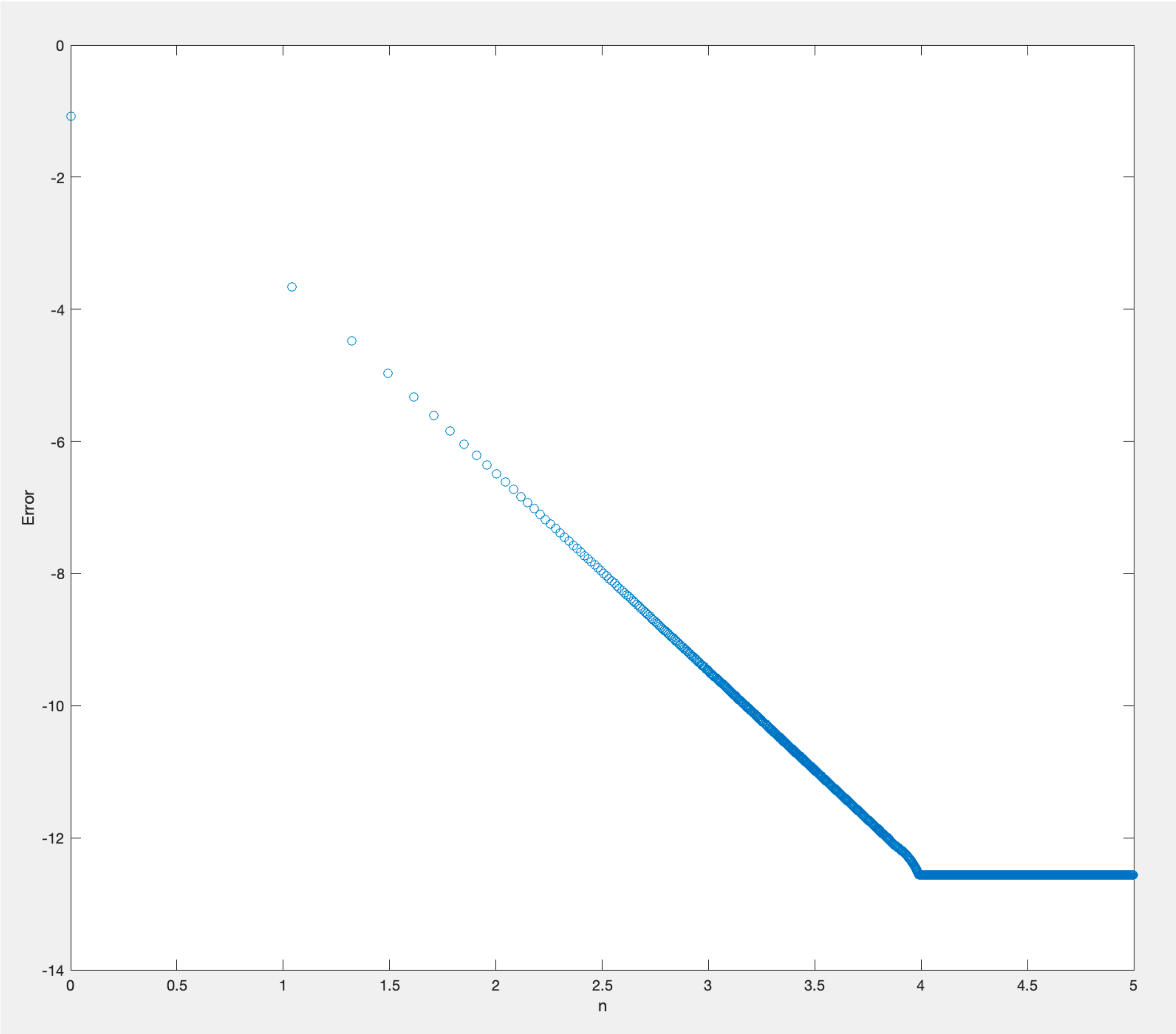
Problem 1




```
function y = CRzeta(n)
% y(0) = 0;
% for i = 1:n
%     y(i) = y(i-1) + 1/(n.^4);
% end

i = 1:n;
x(i) = i.^(-4);
xsums = cumsum(x);
y = xsums(1:1:length(x));
y = y(n);
```

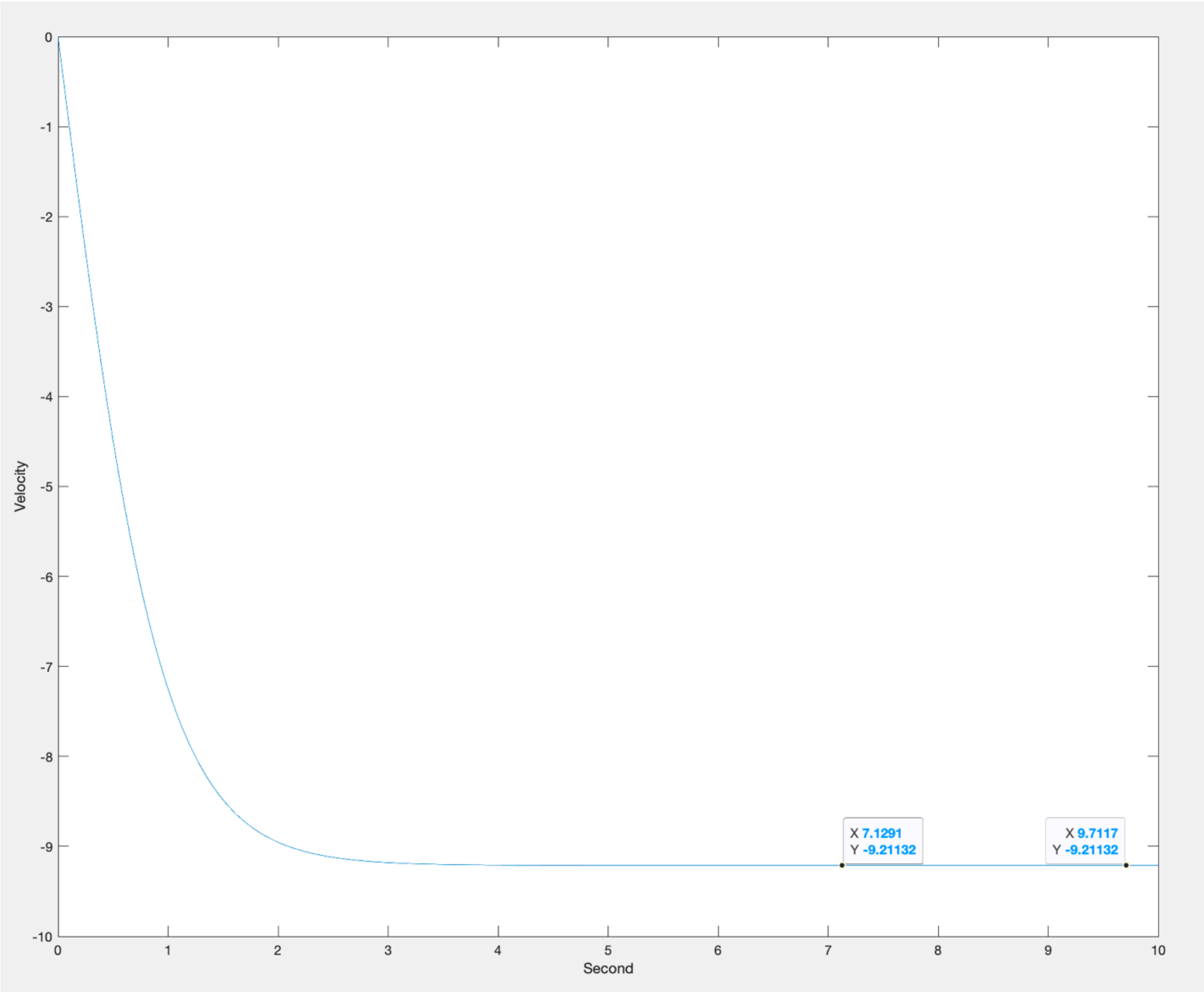
```
function test_zeta()
```

```
N = 1e5;
i = 0;
for n = 1:10:N
    error(n) = abs(CRzeta(n) - (pi.^4)/90);
    i = i + 1;
    err(i) = error(n);
end
n = 1:10:N;
n = log(n)/log(10);
err = log(err)/log(10);
plot(n,err,'O')
xlabel('n')
ylabel('Error')
```

- Please explain why the error drops with increasing N until it levels off when $N \approx 10000$.

  Because numbers are represented by floating point numbers in a computer. That is how computers perform calculations and is bound to confront the limitation of the representation of real numbers versus a continuous space. Once all floating points are occupied, there will be no more room for trivial ranges of numbers that can be presented on a computer. That illuminates why the error diminishes with booming N until it possesses access to the vicinity of a still status.

Problem 2




```
function f = dvdt(v)
m = 80;
g = 9.8;
rho = 1.2;
A = 14;
C = 1.1;
f = -g + 1/(2*m)*rho*v.^2*A*C;


function y = Forward_velocity(T)
global dt
y(1) = 0;
dt = 1e-4;
for i = 2:T
    y(i) = y(i-1) + dvdt(y(i-1))*dt;
end
```

```matlab
function test_Forward_velocity()
global dt
tol = 1e4;
T = 1e5;
y = Forward_velocity(T);
t = 1:T;
t = t*1e-5;
plot(t,y)
xlabel('Second')
ylabel('Velocity')
for i = 2:length(y)
    if (abs((y(i)-y(i-1))/dt) < tol)
        terminal_velocity = y(i);
        break
    end
end
if (abs(terminal_velocity - (-9.2113)) < tol)
    fprintf('True')
else
    error('False')
end
```

$$m \frac{dv}{dt} = -mg + \frac{1}{2} \rho v^2 A C_d$$

$$\frac{dv}{dt} = -g + \frac{1}{2m} \rho v^2 A C_d$$

$$\frac{dv}{dt} = \frac{1}{\frac{1}{-g + \frac{1}{2m} \rho v^2 A C_d}}$$

$$\frac{dv}{-g + \frac{1}{2m} \rho v^2 A C_d} = dt$$

$$\int \frac{dv}{-g + \frac{1}{2m} \rho v^2 A C_d} = \int dt$$

$$-\frac{1}{2\sqrt{a}\sqrt{g}} \left(\ln\left|\sqrt{\frac{a}{g}} v + 1\right| - \ln\left|\sqrt{\frac{a}{g}} v - 1\right| \right) + C = t$$

where $a = \frac{1}{2m} \rho A C_d$

When $t = 0$, $v = 0$.
Hence, $C = 0$
Rewrite the equation into

$$e^{-\frac{1}{2\sqrt{a}\sqrt{g}} \left(\ln\left|\sqrt{\frac{a}{g}} v + 1\right| - \ln\left|\sqrt{\frac{a}{g}} v - 1\right| \right)} = e^t$$

$$e^{-\frac{1}{2\sqrt{a}\sqrt{g}} \ln\left|\sqrt{\frac{a}{g}} v + 1\right|} \cdot e^{\frac{1}{2\sqrt{a}\sqrt{g}} \ln\left|\sqrt{\frac{a}{g}} v - 1\right|} = e^t$$

$$\left|\sqrt{\frac{a}{g}} v + 1\right|^{-\frac{1}{2\sqrt{a}\sqrt{g}}} \cdot \left|\sqrt{\frac{a}{g}} v - 1\right|^{\frac{1}{2\sqrt{a}\sqrt{g}}} = e^t$$

$$\left|\sqrt{\frac{a}{g}}v+1\right|^{-1}\cdot\left|\sqrt{\frac{a}{g}}v-1\right| = \left(e^{t}\right)^{2\sqrt{a}\sqrt{g}}$$

$$\frac{\left|\sqrt{\frac{a}{g}}v-1\right|}{\left|\sqrt{\frac{a}{g}}v+1\right|} = e^{2t\sqrt{a}\sqrt{g}}$$

Since $\sqrt{\frac{a}{g}}v<0$, $\sqrt{\frac{a}{g}} \approx 9.2113$, $a = 0.1155$

When $v < -\left(\frac{a}{g}\right)^{-\frac{1}{2}}$,

it becomes

$$\frac{-\sqrt{\frac{a}{g}}v+1}{-\sqrt{\frac{a}{g}}v-1} = e^{2t\sqrt{a}\sqrt{g}}$$

$$\frac{-\sqrt{\frac{a}{g}}v-1+2}{-\sqrt{\frac{a}{g}}v-1} = e^{2t\sqrt{a}\sqrt{g}}$$

$$1+\frac{2}{-\sqrt{\frac{a}{g}}v-1} = e^{2t\sqrt{a}\sqrt{g}}$$

$$\frac{2}{-\sqrt{\frac{a}{g}}v-1} = e^{2t\sqrt{a}\sqrt{g}}-1$$

$$-\sqrt{\frac{a}{g}}v-1 = \frac{2}{e^{2t\sqrt{a}\sqrt{g}}-1}$$

$$v = \left(\frac{2}{e^{2t\sqrt{a}\sqrt{g}} - 1} + 1 \right) \left(-\sqrt{\frac{g}{a}} \right)$$

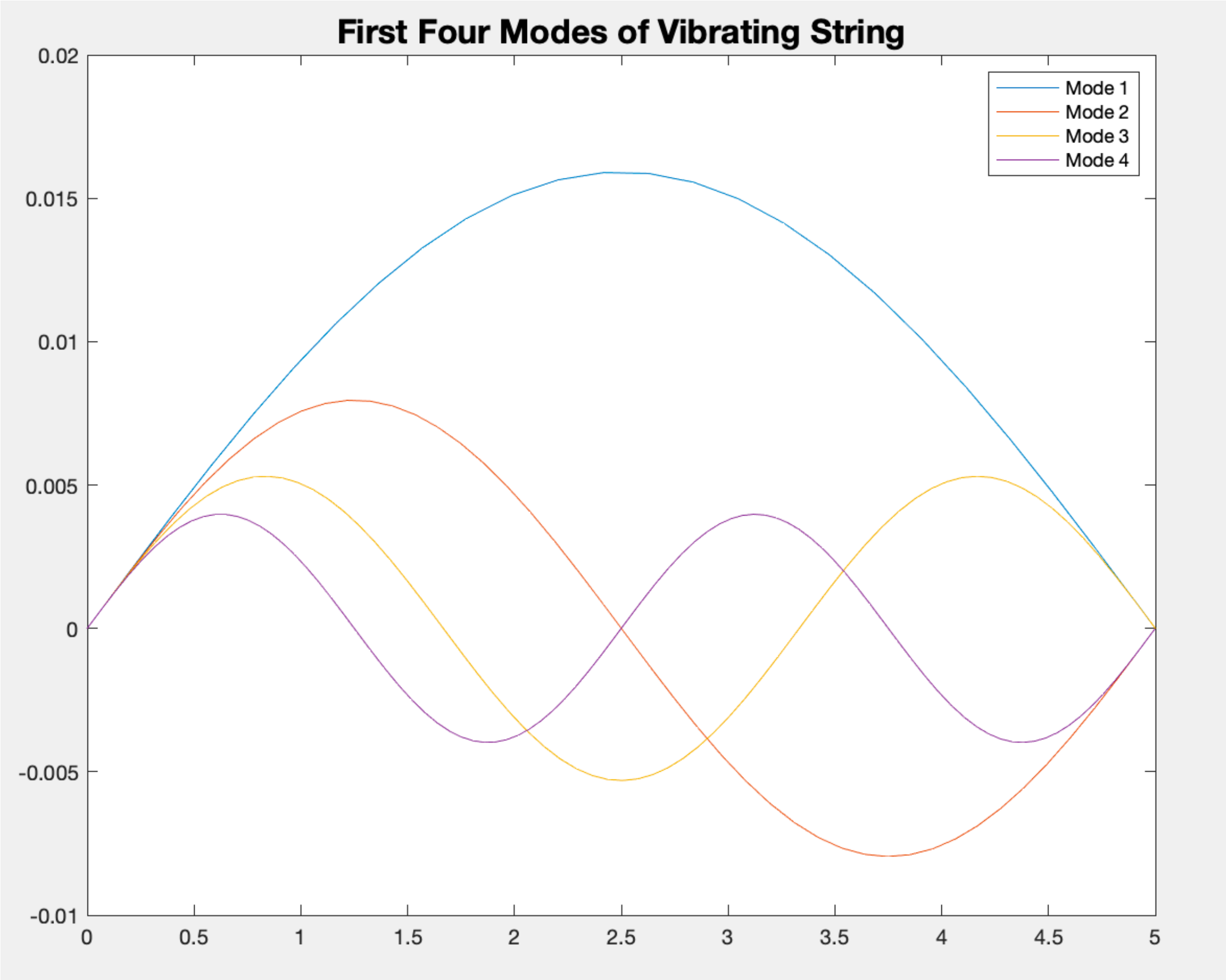
$$\lim_{t \to \infty} v = \lim_{t \to \infty} \left(\frac{2}{e^{2t\sqrt{a}\sqrt{g}} - 1} + 1 \right) \left(-\sqrt{\frac{g}{a}} \right)$$

$$= \lim_{t \to \infty} (0 + 1) \left(-\sqrt{\frac{g}{a}} \right)$$

$$= -\sqrt{\frac{g}{a}}$$

$$= -9.2113$$

Problem 3




```
function out = dydx(t, y)
  % This fcn returns the RHS system of equations for the catenary
  % system.

  global k

  y1 = y(1);  % y value
  y2 = y(2);  % y' value

  dy1dt = y2;
  dy2dt = -k.^2*y1;

  out = [dy1dt; dy2dt];
end



function [x, y] = bvp(y10, y11)
  % This fcn performs shooting method to solve the catenary BVP over the
  % interval x = [0, 1].  The inputs are the initial y value y1(0) and
```

```
  % final y value y1(1).  The BVP equation is defined in the file
  % dydx.m.  The returns are [x, y] where x is a vector of the x values
  % and y is a vector of the chain heights.

  global k;

  % Parameters for shooting method
  tol = 1e-3;
  Ntrials = 50;

  % Parameters for ode solver
  opts = odeset('RelTol',1e-8);  % Solution tolerance.

  % Use binary search to find appropriate value for deriv y2(0).  These
  % are min and max starting guesses.
  y20max = 20;
  y20min = 0;

  % Iterate for Ntrials.  Don't use a "while" loop because it can iterate
forever,
  % causing apparent system hang due to infinite loop.
  for idx = 1:Ntrials
    % Calculate initial guess for slope as midpoint of min and max.
    y20 = (y20max + y20min)/2;

    % Compute solution to IVP using ode45
    s = ode45 (@dydx, [0 5], [y10, y20], opts);

    % Get computed value at right hand end of chain.
    y11trial = s.y(1, end);  % computed end value y(1)
    fprintf('y20max = %f, y20min = %f, y20 = %f, y11trial = %f\n', y20max,
y20min, y20, y11trial)

    % Check if we're done
    if (abs(y11trial - y11) < tol)
      % Close enough.  We're done.  Return computed function.
      x = s.x;
      y = s.y(1, :);
      return
    end

    % Must try again.  Narrow search limits and iterate.
    if (y11trial > y11)
      % Must decrease initial guess
      y20max = y20;
    else
      % Must increase initial guess
      y20min = y20;
    end

  end
  % If we get here it's because we did not converge.
  error('Did not converge!')

end
```

```matlab
function run_bvp()
global k
k = pi/5;
[x, y] = bvp(0,0);
y = (1e-3)*y;
plot(x, y)
hold on
k = 2*pi/5;
[x, y] = bvp(0,0);
y = (1e-3)*y;
plot(x, y)
k = 3*pi/5;
[x, y] = bvp(0,0);
y = (1e-3)*y;
plot(x, y)
k = 4*pi/5;
[x, y] = bvp(0,0);
y = (1e-3)*y;
plot(x, y)
hold off
legend('Mode 1','Mode 2','Mode 3','Mode 4')
title('First Four Modes of Vibrating String','FontSize',16)
```

```matlab
function test_byp()
global k
tol = 1e-4;

for n = 1:4
    k = n*pi/5;
    [x, y] = bvp(0,0);
    y = (1e-3)*y;
    A = max(y);
    f = A*sin(k.*x);
    for i = 1:length(x)
        if (abs(y(i)-f(i)) >= tol)
            disp(i)
            disp(abs(y(i)-f(i)))
            error('The difference is too large!')
        end
    end
end
fprintf('The difference is small.')
```

```
>> test_byp
y20max = 20.000000, y20min = 0.000000, y20 = 10.000000, y11trial = -0.000000
y20max = 20.000000, y20min = 0.000000, y20 = 10.000000, y11trial = 0.000000
y20max = 20.000000, y20min = 0.000000, y20 = 10.000000, y11trial = -0.000000
y20max = 20.000000, y20min = 0.000000, y20 = 10.000000, y11trial = 0.000000
The difference is small.
```