Chapter 7

Post Processing, Visualization, and Sample Output

Upon successful execution of an ADCIRC run, a number of output files will be created. Specifically which files are created depends upon how the fort.15 file was structured, i.e. what type of output was requested.

ADCIRC output can be written either in ASCII (simple text) or binary format. The former has the advantage of being easily read in by a wide variety of applications (text editors, spreadsheets, etc.) while the latter has the advantage of being far more compact. As ADCIRC output is very dense, both spatially (large finite-element grids) and temporally (output is available at as many time steps as the user desires), output data files can quickly become extremely large.

Finding efficient and attractive ways to visualize this output is crucial to the end user. The intent of the current chapter is to outline several Matlab scripts written by the author with the goal of visualizing and / or animating the model output. Sample results will be provided to help illustrate the capability of these scripts. Specific and more detailed results, pertaining to issues of interest in Glacier Bay, are reserved for later chapters.

7.1 Water Surface Elevation

Water surface elevation data can be written in one of two ways. First, data can be written at only specified points (fort.61 files). Second, data can be written at *all* nodal points in the mesh (fort.63 files). The former is useful

if the user wants to inspect a highly-resolved time series of elevation (much like a tide gage) while the latter is useful if the user wishes to plot a contour map of elevation over the entire (or a portion) domain.

With regards to temporal resolution, note that, since fort.61 only outputs data at (presumably limited) requested points, data may be requested at a very small time interval while still yielding an output file of modest size. As a point of comparison, NOAA frequently provides water elevation data at 6 minute intervals. For the purposes of comparison (model output to data), therefore, there is no need to request time series output at a smaller time step than this.

In principle, one can extract time series of elevation from the global fort.63 file, but the temporal resolution of global output is generally limited (say, one half hour to one hour) by file size. Therefore, it is recommended that the fort.61 output option be used when time series at limited points are desired.

7.1.1 Time Series of Elevation

The Matlab script read_fort61.m is used to load and plot the time series data written to fort.61. When executed, the user first loads the specified files and is then presented with a graphical rendering of the domain (Fig. 7.1. Symbols indicate the locations (as was specified in the fort.15 file for that run) where time series elevation data was requested. The user can then click on any number of these stations in order to select a subset for plotting. Upon completion of this selection, time series from the selected stations are plotted in a second figure (Fig. 7.2. The horizontal axis is in days and is referenced to 12:00 a.m. (GMT) of the first day of the simulation period.

7.1.2 Contour Plots / Animations of Elevation

While the time series are helpful in understanding the tides at a give point or points, it is also useful to visualize the tides as a 'field variable.' In other words, it is helpful to make a two-dimensional plot or 'snapshot' of the water surface elevation at a given time.

The Matlab script view_elevation.m allows the user to view such a snapshot at any of the times that were written to the fort.63 file. Additionally, the program allows the user to create an animation of the elevation fields over the entire duration of the simulation. This animation can then be written out to an .avi file, which is readily viewed by a number of graphics applications.

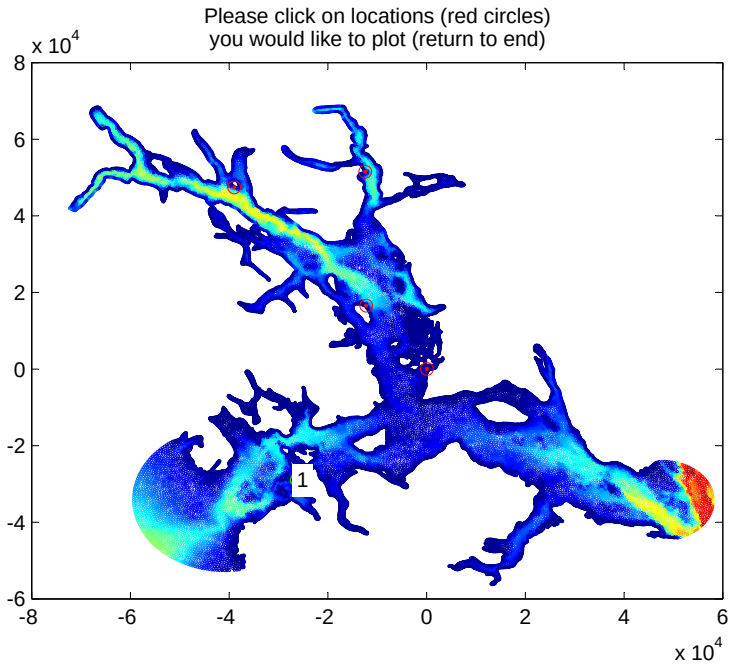

Figure 7.1: User selection of the fort.61 stations to plot as time series.

Upon execution of this script, the user is prompted to load the related data files and then is asked to choose between making a single contour plot of the elevation at a given time or an animation. In the case of the animation, the user will need to (if desired) zoom in on the region of interest before proceeding. Additionally, it is necessary for the user to specify expected minimum and maximum elevation, as these values set the range of the 'colorbar' used for visualizing the results. Values on the order of -3 m and 3 m work well in most cases for the Glacier Bay domain. Finally, if an animation is selected, the user is prompted to specify the file name (.avi format) to which the animation should be saved. A sample 'snapshot' of elevation is given in Fig. 7.3. Note that, in the case of the animation, the script is fairly slow as a tremendous amount of data has to be read and then rendered to the screen as graphical output.

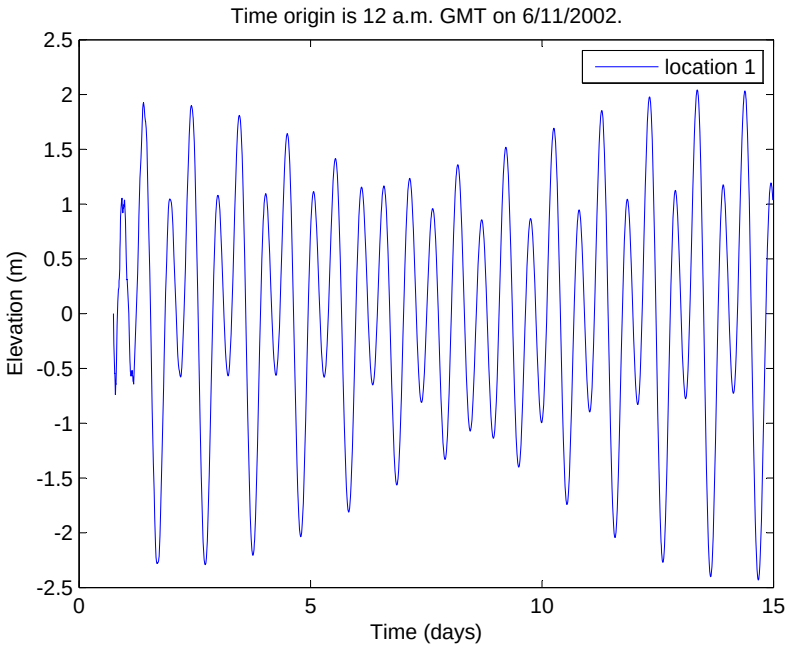

Figure 7.2: Time series of water surface elevation for the selected station(s).

7.2 Two Dimensional Velocity Fields

The work described in this report comes from ADCIRC runs in 'two-dimensional' mode. This means that, at each grid point, a two-dimensional horizontal velocity vector is output. This velocity vector represents the vertical average over the water column. For a system, such as Glacier Bay, where significant salinity gradients exist, a fully three-dimensional approach would be ideal and this is the subject of ongoing work.

As was the case with elevation, velocity information can be reported in one of two ways. The first, contained in the fort.62 file, reports velocity at specified locations only. This would be useful, for example, for comparing calculations to data obtained from a moored velocity meter, such as an acoustic doppler current profiler. The second, contained in the fort.64 file, reports velocity in the 'global' sense, i.e. at all grid elements. In this latter case, it is very useful to make a plot where the two-dimensional velocity *vectors* are displayed. This provides the user with a good grasp of the flow field at the given time.

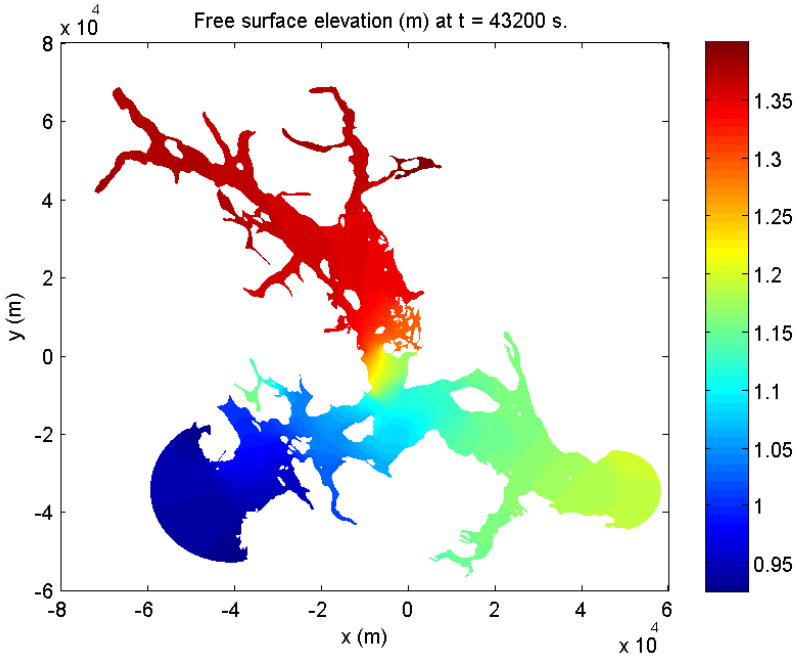

Figure 7.3: Sample contour plot of free surface elevation at a given simulation time.

The Matlab script view_global_velocity.m has many of the same features as the view_global_elevation.m script described above. Upon execution, it will prompt the user to load the fort.14, fort.15, and fort.64 files. The user can then select between a plot at a single time step, or an animation of all of the results available in the fort.64 file. In the case of a static plot, the entire domain is plotted and the user can subsequently go in and zoom in / out and pan around the domain, in order to explore the results more closely.

In both cases (static plot and animation), the user has the choice of plotting, along with the velocity vectors, contours of either the bathymetry, the water speed, or the vorticity. Vorticity is a measure of the 'rotation' of a fluid and regions of high vorticity are indicative of swirling motions and possibly eddies. For the case of a static plot, the minimum and maximum limits of the contour plot are set by the global minimum and maximum from the entire domain. For an animation, the user will have to specify expected minimum and maximum values. While this may take a bit of practice and experience, it is necessary to ensure that all contour plots throughout the

animation are on an identical scale. Figures 7.4-7.5 illustrate sample results in the area of Sitakady Narrows.

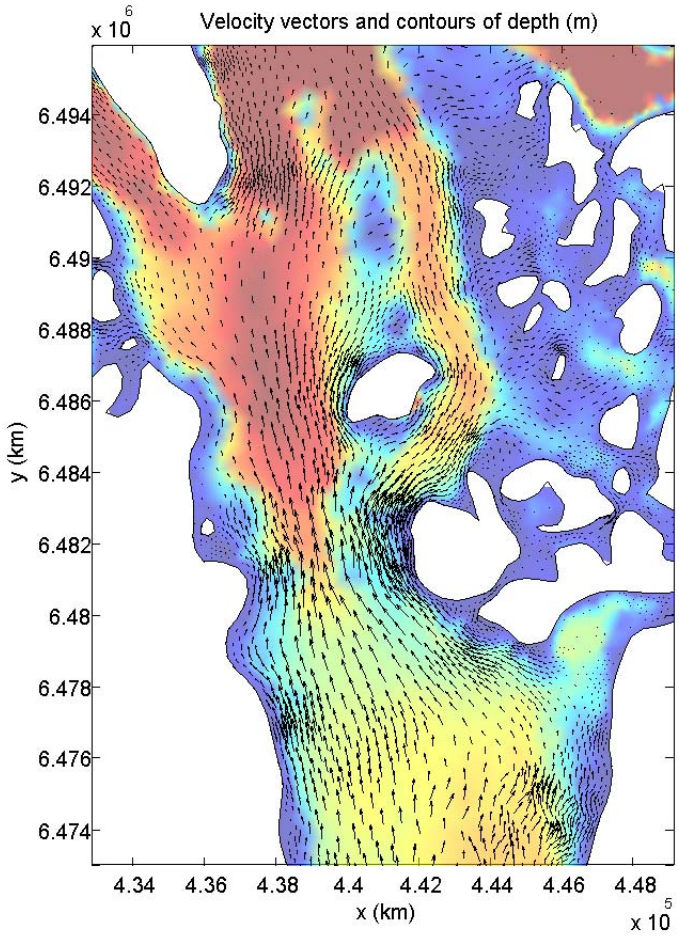

Figure 7.4: Sample velocity vectors showing the two-dimensional flow field and contours of bathymetric depth in the Sitakady Narrows area.

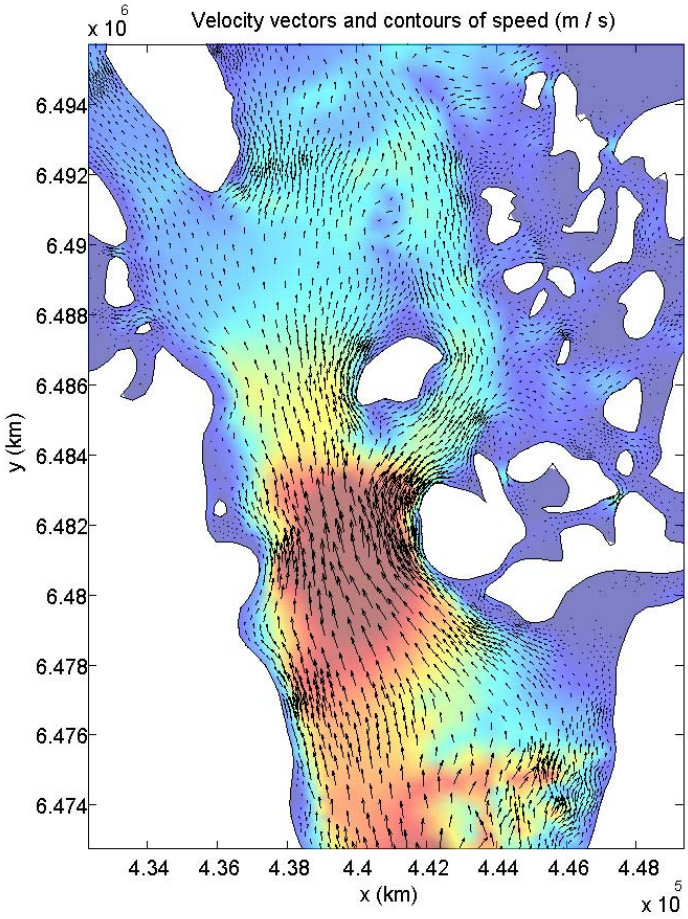

Figure 7.5: Sample velocity vectors showing the two-dimensional flow field and contours of water speed in the Sitakady Narrows area.

7.3 Particle Trajectories

Finally, it is of great interest to be able to predict that path, or trajectory, taken by an identified parcel of water over time. This concept brings into focus the difference between Eulerian and Lagrangian approaches to mechanics. An Eulerian approach calculates output variables as 'field variables.' In other words, velocity or elevation information gets reported at fixed points in space. In this framework, a flow variable is a function of both space and time.

A Lagrangian approach, on the other hand, identifies what are called 'material elements' and tracks them as they move. In this case, which is precisely the case of a released drifter or drogue, velocity information is solely a function of time and of the initial position of the material element.

For an environmental hydraulics application, there are a number of reasons why it would be of interest to calculate trajectories taken by marked particles. One might be interested in the transport of a spilled contaminant, or of a passive biological species. In any case, provided that the assumption that the tracer of interest follows the flow is a suitable one, the velocity fields output by ADCIRC may be used to calculate trajectories over time.

Before illustrating this, there are a few practical considerations to take into account. ADCIRC generally utilizes a very small time step in making its calculations. However, and as has been discussed previously, results are not written out at each of these time steps. If that were the case, then output files would be excessively large. Instead, the model outputs data at a user-specified interval. Given the large mesh for the present application, a fort.64 file containing ASCII output at 100 times results in a file of approximately 150 MB. If these times span a single day, then the time resolution of the output is on the order of 15 minutes.

The issue of consequence here has to do with the calculation of Lagrangian particle paths based upon Eulerian data at relatively coarse temporal resolution. Recall first that the semi-diurnal tide repeats every 12 hours. If velocity data were only available every hour, calculated trajectories would demonstrate significant error. There is, therefore, a balance to be struck between managable file size and requirements for accuracy.

7.3.1 Numerical Integration

Next, consider a very brief introduction on exactly how particle paths may be calculated from the velocity fields in the fort.64 file. The two horizontal velocity components are defined by

$$\frac{dx}{dt} = u \equiv f_1(\underline{x}, t) \tag{7.1}$$

$$\frac{dy}{dt} = v \equiv f_2(\underline{x}, t). \tag{7.2}$$

The functions f_1 and f_2 are called 'derivative functions.' These differential equations are continuous and smoothly varying in time. However, since we have information only at *discrete* points in time (separated by the user-specified interval), we choose to replace the differential equations with equivalent difference equations.

Euler Method

The simplest way of doing this is to replace, for example,

$$\frac{dx}{dt} \rightarrow \frac{x_{n+1} - x_n}{\Delta t}, \tag{7.3}$$

where Δt is the time interval and the subscripts refer to the time step. Thus, the equation is rearranged to obtain

$$x_{n+1} = u_n \Delta t + x_n. \tag{7.4}$$

This so-called 'explicit Euler method' states that the position of a particle at the 'next' time step can be computed from the position and velocity of the particle at the present time step. While this method is extremely easy to implement, it is relatively low-order and can lead to large errors accumulated over time, especially if Δt is too large.

Consider as a simple example the case of 'stagnation flow.' This steady, two-dimensional flow field is given by the velocity components $u = y$ and $v = -x$. Based upon an initial location of x_0, y_0, it is straightforward to show that the exact answer for the trajectory taken by a particle upon its release is given by the equation $xy = x_0 y_0$. This exact answer, along with two approximate trajectories, as calculated by the Euler method, is shown in Fig. 7.6. Note that the flow comes in from the top and exits to the right. Note also the significant variation of the trajectory with time step.

Runge Kutta Method

A far more accurate method than the first-order Euler method is the fourth order Runge-Kutta method. With this method, we have (using, as above, the x component of velocity as an example)

$$x_{n+1} = x_n + \frac{1}{6}(\Delta x_1 + 2\Delta x_2 + 2\Delta x_3 + \Delta x_4), \tag{7.5}$$

where

$$\Delta x_1 = \Delta t f_1(x_n, t_n) \tag{7.6}$$

$$\Delta x_2 = \Delta t f_1\left(x_n + \frac{\Delta x_1}{2}, t_n + \frac{\Delta t}{2}\right) \tag{7.7}$$

$$\Delta x_3 = \Delta t f_1\left(x_n + \frac{\Delta x_2}{2}, t_n + \frac{\Delta t}{2}\right) \tag{7.8}$$

$$\Delta x_4 = \Delta t(x_n + \Delta x_4, t_n + \Delta t). \tag{7.9}$$

Similar equations exist for y. As is evident from this formulation, calculation of x_{n+1} now requires more than just x_n and $u_n = f_1|_n$. Sample results, for the same time steps used in Fig. 7.6, are given in Fig. 7.7. Note that, even for the very coarse time step, the approximate results are in excellent agreement with the exact results.

7.3.2 Application to ADCIRC Output

The Matlab script trajectory.m was written in order to allow the user to visualize the paths taken by identified material elements. Presently, there are a number of ways in which the initial locations of the elements may be set. When the script is executed, and the grid and model run files (fort.14 and fort.15) are loaded, the user is presented with the option of placing individual particles, a 'cloud' of particles, or a 'rake' (evenly distributed along a line) of particles. Note that it is possible to combine multiple instances of all of these release methods.

When the user is finished, the velocity data is loaded and the Lagrangian paths taken by the particles are computed via the Runge-Kutta method. Figure 7.8 illustrates the consequences of relying on the less-accurate Euler method. For this example, the global velocity output is at 5 minute intervals. Many of the trajectories show a steadily accumulating discrepancy

between the Euler and Runge-Kutta methods. One, in particular, shows how these small errors can, depending on the bathymetric details, lead to wildly different ending locations for the particles.

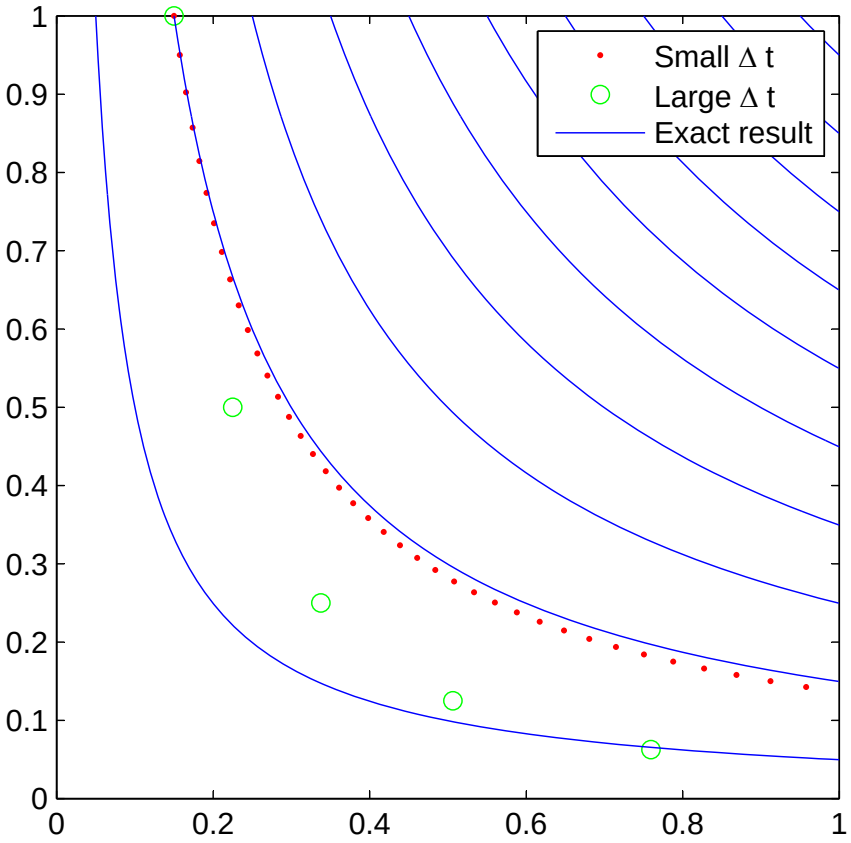

Figure 7.6: Sample particle trajectories, based upon the Euler method, using small and large time steps. Also shown are the exact trajectories.

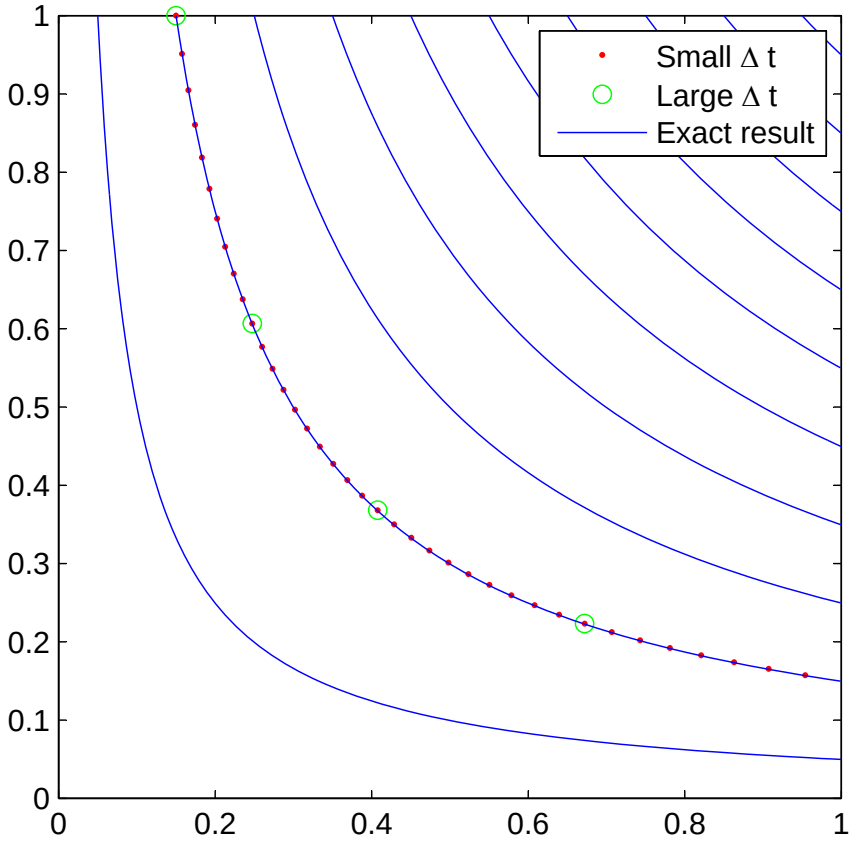

Figure 7.7: Sample particle trajectories, based upon the Runge-Kutta method, using small and large time steps. Also shown are the exact trajectories. Note that the time steps used are the same as in Fig. 7.6.

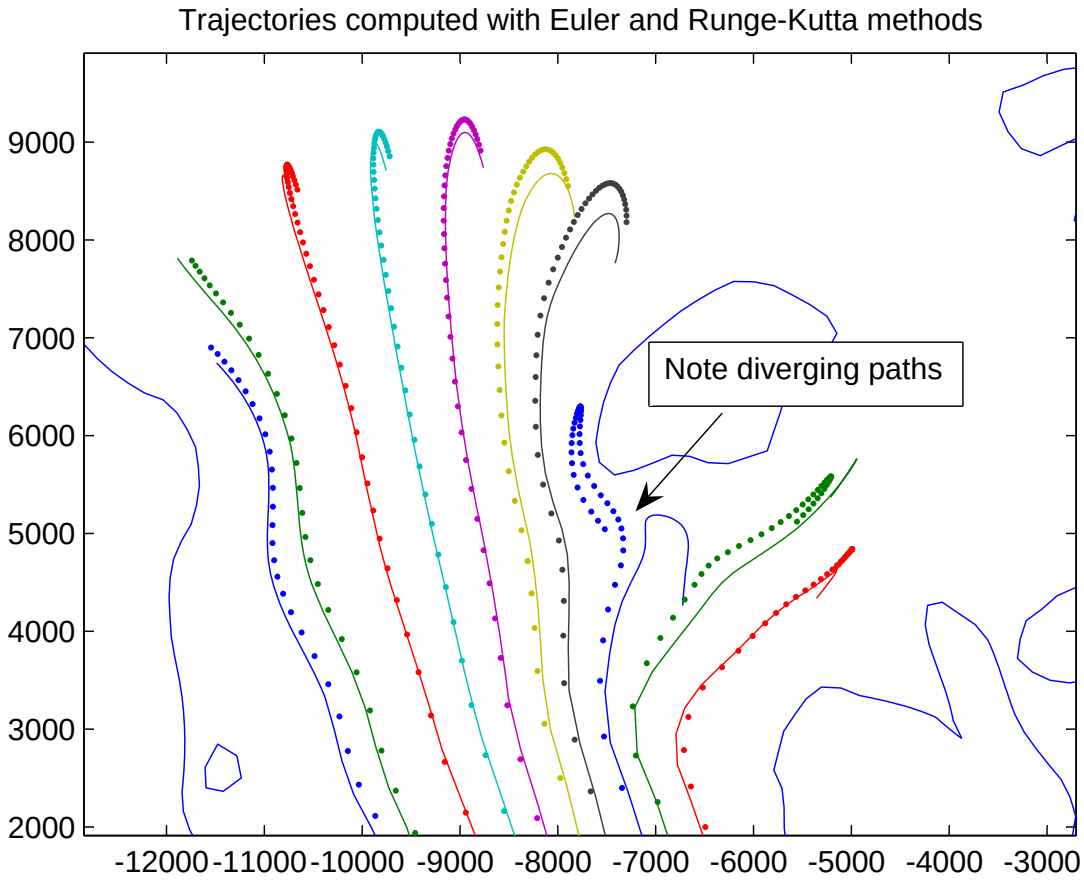

Figure 7.8: Particle trajectories, in the Beardslee Islands area, calculated with both the Euler (solid lines) and the Runge-Kutta methods (dots).