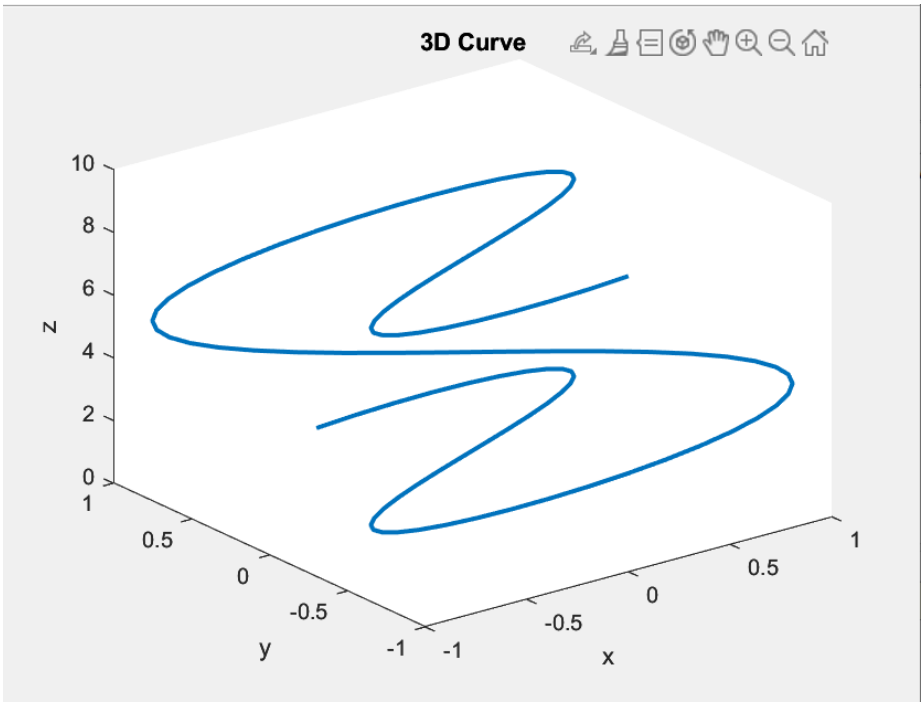
HW 1

1)

```
t = linspace(0, 3*pi, 100);

% Define the x, y, and z coordinates of the curve
x = sin(2*t);
y = cos(t);
z = t;

% Plot the 3D curve
figure
plot3(x, y, z, 'LineWidth', 2);
xlabel('x');
ylabel('y');
zlabel('z');
title('3D Curve');

% Compute the arc length of the curve
f = @(t)(((2*cos(2*t)).^2 + (-sin(t)).^2 + 1.^2).^(1/2));
integral(f,0,3*pi)
```



The arclength of the curve is 17.222032186553953

2)

a)

```
% Single precision

a = single(1e38);
b = single(1e38);
c = a*b;
disp(c)

% Double precision
a = 1e308;
b = 1e308;
c = a*b;
disp(c)
```

Both codes generate an overflow (Inf) as a result.

b)

```
% Divide a normal number by infinity

a = 5;
b = inf;
c = a/b;
disp(c)

% Divide a normal number by zero
a = 5;
b = 0;
c = a/b;
disp(c)

% Divide infinity by -infinity
a = inf;
b = -inf;
c = a/b;
disp(c)

% Divide 0 by 0
a = 0;
b = 0;
c = a/b;
disp(c)
```

The results generated as 0, Inf, Nan and Nan.

c)

```
% Compare infinities
a = inf;
b = -inf;
```

```
c = a < b;
disp(c);

% Compare NaN
a = NaN;
b = 0;
c = a == b;
disp(c);

% Compare normal number
a = 5;
b = 3;
c = a > b;
disp(c);
```

The results generated are 0,0 and 1

d)

```
% Generate a NaN
d = 0;
e = 0;
f = d/e;
disp(f);

% Add NaN to a normal number
g = 4;
add = g + f;
disp(add);

% Multiply NaN to a normal number
g = 4;
mult = g * f;
disp(mult);

% Divide NaN to a normal number
g = 4;
divi = g/f;
disp(divi);
```

All NaN's are the repsonses

e)

```
% Produce positive zero
p_zero = 1/inf;

% Produce negative zero
n_zero = -1/inf;

% Take the reciprocal
p_reciprocal = 1/p_zero;
```

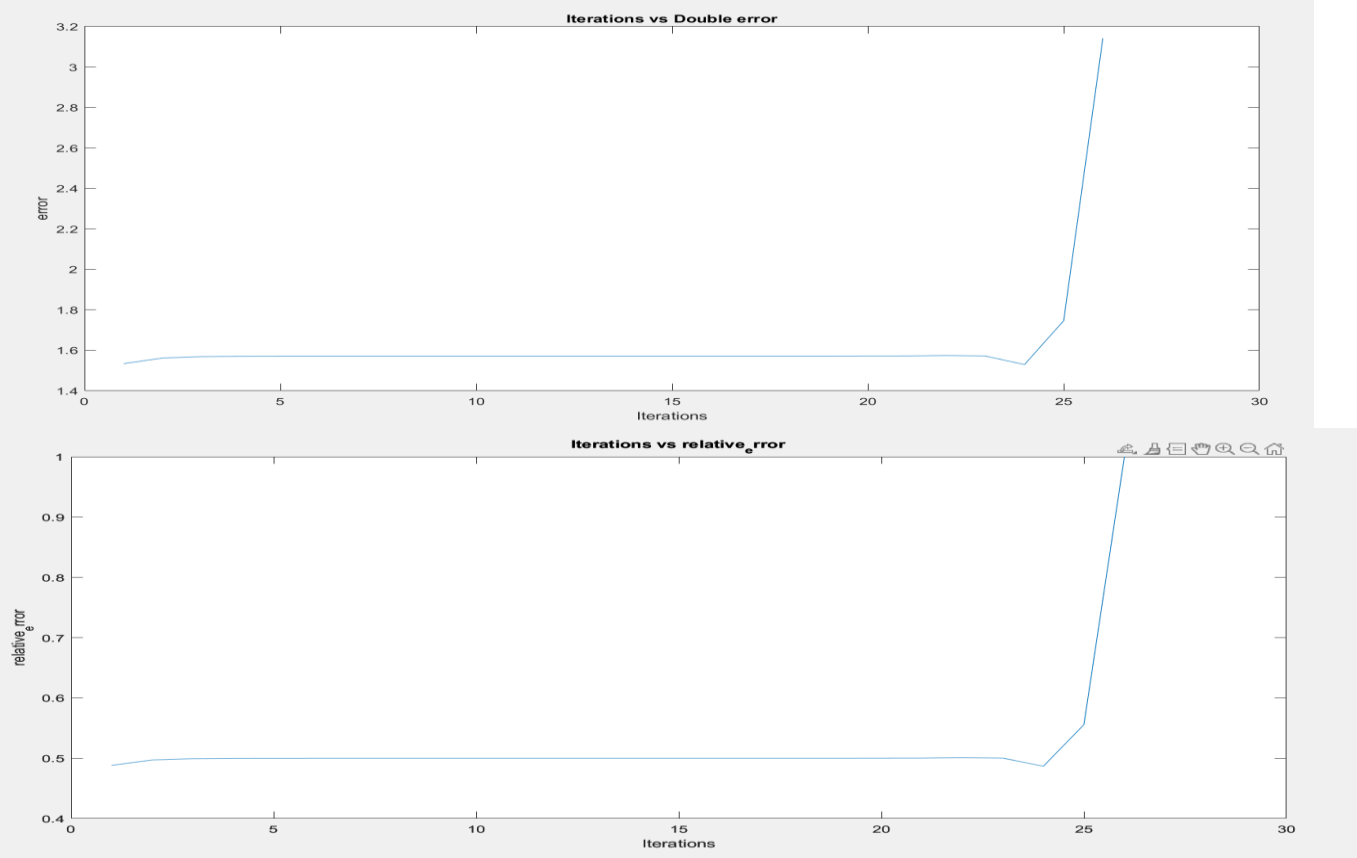
```
n_reciprocal = 1/n_zero;

% Display the results
disp(p_reciprocal);
disp(n_reciprocal);
```

The results are inf and -inf. I don't think I can detect any differences between positive zero and negative zer0

3)

First I did the double precision and graphed it

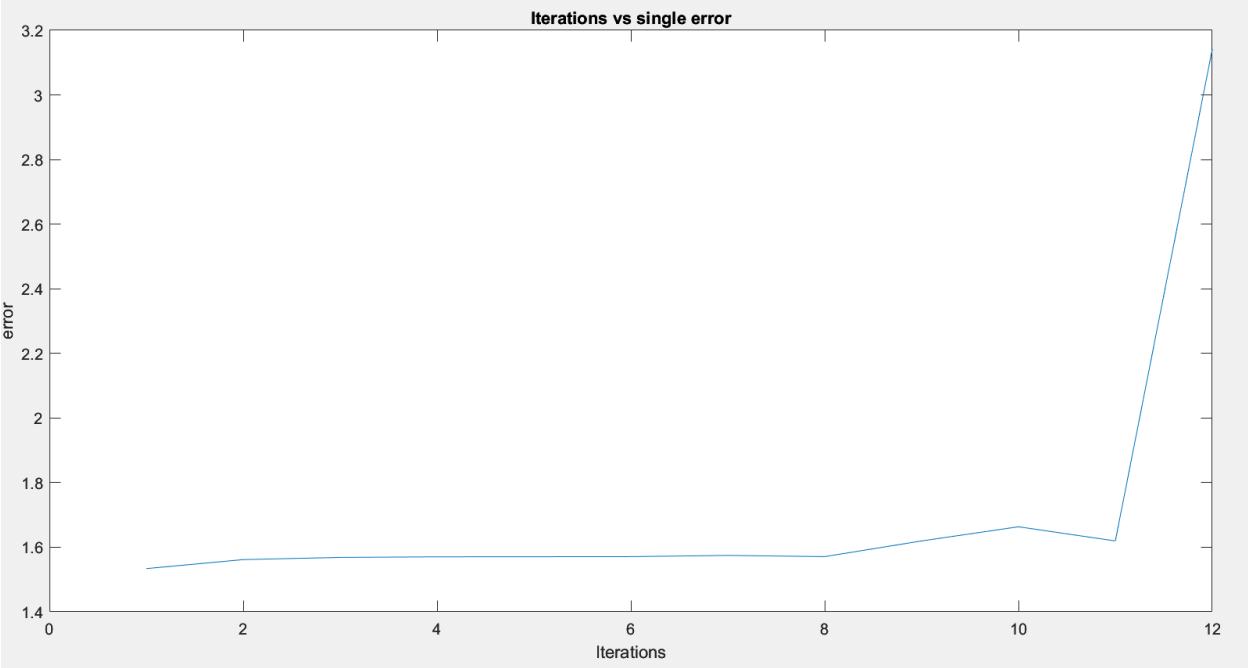
```
% Values
t1 = 1/sqrt(3);
i = 1;
precision = 'double'; % or 'single'

% Data storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = ((1+t1.^2).^(1/2)-1)/t1;
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs Double error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
plot(iterations, relative_error);
xlabel('Iterations');
ylabel('relative_error');
title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)
```

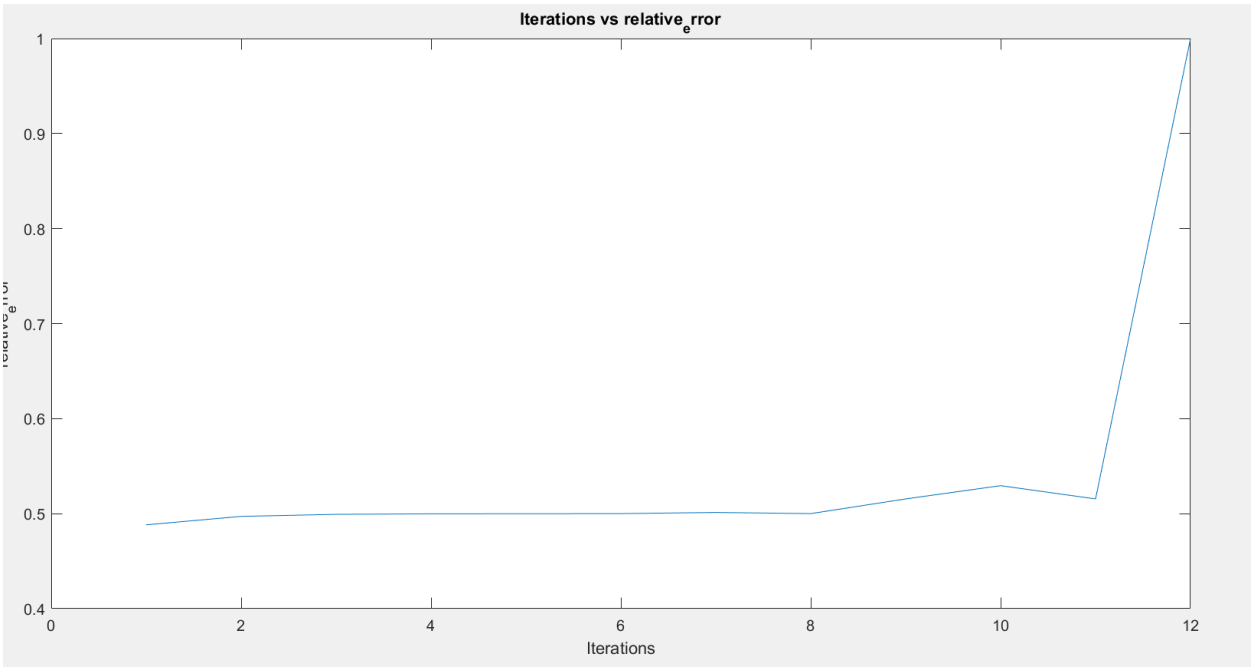


Then I did single precision and graphed it

```
% Values
t1 = single(1/sqrt(3));
i = single(1);
precision = 'single'; % or 'double'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = single(((1+t1.^2).^(1/2)-1)/t1);
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
```

```
ylabel('error');
title('Iterations vs single error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
plot(iterations, relative_error);
xlabel('Iterations');
ylabel('relative_error');
title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)
```





Iterations	Single Precision	Double Precision	Single Precision Error	Double Precision Error	Relative Single Precision Error	Relative Double Precision Error
1	1.60769500	1.6076951545867	1.53389760	1.533897499	0.48825480	0.4882547383252640
2	1.57982990	1.5798299710488	1.56176270	1.561762683	0.49712450	0.4971245017257290
3	1.57304380	1.5730431075657	1.56854880	1.568549546	0.49928460	0.4992848274685550
4	1.57133980	1.5713572998228	1.57025280	1.570235354	0.49982700	0.4998214367393400
5	1.57103380	1.5709365249899	1.57055880	1.570656129	0.49992440	0.4999553735284950
6	1.57064040	1.5708313735275	1.57095220	1.57076128	0.50004960	0.4999888442785230
7	1.56683680	1.5708050882998	1.57475580	1.570787565	0.50126030	0.4999972111263840
8	1.57064040	1.5707985171617	1.57095220	1.570794136	0.50004960	0.4999993027846020
9	1.52207020	1.5707968744084	1.61952250	1.570795779	0.51551000	0.4999998256891990
10	1.47824990	1.5707964639368	1.66334270	1.57079619	0.52945840	0.4999999563463710
11	1.52207020	1.5707963628113	1.61952250	1.570796291	0.51551000	0.4999999885356240
12	0	1.5707963358708	3.14159270	1.570796318	1.00000000	0.4999999971110590
13	NaN	1.5707963094504	NaN	1.570796344	NaN	0.5000000055209110
14	NaN	1.5707963358708	NaN	1.570796318	NaN	0.4999999971110590
15	NaN	1.5707959679410	NaN	1.570796686	NaN	0.5000001142267470
16	NaN	1.5707963358708	NaN	1.570796318	NaN	0.4999999971110590
17	NaN	1.5707905037897	NaN	1.57080215	NaN	0.5000018535201270
18	NaN	1.5707963358708	NaN	1.570796318	NaN	0.4999999971110590
19	NaN	1.5707030773688	NaN	1.570889576	NaN	0.5000296822142040
20	NaN	1.5702717462006	NaN	1.571320907	NaN	0.5001669791892810
21	NaN	1.5700034323455	NaN	1.571589221	NaN	0.5002523861419480
22	NaN	1.5674726878294	NaN	1.574119966	NaN	0.5010579471408150
23	NaN	1.5700034323455	NaN	1.571589221	NaN	0.5002523861419480
24	NaN	1.6122576217674	NaN	1.529335032	NaN	0.4868024599162670
25	NaN	1.3955586065293	NaN	1.746034047	NaN	0.5557798987928430
26	NaN	0	NaN	3.141592654	NaN	1.0000000000000000
27	NaN	NaN	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN	NaN	NaN
29	NaN	NaN	NaN	NaN	NaN	NaN
30	NaN	NaN	NaN	NaN	NaN	NaN

As seen in the chart above:

The single precision error and double precision error increases as iterations increase.

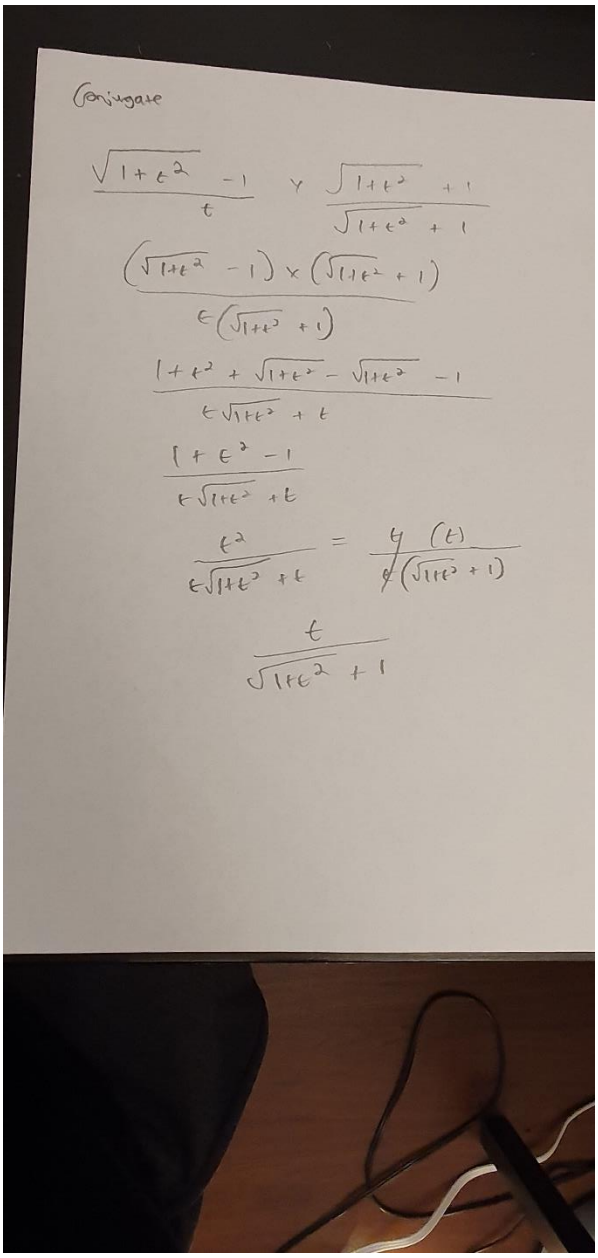
The Relative Single Precision Error is the highest at iteration 12 when single precision error hits the value of PI(3.14)

The Relative Double Precision Error is the highest at iteration 26 when double precision error hits the value of PI(3.14)

The relative error increases as the number of iterations increase. Starting off the relative error is small but as the number of iterations increases the relative error increases.

For small i, the errors are dominated by the truncation error. While for large i, the errors are dominated by the round-off error.

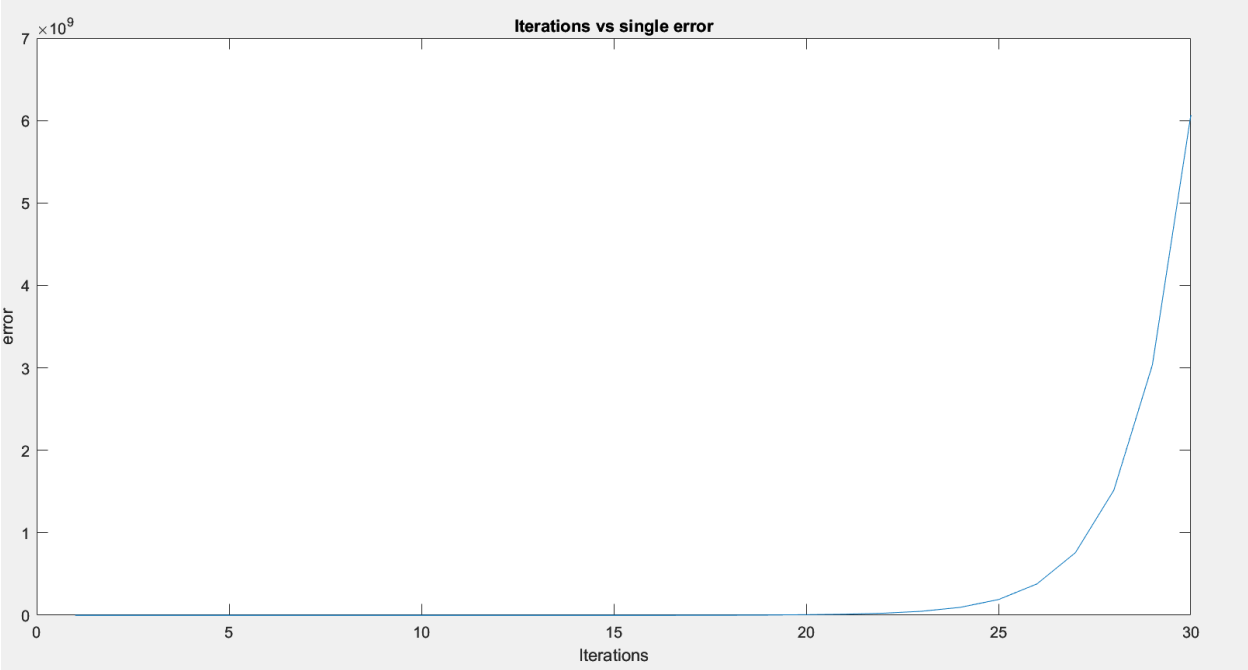
The fix(the conjugate):



First the single and then graph it

```
% values
t1 = single(1/sqrt(3));
i = single(1);
precision = 'single'; % or 'double'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
```

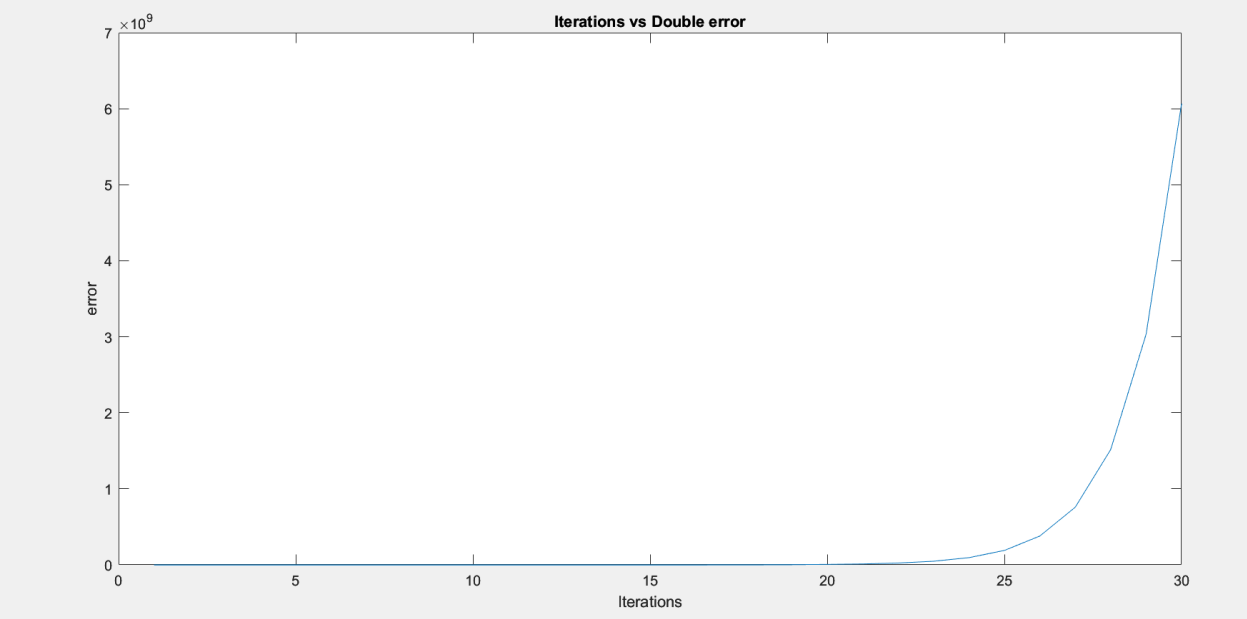
```
for i = 1:30
ti = single((t1)/((1+t1.^2).^(1/2))+1);
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs single error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
%plot(iterations, relative_error);
%xlabel('Iterations');
%ylabel('relative_error');
%title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)
```



Then I did double and then graphed it

```
% values
t1 = 1/sqrt(3);
i = 1;
```

```
precision = 'double'; % or 'single'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = (t1)/((1+t1.^2).^(1/2))+1;
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs Double error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
%plot(iterations, relative_error);
%xlabel('Iterations');
%ylabel('relative_error');
%title('Iterations vs relative_error')

disp(approximations);
disp(errors);
disp(relative_error)
```



The table is as follows:

Iterations	Single Precision	Double Precision	Single Precision Error	Double Precision Error	Relative Single Precision Error	Relative Double Precision Error
	1.0e+09 *	1.0e+09 *	1.0e+09 *	1.0e+09 *	1.0e+09 *	1.0e+09 *
1	0.0000000	0.000000009	0.0000000	5.85841E-09	0.0000000	1.86479E-09
2	0.0000000	2.19846E-08	0.0000000	1.8843E-08	0.0000000	5.99792E-09
3	0.0000000	4.50661E-08	0.0000000	4.19245E-08	0.0000000	1.3345E-08
4	0.0000001	9.03667E-08	0.0000001	8.72251E-08	0.0000000	2.77646E-08
5	0.0000002	1.80782E-07	0.0000002	1.7764E-07	0.0000001	5.65447E-08
6	0.0000004	3.61574E-07	0.0000004	3.58432E-07	0.0000001	1.14093E-07
7	0.0000007	7.2315E-07	0.0000007	7.20008E-07	0.0000002	2.29186E-07
8	0.0000014	1.4463E-06	0.0000014	1.44316E-06	0.0000005	4.59372E-07
9	0.0000029	2.8926E-06	0.0000029	2.88946E-06	0.0000009	9.19743E-07
10	0.0000058	5.7852E-06	0.0000058	5.78206E-06	0.0000018	1.84049E-06
11	0.0000116	1.15704E-05	0.0000116	1.15673E-05	0.0000037	3.68197E-06
12	0.0000231	2.31408E-05	0.0000231	2.31377E-05	0.0000074	7.36495E-06
13	0.0000463	4.62816E-05	0.0000463	4.62785E-05	0.0000147	1.47309E-05
14	0.0000926	9.25632E-05	0.0000926	9.25601E-05	0.0000295	2.94628E-05
15	0.0001851	0.000185126	0.0001851	0.000185123	0.0000589	5.89266E-05
16	0.0003703	0.000370253	0.0003702	0.00037025	0.0001179	0.000117854
17	0.0007405	0.000740506	0.0007405	0.000740503	0.0002357	0.000235709
18	0.0014810	0.001481011	0.0014810	0.001481008	0.0004714	0.00047142
19	0.0029620	0.002962023	0.0029620	0.00296202	0.0009428	0.00094284
20	0.0059240	0.005924046	0.0059240	0.005924043	0.0018857	0.001885681
21	0.0118481	0.011848092	0.0118481	0.011848089	0.0037714	0.003771364
22	0.0236962	0.023696184	0.0236962	0.023696181	0.0075427	0.007542729
23	0.0473924	0.047392368	0.0473924	0.047392365	0.0150855	0.015085458
24	0.0947847	0.094784736	0.0947847	0.094784733	0.0301709	0.030170918
25	0.1895695	0.189569472	0.1895695	0.189569469	0.0603418	0.060341836
26	0.3791389	0.379138944	0.3791389	0.379138941	0.1206837	0.120683673
27	0.7582779	0.758277888	0.7582779	0.758277885	0.2413673	0.241367347
28	1.5165558	1.516555776	1.5165558	1.516555772	0.4827347	0.482734695
29	3.0331116	3.033111551	3.0331116	3.033111548	0.9654694	0.965469392
30	6.0662231	6.066223102	6.0662231	6.066223099	1.9309387	1.930938784

Looking at the two graphs above and the chart, the error starts to climb at iteration 25 and then rapidly increases as the iterations continue. From my Matlab code results, the double precision has 15 decimal places while the single precision has 7 decimal places.

4)

For double

```
x0 = double(pi/4);
approx = zeros(1,10);
h = zeros(1,10);
errors = zeros(1,10);

for m = 1:10
h = double(2^(-m));
approx = double((sin(x0 + h) - 2*sin(x0) + sin(x0 - h)) / h^2);
error = abs(double(-sin(x0) - approx));
fprintf('h = 2^-%d, Approximation: %.16f, Error: %.16f\n', m, approx, error)
end
```

h = 2^-1, Approximation: -0.6924976049824152, Error: 0.0146091762041323

h = 2^-2, Approximation: -0.7034315974102636, Error: 0.0036751837762838

h = 2^-3, Approximation: -0.7061865486355714, Error: 0.0009202325509761

h = 2^-4, Approximation: -0.7068766331668428, Error: 0.0002301480197047

h = 2^-5, Approximation: -0.7070492385623766, Error: 0.0000575426241709

h = 2^-6, Approximation: -0.7070923951791883, Error: 0.0000143860073591

h = 2^-7, Approximation: -0.7071031846626283, Error: 0.0000035965239191

h = 2^-8, Approximation: -0.7071058820511098, Error: 0.0000008991354377

h = 2^-9, Approximation: -0.7071065563941374, Error: 0.0000002247924100

h = 2^-10, Approximation: -0.7071067248471081, Error: 0.0000000563394393

It seems that as h increases the accuracy goes up as well

For single:

```
x0 = single(pi/4);
approx = zeros(1,10,'single');
h = zeros(1,10,"single");
errors = zeros(1,10,"single");

for m = 1:10
h = single(2^(-m));
approx = single((sin(x0 + h) - 2*sin(x0) + sin(x0 - h)) / h^2);
error = abs(single(-sin(x0) - approx));
fprintf('h = 2^-%d, Approximation: %.8f, Error: %.8f\n', m, approx, error)
end
```

h = 2^-1, Approximation: -0.69249725, Error: 0.01460952

h = 2^-2, Approximation: -0.70343018, Error: 0.00367659

h = 2^-3, Approximation: -0.70618057, Error: 0.00092620

h = 2^-4, Approximation: -0.70686340, Error: 0.00024337

h = 2^-5, Approximation: -0.70703125, Error: 0.00007552

h = 2^-6, Approximation: -0.70678711, Error: 0.00031966

h = 2^-7, Approximation: -0.70703125, Error: 0.00007552

h = 2^-8, Approximation: -0.70312500, Error: 0.00398177

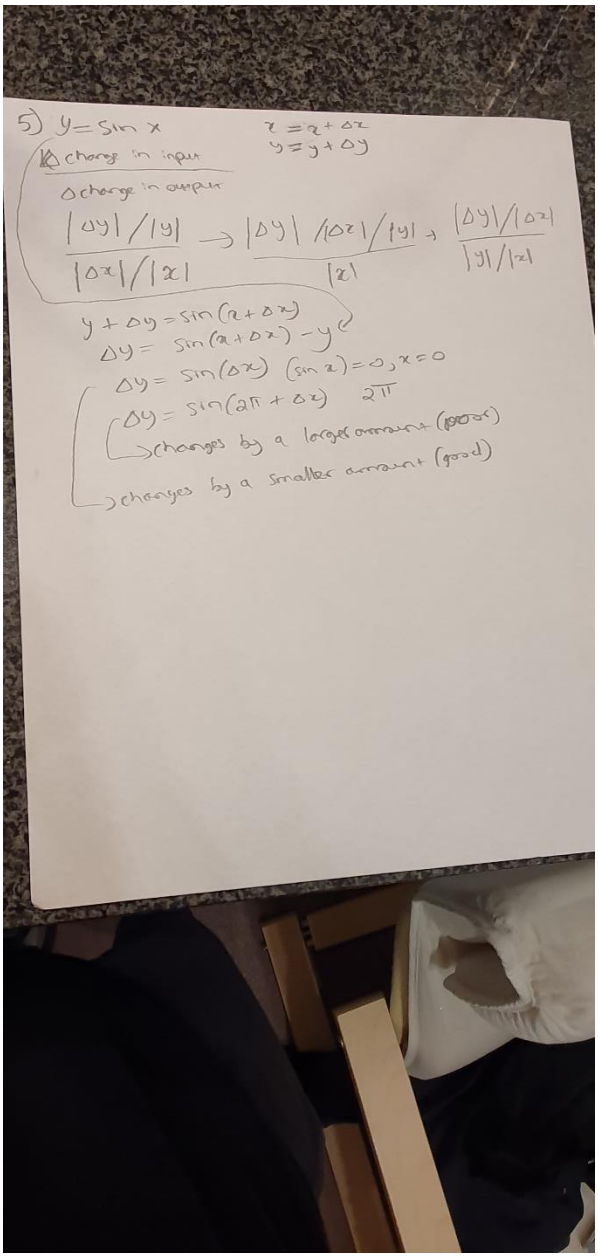
h = 2^-9, Approximation: -0.68750000, Error: 0.01960677

h = 2^-10, Approximation: -0.68750000, Error: 0.01960677

It seems that 4, 5 and 6 seems to be the closest to the answer.

Iterations	Single Precision	Double Precision	Single Precision Error	Double Precision Error
1	**-0.6924973**	**-0.692497605**	**0.01460952**	**0.014609176**
2	**-0.7034302**	**-0.703431597**	**0.00367659**	**0.003675184**
3	**-0.7061806**	**-0.706186549**	**0.0009262**	**0.000920233**
4	**-0.7068634**	**-0.706876633**	**0.00024337**	**0.000230148**
5	**-0.7070313**	**-0.707049239**	**0.00007552**	**5.75426E-05**
6	**-0.7067871**	**-0.707092395**	**0.00031966**	**1.4386E-05**
7	**-0.7070313**	**-0.707103185**	**0.00007552**	**3.59652E-06**
8	**-0.703125**	**-0.707105882**	**0.00398177**	**8.99135E-07**
9	**-0.6875**	**-0.707106556**	**0.01960677**	**2.24792E-07**
10	**-0.6875**	**-0.707106725**	**0.01960677**	**5.63394E-08**

5)


5) y = sin x
x = x + Δx
y = y + Δy
Δ change in input
Δ change in output
|Δy| / |y| → |Δy| / |Δx| / |y| → |Δy| / |Δx| / |y| / |x|
|Δx| / |x| |x|
y + Δy = sin(x + Δx)
Δy = sin(x + Δx) − y
Δy = sin(Δx) (sin x) = 0, x = 0
Δy = sin(2π + Δx) 2π
changes by a larger amount (good)
changes by a smaller amount (good)

Code:

1)

```
t = linspace(0, 3*pi, 100);

% Define the x, y, and z coordinates of the curve
x = sin(2*t);
y = cos(t);
z = t;

% Plot the 3D curve
figure
plot3(x, y, z, 'LineWidth', 2);
xlabel('x');
ylabel('y');
zlabel('z');
title('3D Curve');

% Compute the arc length of the curve
f = @(t)(((2*cos(2*t)).^2 + (-sin(t)).^2 + 1.^2).^(1/2));
integral(f,0,3*pi)
```

2)

```
% Single precision

a = single(1e38);
b = single(1e38);
c = a*b;
disp(c)

% Double precision
a = 1e308;
b = 1e308;
c = a*b;
disp(c)


% Divide a normal number by infinity

a = 5;
b = inf;
c = a/b;
disp(c)

% Divide a normal number by zero
a = 5;
b = 0;
c = a/b;
disp(c)

% Divide infinity by -infinity
a = inf;
b = -inf;
```

```
c = a/b;
disp(c)

% Divide 0 by 0
a = 0;
b = 0;
c = a/b;
disp(c)


% Compare infinities
a = inf;
b = -inf;
c = a < b;
disp(c);

% Compare NaN
a = NaN;
b = 0;
c = a == b;
disp(c);

% Compare normal number
a = 5;
b = 3;
c = a > b;
disp(c);


% Generate a NaN
d = 0;
e = 0;
f = d/e;
disp(f);

% Add NaN to a normal number
g = 4;
add = g + f;
disp(add);

% Multiply NaN to a normal number
g = 4;
mult = g * f;
disp(mult);

% Divide NaN to a normal number
g = 4;
divi = g/f;
disp(divi);


% Produce positive zero
p_zero = 1/inf;

% Produce negative zero
```

```
n_zero = -1/inf;

% Take the reciprocal
p_reciprocal = 1/p_zero;
n_reciprocal = 1/n_zero;

% Display the results
disp(p_reciprocal);
disp(n_reciprocal);
```

3)

```
% Values
t1 = 1/sqrt(3);
i = 1;
precision = 'double'; % or 'single'

% Data storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = ((1+t1.^2).^(1/2)-1)/t1;
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs Double error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
plot(iterations, relative_error);
xlabel('Iterations');
ylabel('relative_error');
title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)
```

```
% Values
t1 = single(1/sqrt(3));
i = single(1);
precision = 'single'; % or 'double'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = single(((1+t1.^2).^(1/2)-1)/t1);
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs single error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
plot(iterations, relative_error);
xlabel('Iterations');
ylabel('relative_error');
title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)


% values
t1 = single(1/sqrt(3));
i = single(1);
precision = 'single'; % or 'double'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = single((t1)/((1+t1.^2).^(1/2))+1);
approximations(i) = 6 * (2^(i-1)) * ti;
```

```
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs single error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
%plot(iterations, relative_error);
%xlabel('Iterations');
%ylabel('relative_error');
%title('Iterations vs relative_error')

disp(approximations)
disp(errors)
disp(relative_error)


% values
t1 = 1/sqrt(3);
i = 1;
precision = 'double'; % or 'single'

% Data Storage
iterations = zeros(1,30,precision);
approximations = zeros(1,30,precision);
errors = zeros(1,30,precision);
relative_error = zeros(1,30,precision);

% Iterate and calculate
for i = 1:30
ti = (t1)/((1+t1.^2).^(1/2))+1;
approximations(i) = 6 * (2^(i-1)) * ti;
errors(i) = abs((approximations(i) - pi));
relative_error(i) = abs(errors(i))/pi;
iterations(i) = i;
t1 = ti;
end

% Plot error vs iterations
plot(iterations, errors);
xlabel('Iterations');
ylabel('error');
title('Iterations vs Double error')

% Plot relative_error vs iterations
% Matlab seems to only do one graph at a time so I had to do this one by
% one by making the code for the other plot into comments by adding %
```

```
%plot(iterations, relative_error);
%xlabel('Iterations');
%ylabel('relative_error');
%title('Iterations vs relative_error')

disp(approximations);
disp(errors);
disp(relative_error)
```

4)

```
x0 = double(pi/4);
approx = zeros(1,10);
h = zeros(1,10);
errors = zeros(1,10);

for m = 1:10
h = double(2^(-m));
approx = double((sin(x0 + h) - 2*sin(x0) + sin(x0 - h)) / h^2);
error = abs(double(-sin(x0) - approx));
fprintf('h = 2^-%d, Approximation: %.16f, Error: %.16f\n', m, approx, error)
end


x0 = single(pi/4);
approx = zeros(1,10,'single');
h = zeros(1,10,"single");
errors = zeros(1,10,"single");

for m = 1:10
h = single(2^(-m));
approx = single((sin(x0 + h) - 2*sin(x0) + sin(x0 - h)) / h^2);
error = abs(single(-sin(x0) - approx));
fprintf('h = 2^-%d, Approximation: %.8f, Error: %.8f\n', m, approx, error)
end
```