

Ordinary Differential Equations

Computational Physics

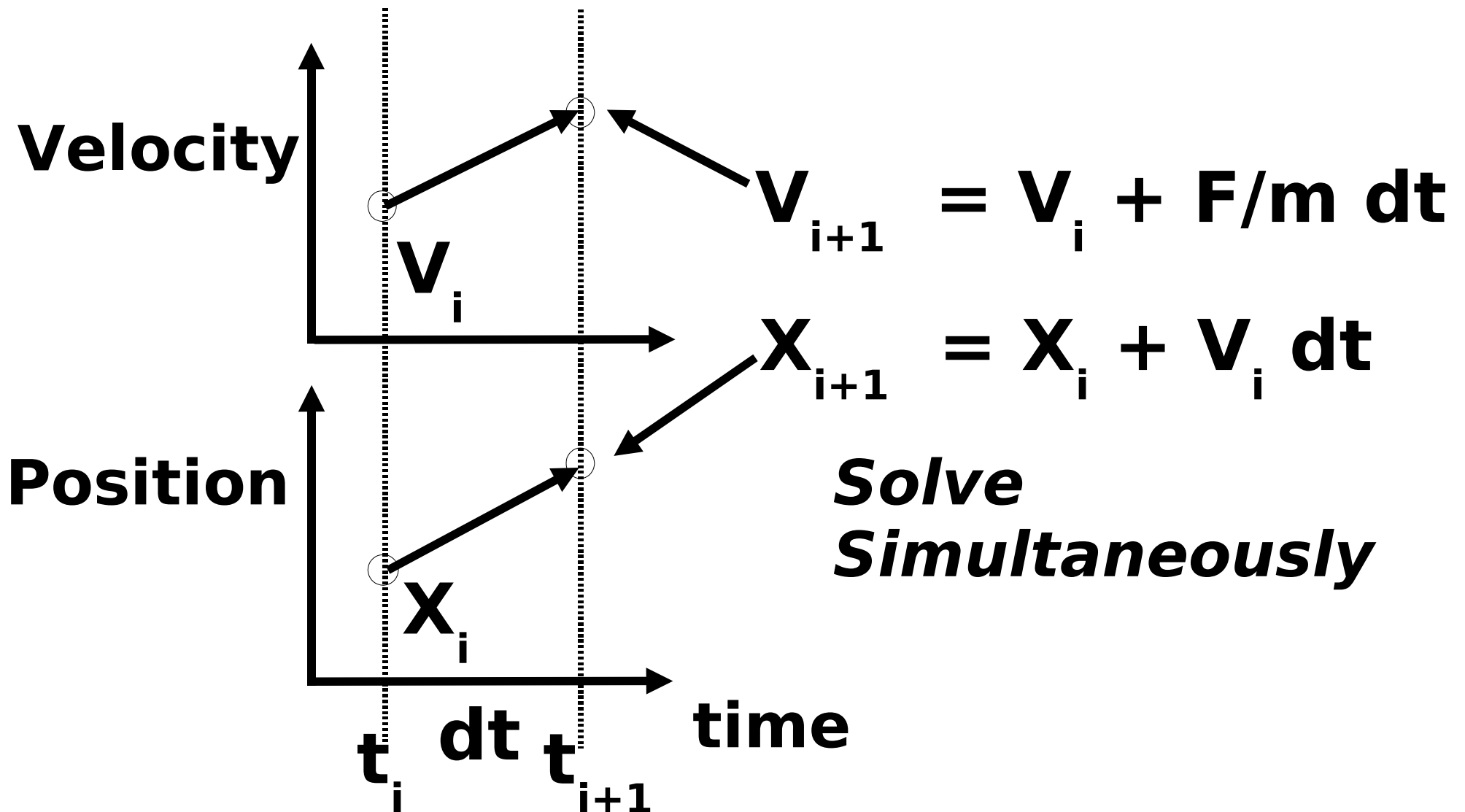
Ordinary Differential Equations
Improvements on the Euler Method

Outline

- Problems with Euler Method
- Runge-Kutta Approach (Second Order)
- Accuracy and Roundoff Error

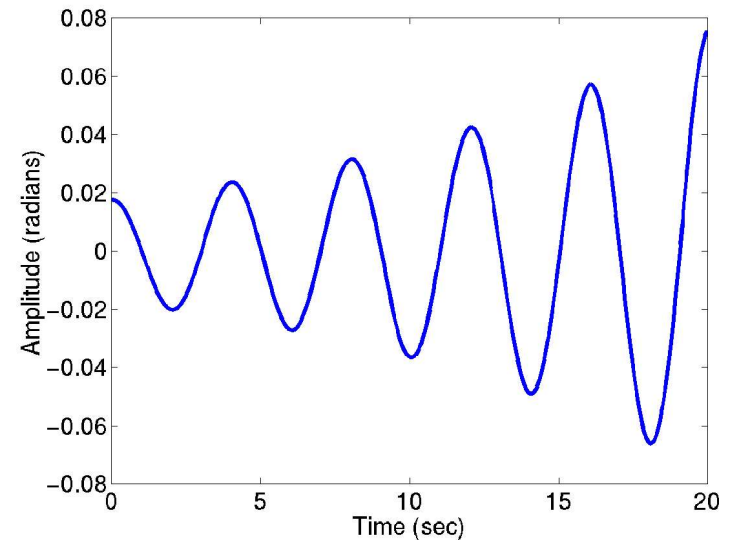
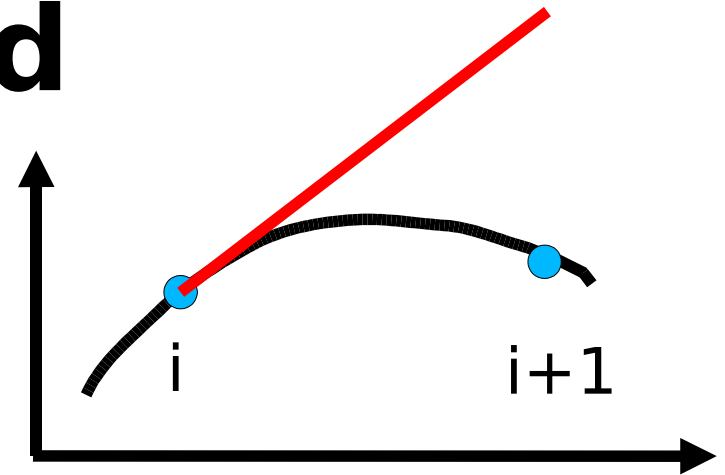
Second Order DEQ

Euler Method



Problems with Euler Method

- Not a good approximation
 - assumes derivative is constant in interval
- Energy is not conserved
 - periodic phenomena can't be explored
- Solution is to devise new methods

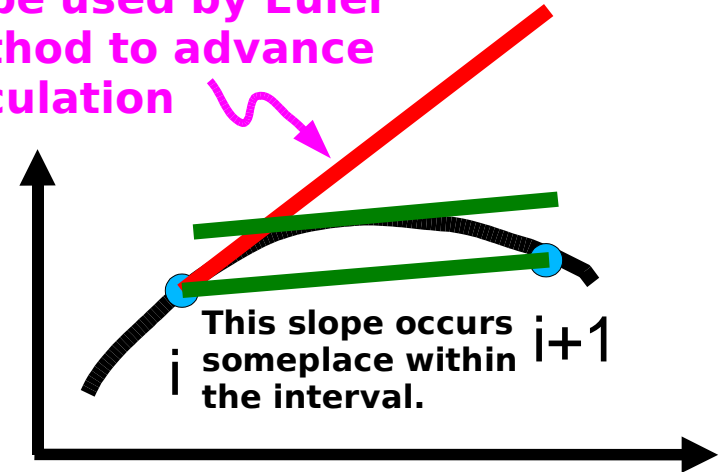


Solution for Pendulum with Euler Method

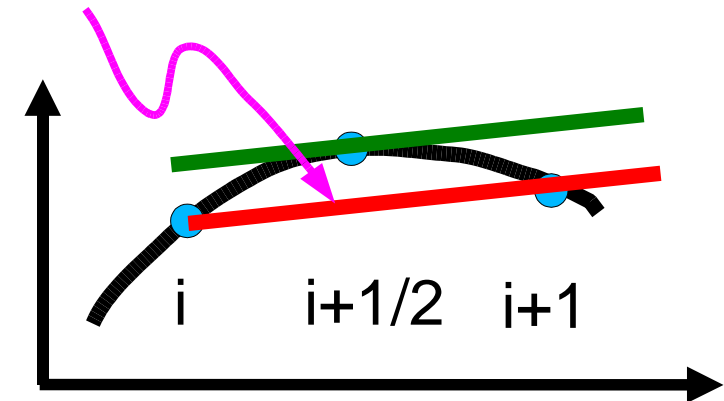
Runge-Kutta Approach

- Mean Value Theorem
- Estimate typical derivative *within* the interval by finding its value at some point within the interval.
- Second Order Runge-Kutta: estimate *typical* value of derivative from its value at half-way point

Slope used by Euler Method to advance calculation



Runge-Kutta uses slope found at midpoint to advance calculation



Second Order Runge-Kutta

$$\frac{dy}{dt} = f(y, t)$$

**Differential
Equation**

$$y\left(t + \frac{\Delta t}{2}\right) = y(t) + f(y(t), t) \frac{\Delta t}{2}$$

**Estimate value
of y at half-step
(Euler Method)**

$$y(t + \Delta t) = y(t) + f\left(y\left(t + \frac{\Delta t}{2}\right), t + \frac{\Delta t}{2}\right) \Delta t$$

**Use value at
half-step to
find new estimate
of derivative**

Second Order DEQ

Runge-Kutta (2nd Order)

Compute estimate of V and X at midpoint:

$$v_{i+1/2} = v_i + \frac{F(x_i, v_i) dt}{m} \quad x_{i+1/2} = x_i + v_i \frac{dt}{2}$$

Use midpoint values to move ahead:

$$v_{i+1} = v_i + \frac{F(x_{i+1/2}, v_{i+1/2}) dt}{m} \quad \text{VELOCITY}$$

$$x_{i+1} = x_i + v_{i+1/2} dt \quad \text{POSITION}$$

% Example: Falling Ball with Second Order Runge-Kutta

```
clear
g = 9.8;           % gravitational acceleration
X0 = 0;           % initial value: position
V0 = 0;           % initial value: velocity
dt = 2;           % step size
t = 0:dt:100;     % array of times for calculation
```

```
X = zeros(length(t),1); % define solution arrays
```

```
V = zeros(length(t),1);
```

```
X(1) = X0;
```

```
V(1) = V0;
```

```
for i=1:length(t)-1
```

```
    Vmid = V(i) - g*dt/2;
```

```
    V(i+1) = V(i) - g*dt;
```

```
    X(i+1) = X(i) + Vmid*dt;
```

```
end
```

```
% make some nice graphs
```

```
plot(t,X,'md')
```

```
hold on
```

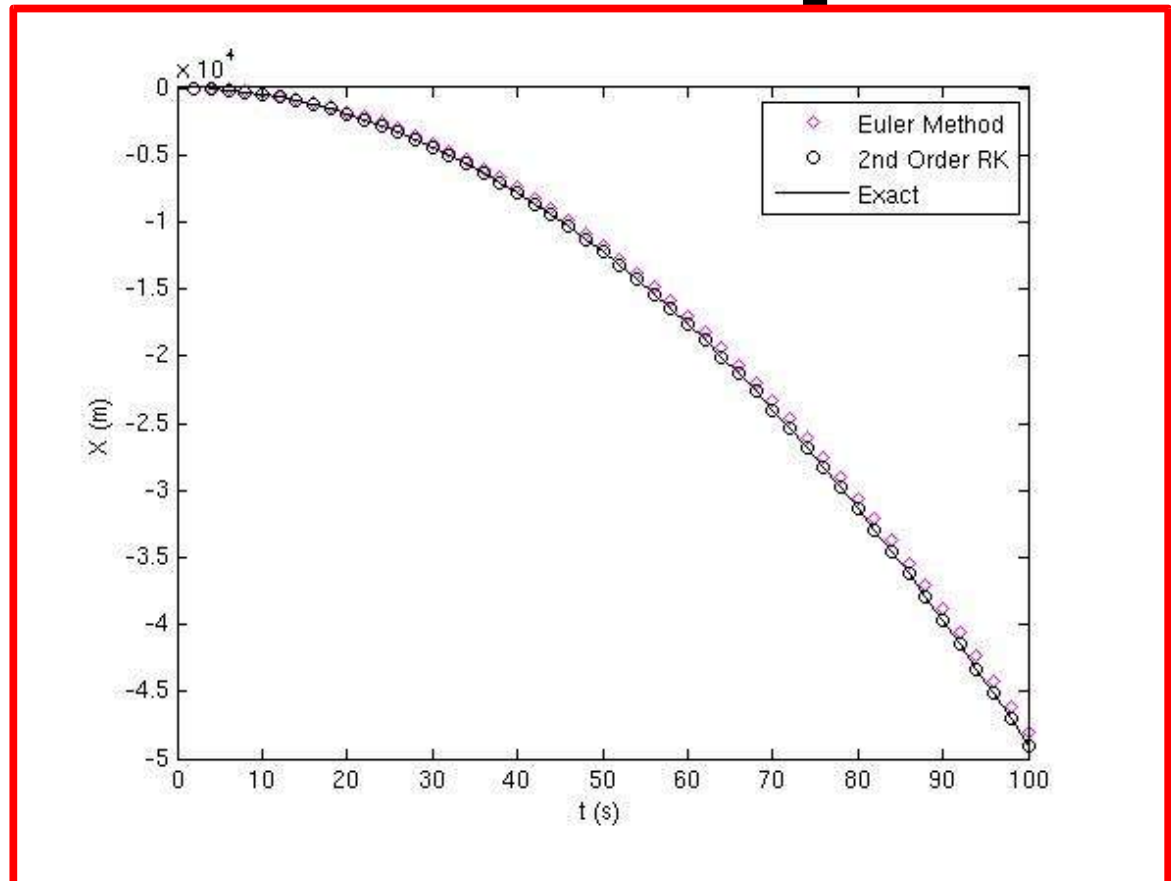
```
plot(t,X0+V0*t-0.5*g*t.*t,'k')
```

```
hold off
```

```
xlabel('t (s)');
```

```
ylabel('X (m)');
```

```
legend('Euler Method','Exact')
```



Second Order Runge-Kutta

Consider Single Step with Falling Ball

$$v_{mid} = v_0 - g \frac{dt}{2}$$

$$x(dt) = x_0 + v_{mid} dt$$

$$x(dt) = x_0 + v_0 dt - \frac{1}{2}g dt^2$$

Recall that velocity equation was exact for Euler.

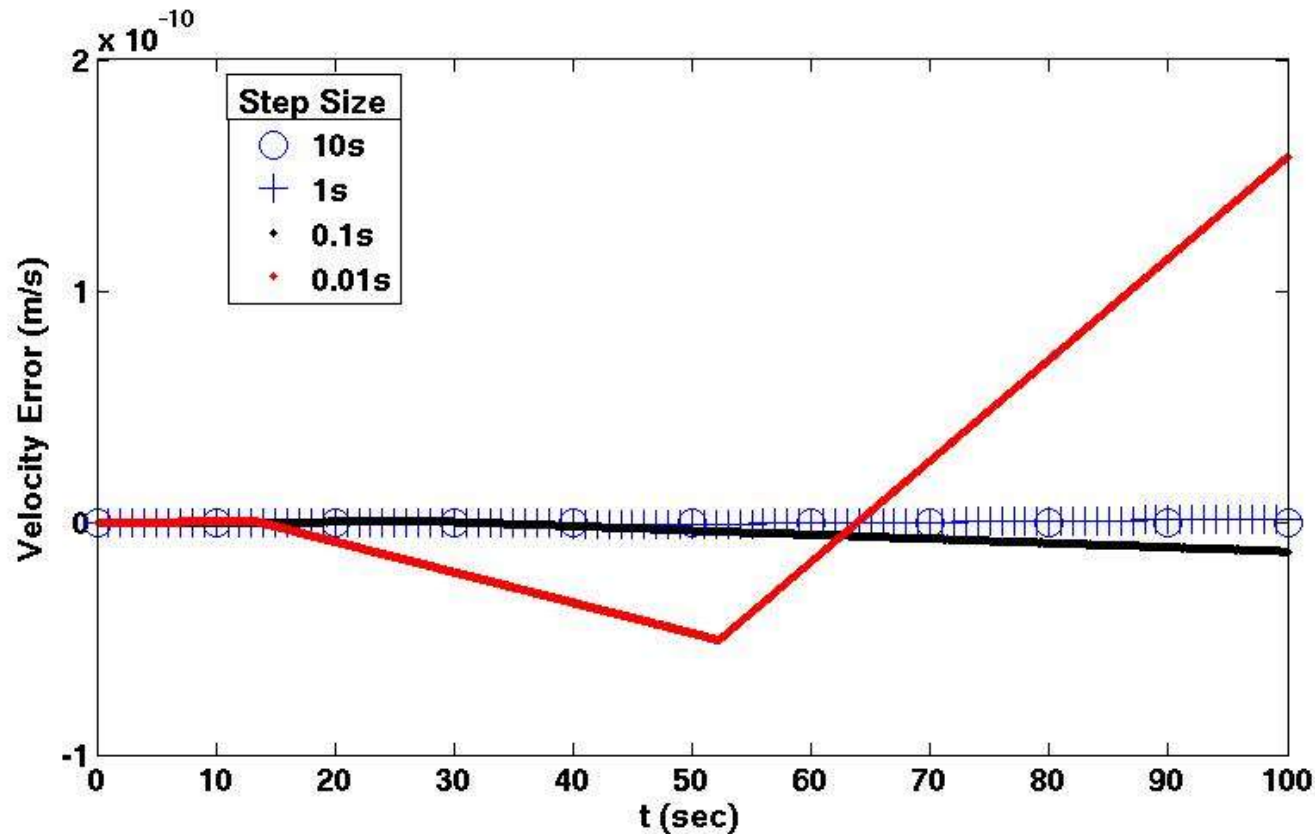
Position equation is now also exact with Runge-Kutta!

$$E(dt) = mg(x_0 + v_0 dt - \frac{1}{2}g dt^2) + \frac{1}{2}m(v_0 - g dt)^2$$

$$E(dt) = mgx_0 + \frac{1}{2}mv_0^2$$

and Energy is conserved!

Accuracy Part II



Velocity equation is supposed to be exact ... but ... comparing solution to the true answer we see that there are small errors! Worse yet, the errors get BIGGER when we take smaller steps. What's going on??

Roundoff Error

- Computers represent numbers with a *finite* number of bits, so all numbers contain small errors: $a' = a(1 + e_a)$

- Subtraction: $c = a - b$

$$c' = a' - b' = a(1 + e_a) - b(1 + e_b)$$

$$c'/c = 1 + a/c e_a - b/c e_b$$

NOTE if a, b are big and $a \sim b$ then c is small and the error can be large: $c'/c \sim 1 + a/c (e_a - e_b)$

- Multiplication: $c = a * b$

$$c' = a' * b' = a(1 + e_a) * b(1 + e_b)$$

$$c'/c \sim 1 + e_a + e_b \text{ (relative error in } c)$$

NOTE successive multiplications add errors like a random walk. In this case, the total error increase goes as $\sqrt{\text{number of steps}}$