# Numerical Methods

## Week 4 code

==========

```
function [ t, x ] = ode_euler(f, t0, T, x0, N)
%ODE_EULER - Euler scheme for ODEs
% t0 - starting time
% T  - final time
% x0 - initial condition at t0
% N  - number of iterations

m = length(x0);
dt = (T-t0)/N;
t = [t0: dt : T];
x = zeros(m, N+1);
x(:,1) = x0;

for k=1:N
    x(:,k+1) = x(:,k) + dt * f(x(:,k), t(k));
end

end
```

==========

```
function [ t, x ] = ode_midpoint(f, t0, T, x0, N)
%ODE_MIDPOINT - Mid point scheme for ODEs
% f  - handle to function which describes the right hand side of ODE
% t0 - starting time
% T  - final time
% x0 - initial condition at t0
% N  - number of iterations

m = length(x0);
dt = (T-t0)/N;
t = [t0: dt : T];
x = zeros(m, N+1);
x(:,1) = x0;

for k=1:N
    z = x(:,k)+0.5*dt*f(x(:,k), t(k));
    x(:,k+1) = x(:,k) + dt * f(z, t(k)+0.5*dt);
end
```

```
end
```

==========

```
function [ t, x ] = ode_trapezoid(f, t0, T, x0, N)
%ODE_TRAPEZOID - Trapezoid scheme for ODEs
% f  - handle to function which describes the right hand side of ODE
% t0 - starting time
% T  - final time
% x0 - initial condition at t0
% N  - number of iterations

m = length(x0);
dt = (T-t0)/N;
t = [t0: dt : T];
x = zeros(m, N+1);
x(:,1) = x0;

for k=1:N
    w = dt * f(x(:,k),t(k));
    z = x(:,k) + w;
    x(:,k+1) = x(:,k) + 0.5*w + 0.5*dt*f(z, t(k+1));
end

end
```

==========

```
function y = lorenz(x, t)
%LORENZ - Lorenz equations

sigma = 10.;
tau   = 28.;
beta  = 8./3.;

y = [sigma*(x(2)-x(1)); (tau-x(3))*x(1)-x(2); x(1)*x(2)-beta*x(3)];
end
```

==========