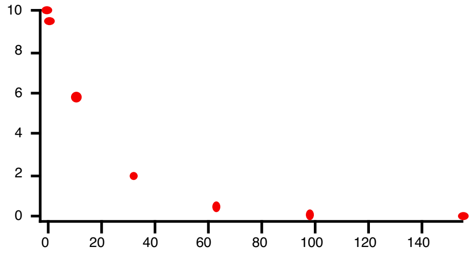
And finally do the integration in free-run mode. The /XRUN flag specifies a suggested first step size and the maximum X value. When the solution passes the maximum X value (100 in this case) or when your waves are filled, IntegrateODE will stop.

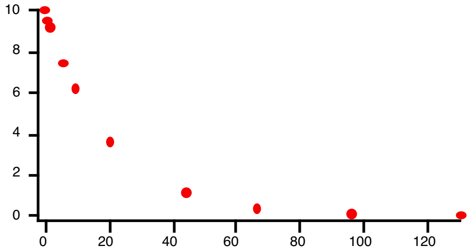
```
FreeRunX = NaN;FreeRunX[0] = 0
IntegrateODE/M=1/X=FreeRunX/XRUN={1,100} FirstOrder, PP, FreeRunY
```

In the earlier example, we (rather arbitrarily) chose 100 steps to make a reasonably smooth plot. In this case, it took 6 steps to cover the same X range, and the steps are closest together at the beginning where the exponential decay is most rapid:



Asking for more accuracy will cause smaller steps to be taken (9 when we executed the following command):

```
FreeRunX = NaN;FreeRunX[0] = 0
IntegrateODE/M=1/X=FreeRunX/XRUN={1,100}/E=1e-14 FirstOrder, PP, FreeRunY
```



After IntegrateODE has finished, you can use Redimension and the V_ODETotalSteps variable to adjust the size of the waves to just the points actually calculated:

```
Redimension/N=(V_ODETotalSteps+1) FreeRunY, FreeRunX
```

Note that we added 1 to V_ODETotalSteps to account for the initial value in row zero.

Stiff Systems

Some systems of differential equations involve components having very different time (or decay) constants. This can create what is called a “stiff” system; even though the short time constant decays rapidly and contributes negligibly to the solution after a very short time, ordinary solution methods (/M = 0, 1, and 2) are unstable because of the presence of the short time-constant component. IntegrateODE offers the Backward Differentiation Formula method (BDF, flag /M=3) to handle stiff systems.

A rather artificial example is the system (see “Numerical Recipes in C”, edition 2, page 734; see **References** on page III-349)

```
du/dt = 998u + 1998v
dv/dt = -999u - 1999v
```

Here is the derivative function that implements this system:

```
Function StiffODE(pw, tt, yy, dydt)
   Wave pw      // not actually used because the coefficients
                // are hard-coded to give a stiff system
   Variable tt
```