and plot the time versus N. Use the plot to estimate the time required to meet the error tolerance of 10^{-5} with 99.5% confidence.

34.3.2 ▪ Duffing–Van der Pol Oscillator

The Duffing–Van der Pol oscillator describes the motion of a spring-mass system with a nonlinear spring coefficient. Adding a multiplicative noise forcing term, the equation becomes

$$x'' + x' - (\alpha^2 - x^2)x = \sigma x \xi$$

where ξ is white noise. Converting this to a first-order system of stochastic differential equations gives

$$dX_t = Y_t\,dt, \tag{34.18}$$
$$dY_t = ((\alpha^2 - X_t^2)X_t - Y_t)\,dt + \sigma X_t\,dW_t. \tag{34.19}$$

Figure 34.2 illustrates two sample paths, one for which $\sigma = 0$, i.e., with no noise, and one for $\sigma = 0.5$. Note that without noise, the path will converge asymptotically to $(\pm\alpha, 0)$ depending on the initial condition, but with noise, the solution may bounce back and forth between the two equilibria.

When solving such an equation, it is not done just once but done many, many times in order to get statistics about, for example, the value of X_T at some terminal time T. Figure 34.3 illustrates the histogram for the (X, Y) phase plane for the terminal time $T = 10$, where $(X_0, Y_0) = (0.1\,N(0;1), 0)$ and with $\alpha = 1$, $\sigma = 0.5$. The value $N(0;1)$ is a normally distributed random value with mean zero and variance one, i.e., a standard Gaussian distribution. The two stable equlibria at $(\pm 1, 0)$ are clearly evident as expected. Simulating any single evolution may be cheap, but repeating the calculation to the point where the distribution of X_T is resolved may require a large amount of sampling.

This is a great example of an embarrassingly parallelizable algorithm. Suppose this simulation must be run N times, where N is large. The simplest strategy in this instance is to ask each of P cores to do N/P simulations and to keep their own statistics.

Assignment. Use the Euler–Maruyama method to solve the Duffing–Van der Pol oscillator equation.

Your program should take four input arguments plus an optional argument for the seed so that the program named **vdp** can be run with the command

```
$ vdp 1 0.5 1000 100000
```

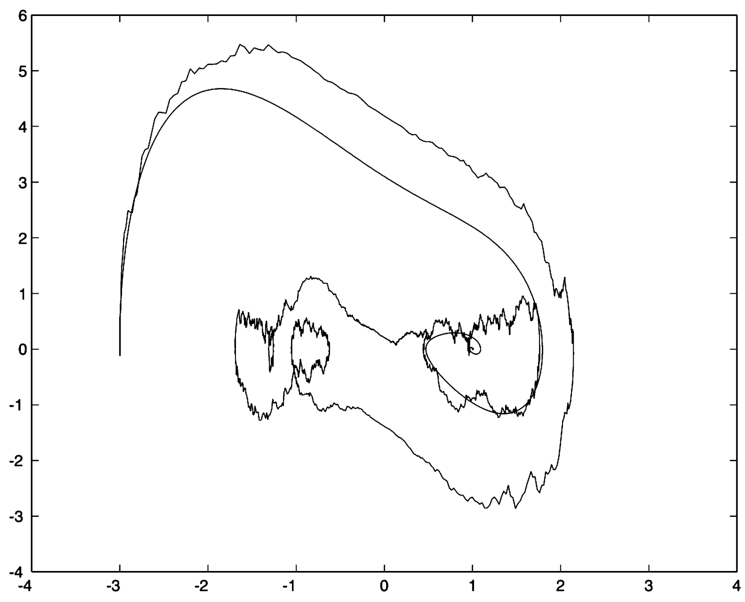

Figure 34.2. *Sample phase plane plots for the cases of* $\sigma = 0$ *and* $\sigma = 0.5$ *for the Duffing–Van der Pol oscillator with multiplicative noise forcing, Equations (34.18), (34.19), and using* $\alpha = 1$.

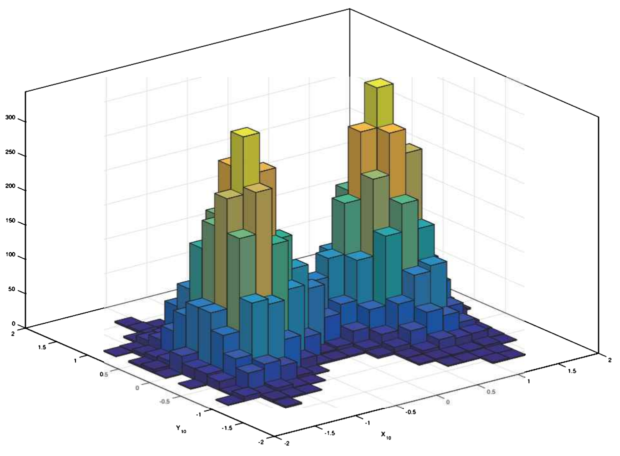

Figure 34.3. *Example of a histogram for the Duffing–Van der Pol oscillator at terminal time* $T = 10$ *with initial condition* $(X_0, Y_0) = (0.1\,N(0;1), 0)$, *and with parameter values* $\alpha = 1$, $\sigma = 0.5$. *The histogram is generated by 10,000 sample runs solving Equations (34.18) and (34.19).*

where $\alpha = 1$ is the half distance between the two stable attractors, $\sigma = 0.5$ is the strength of the noise, $M = 1000$ is the number of time steps to use, and $N = 100{,}000$ is the number of trials. The terminal time for each trial will be $T = 10$ so that the time step size will be $\Delta t = T/M$. Use initial condition $(X_0, Y_0) = (0.1\,N(0;1), 0)$ where $N(0;1)$ is a normally distributed variable with mean zero and variance one. Your program should print the value of the arguments given including the initial seed used so that running the program a second time with the optional seed will give the same result. For example, if the first seed printed is 12345, then running the program a second time using

```
$ vdp 1 0.5 1000 100000 12345
```

will give the exact same results.

Define the function $p(t)$ to be the probability of being close to an equilibrium point by

$$p(t) = P\left\{\|(X_t, Y_t) - (-\alpha, 0)\| \leq \frac{\alpha}{2}\right\} + P\left\{\|(X_t, Y_t) - (\alpha, 0)\| \leq \frac{\alpha}{2}\right\}.$$

To do this, for each time value $t = k/10$, count how many trials resulted in the point (X_t, Y_t) satisfying one of the two above inequalities. Create a double precision array `P[101]` where `P[k]`$\approx p(k/10)$ for $0 \leq k \leq 100$. Your program should save the `P` array to a file called "`Prob.out`" in binary format using the `fwrite` function as described in Section 4.2.

Your program must also print to the screen the total time required to complete the calculation. Use varying values of M and N to generate a plot that illustrates how the time to compute the solutions varies with these values.