Integrators in MATLAB

What are Integrators?

Integrators are algorithms that approximate a trajectory in a system based on initial conditions and knowledge of the equations of motion for that system. The basic idea is that a trajectory may be constructed by integrating its derivative, hence the name *integrator*.

Integrators may be as accurate as desired given perfect precision, arbitrarily large expansions, and arbitrarily small time-steps. But there is a trade-off between an integrator's performance and how long it takes to perform the integration.

In general, integrators use a Taylor expansion to approximate the dynamics in the system. Euler's Method invokes a 1^{st}-order expansion, which works swiftly but with generally poor accuracy. The midpoint method invokes a 2^{nd}-order expansion, which approximates the dynamics much better but still with noticeable errors (i.e., the energy in the system tends to change when it shouldn't). On the other hand, a very high order expansion, e.g., 8^{th}-order, works very well at approximating the dynamics for a very long amount of time, but takes a long time to operate. **The most popular compromise is the Runge-Kutta integrator, which invokes a 4^{th}-order expansion.**

MATLAB's Integrators

Solver	Problem Type	Order of Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. This should be the first solver you try.
ode23	Nonstiff	Low	For problems with crude error tolerances or for solving moderately stiff problems.
ode113	Nonstiff	Low to high	For problems with stringent error tolerances or for solving computationally intensive problems.
ode15s	Stiff	Low to medium	If ode45 is slow because the problem is stiff.
ode23s	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
ode23t	Moderately Stiff	Low	For moderately stiff problems if you need a solution without numerical damping.
ode23tb	Stiff	Low	If using crude error tolerances to solve stiff systems.

ode45.m

MATLAB's pre-built ode45.m integrator is generally the best option. The basic command to call the ode45 integrator looks like this:

```
[t,state] = ode45(@dstate,time,ICs,options);
```

The integrator takes a vector of initial conditions (either a column or row vector) and integrates it using the dynamics given in the *dstate* function. It outputs the approximate trajectory at every time-point given in the vector ***tspan***. The outputs include **t**, the vector of time values that ode45 did integrate to (in general the same vector as **tspan**), and **state**, which is an array containing the state at every point in **tspan**.

Summarized:
Inputs:

- `@dstate:` The name of the file that contains the derivative functions (discussed in the next section);
- `tspan:` The vector of time values to integrate;
- `ICs:` The vector of initial conditions for the system (either row or column);
- `options:` An optional specification for the options given to the integrator (tolerance, etc.).

Outputs:

- `t:` The vector of times that the integrator did produce (usually the same as `tspan`);
- `state:` An array containing the values of the state at every time.

The `dstate` Function

```
function dRV = dstate(time,RV)

x  = RV(1);
y  = RV(2);
z  = RV(3);

constants = (declare what constants you are using)

ax = Some function of x,y,z,time,constants
ay = Some function of x,y,z,time,constants
az = Some function of x,y,z,time,constants

% The function will return the following variables in a vector
dRV = [ax; ay; az];
```

The function must be named '`dstate.m`' (or whatever, so long as the first line of the function reflects its name and the ode45 call has the right function name). It's probably easiest to declare all the constants in the function. If the constants change over the trajectory, you may need to declare the constants global in both the main program and the derivative function. That way, they will change as needed.

The `options` object

There are various options that may be set in ode45. Type "`help odeset`" to see all of the available options. The most important ones concern setting the relative and absolute tolerance for the system. To do that, use the following snippet of code:

```
% Tolerance
tolerance = 1e-12;

% Setting up ODE45
options = odeset('RelTol',tolerance,'AbsTol',tolerance);
```