**Practical Assignment - VI**

**Ashutosh Jha (11811)**

**Ques. 1:** Find the polynomial interpolating the points

| **x** | 1 | 1.3 | 1.6 | 1.9 | 2.2 |
| --- | --- | --- | --- | --- | --- |
| **f(x)** | 0.1411 | -0.6878 | -0.9962 | -0.5507 | 0.3115 |

where f(x) = sin(3x). Compare interpolating polynomials with f(x) (using graph).

**Sol. 1:**

**Program (using C):**

#include <bits/stdc++.h>

using namespace std;

float proterm(int i, float value, float x[])

{

float pro = 1;

for (int j = 0; j < i; j++) {

pro = pro \* (value - x[j]);

}

return pro;

}

void dividedDiffTable(float x[], float y[][10], int n)

{

for (int i = 1; i < n; i++) {

for (int j = 0; j < n - i; j++) {

y[j][i] = (y[j][i - 1] - y[j + 1]

[i - 1]) / (x[j] - x[i + j]);

}

}

}

float applyFormula(float value, float x[],

float y[][10], int n)

{

float sum = y[0][0];

for (int i = 1; i < n; i++) {

sum = sum + (proterm(i, value, x) \* y[0][i]);

}

return sum;

}

void printDiffTable(float y[][10],int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n - i; j++) {

cout << setprecision(4) <<

y[i][j] << " ";

}

cout << "\n\n";

}

}

int main()

{

int n = 5;

float value, sum, y[10][10];

float x[] = {1, 1.3, 1.6, 1.9, 2.2};

y[0][0] = 0.1411 ;

y[1][0] = -0.6878;

y[2][0] = -0.9962;

y[3][0] = -0.5507;

y[4][0] = 0.3115;

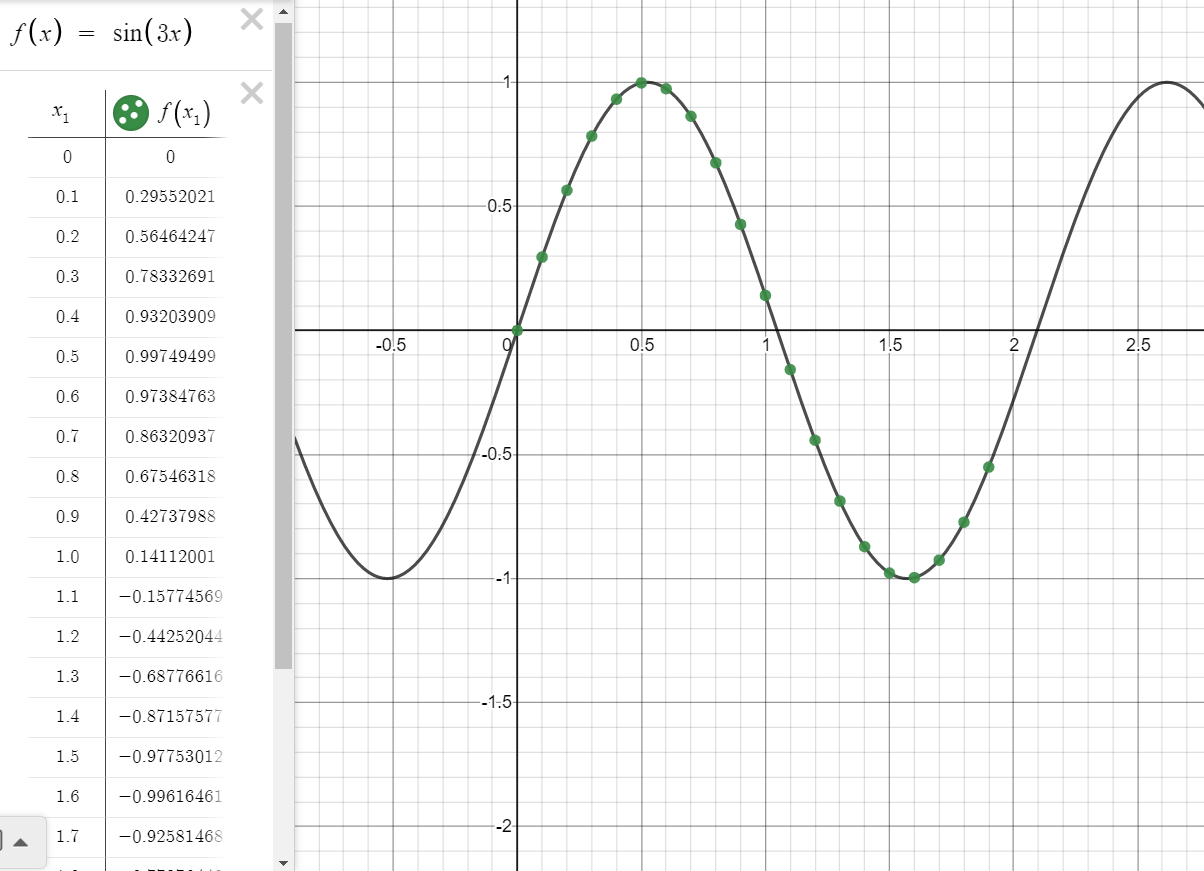
dividedDiffTable(x, y, n);

printDiffTable(y,n);

return 0;

}

**Graph for f(x) = sin(3x):**



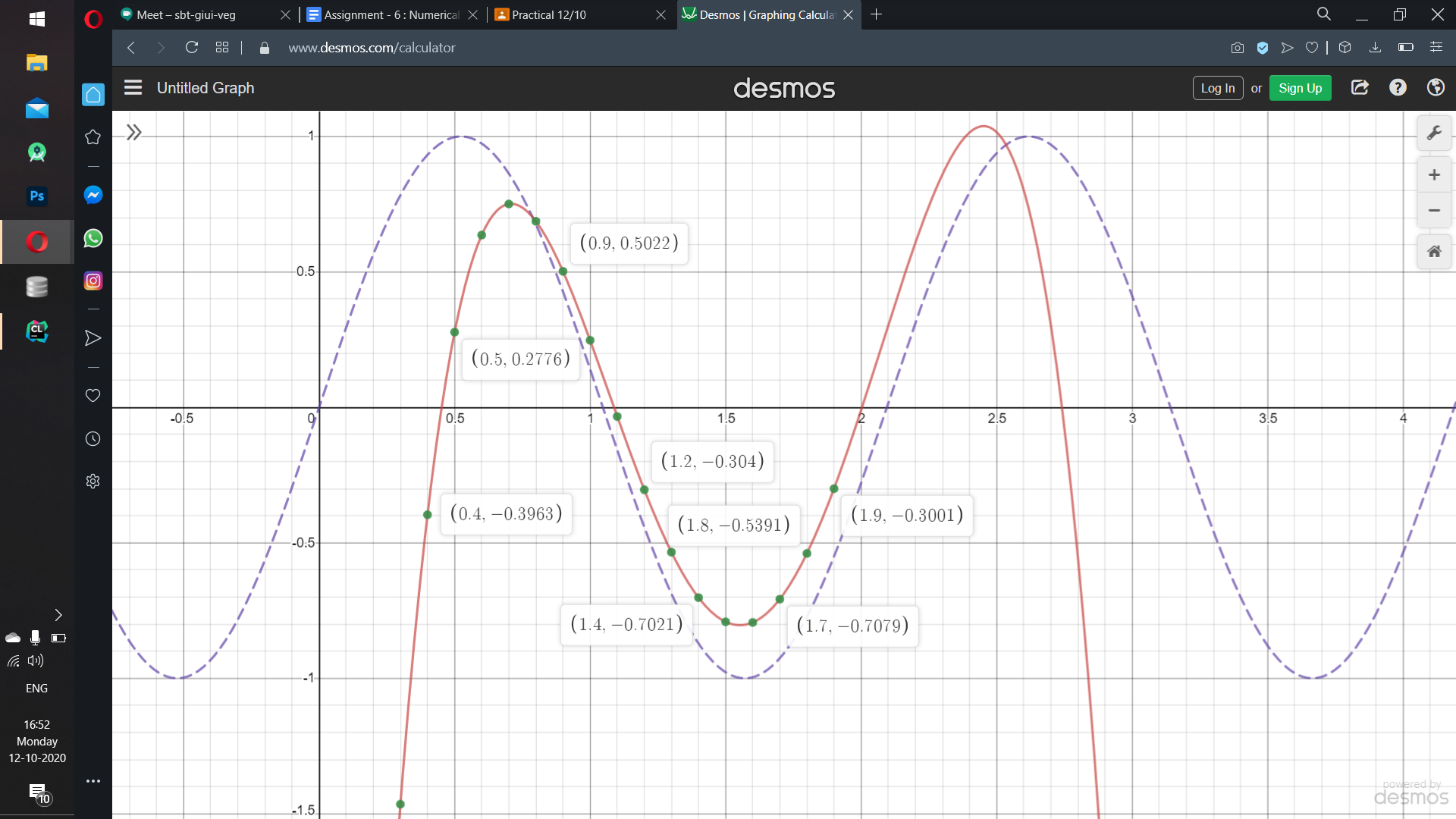
We obtain our interpolating polynomial as g(x) = -2.93512 x4 + 18.447 x3 - 39.0141 x2 + 31.6821 x - 7.9324

**Observation Table:**

| **x** | **sin(3x)** | **g(x)** |
| --- | --- | --- |
| 0 | 0 | -7.9324 |
| 0.1 | 0.29552021 | -5.1361775 |
| 0.2 | 0.56464247 | -3.0136642 |
| 0.3 | 0.78332691 | -1.4647445 |
| 0.4 | 0.93203909 | -0.39634707 |
| 0.5 | 0.99749499 | 0.277555 |
| 0.6 | 0.97384763 | 0.63594445 |
| 0.7 | 0.86320937 | 0.75075969 |
| 0.8 | 0.67546318 | 0.68689485 |
| 0.9 | 0.42737988 | 0.50219977 |
| 1 | 0.14112001 | 0.2478 |
| 1.1 | -0.15774569 | -0.033503192 |
| 1.2 | -0.44252044 | -0.30403283 |
| 1.3 | -0.68776616 | -0.53443623 |
| 1.4 | -0.87157577 | -0.70208499 |
| 1.5 | -0.97753012 | -0.791395 |
| 1.6 | -0.99616461 | -0.79352643 |
| 1.7 | -0.92581468 | -0.70788375 |
| 1.8 | -0.77276449 | -0.53911571 |
| 1.9 | -0.55068554 | -0.30011535 |

**Graph for g(x) = -2.93512 x4 + 18.447 x3 - 39.0141 x2 + 31.6821 x - 7.9324 along with f(x) = sin(3x)**

Here violet line represents sin(3x) and red line represents g(x) =-2.93512 x4 + 18.447 x3 - 39.0141 x2 + 31.6821 x - 7.9324



**Conclusion:** Except for the range [0.9, 1.8], the interpolated polynomial formed shows a divergence effect i.e. error from function sin(3x) keeps on increasing

**Ques. 2:** For functions f(x) = cos(x) and f(x) = , find the interpolating polynomial through a set of equidistant points in the interval [-5, 5].

Find the interpolating polynomial for n = 6 and then for n = 11 and compare both these graphs with f(x). (n= number of points).

And write a short note for your error analysis.

**Sol. 2:**

**For f(x) = cos(x) and n = 6**

**Program (using C):**

#include <bits/stdc++.h>

using namespace std;

float proterm(int i, float value, float x[])

{

float pro = 1;

for (int j = 0; j < i; j++) {

pro = pro \* (value - x[j]);

}

return pro;

}

void dividedDiffTable(float x[], float y[][10], int n)

{

for (int i = 1; i < n; i++) {

for (int j = 0; j < n - i; j++) {

y[j][i] = (y[j][i - 1] - y[j + 1]

[i - 1]) / (x[j] - x[i + j]);

}

}

}

float applyFormula(float value, float x[],

float y[][10], int n)

{

float sum = y[0][0];

for (int i = 1; i < n; i++) {

sum = sum + (proterm(i, value, x) \* y[0][i]);

}

return sum;

}

void printDiffTable(float y[][10],int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n - i; j++) {

cout << setprecision(4) <<

y[i][j] << " ";

}

cout << "\n\n";

}

}

int main()

{

int n = 6;

float value, sum, y[10][10];

float x[] = {-5, -4, -3, -2, -1, 0};

y[0][0] = 0.2836 ;

y[1][0] = -0.6536;

y[2][0] = -0.9899;

y[3][0] = -0.4161;

y[4][0] = 0.5403 ;

y[5][0] = 1 ;

dividedDiffTable(x, y, n);

printDiffTable(y,n);

return 0;

}

**For f(x) = cos(x) and n = 11**

**Program (using C):**

#include <bits/stdc++.h>

using namespace std;

float proterm(int i, float value, float x[])

{

float pro = 1;

for (int j = 0; j < i; j++) {

pro = pro \* (value - x[j]);

}

return pro;

}

void dividedDiffTable(float x[], float y[][10], int n)

{

for (int i = 1; i < n; i++) {

for (int j = 0; j < n - i; j++) {

y[j][i] = (y[j][i - 1] - y[j + 1]

[i - 1]) / (x[j] - x[i + j]);

}

}

}

float applyFormula(float value, float x[],

float y[][10], int n)

{

float sum = y[0][0];

for (int i = 1; i < n; i++) {

sum = sum + (proterm(i, value, x) \* y[0][i]);

}

return sum;

}

void printDiffTable(float y[][10],int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n - i; j++) {

cout << setprecision(4) <<

y[i][j] << " ";

}

cout << "\n\n";

}

}

int main()

{

int n = 11;

float value, sum, y[10][10];

float x[] = {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5};

y[0][0] = 0.2836 ;

y[1][0] = -0.6536;

y[2][0] = -0.9899;

y[3][0] = -0.4161;

y[4][0] = 0.5403 ;

y[5][0] = 1 ;

y[6][0] = 0.5403 ;

y[7][0] = -0.4161;

y[8][0] = -0.9899;

y[9][0] = -0.6536;

y[10][0] = 0.2836 ;

dividedDiffTable(x, y, n);

printDiffTable(y,n);

return 0;

}

So, our interpolating polynomials are :

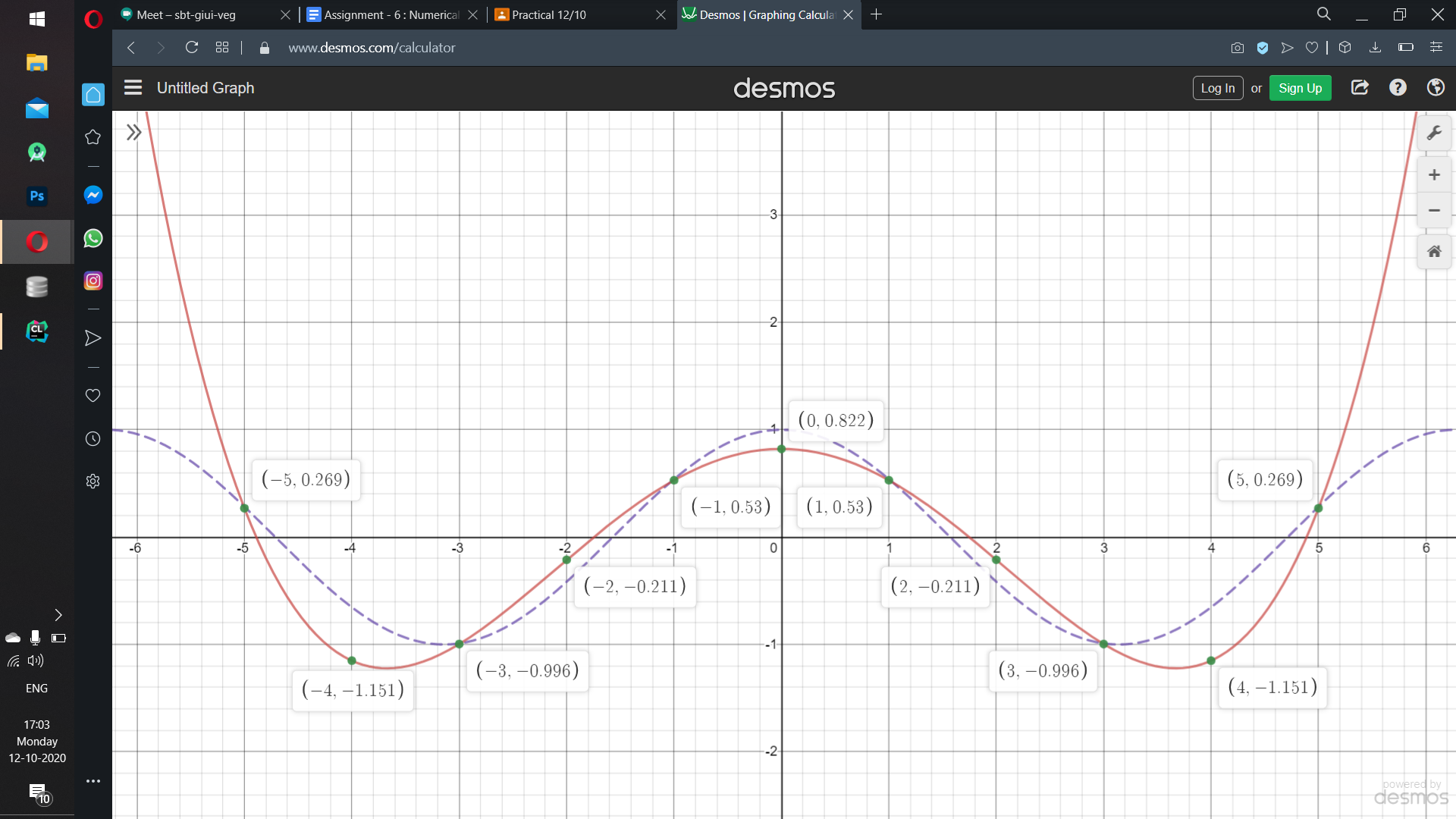
**g(x) {for n=6}:** -8.799 \* 10-8 x5 + 0.01124 x4 + 2.25 \* 10-6 x3 - 0.3031 x2 - 2.192 \*10-6 x + 0.8215

**g(x) {for n=11}:** -1.411 \* 10-11 x10 - 1.2412 \* 10-21 x9 + 7.761 \* 10-10 x8 + 1.250 \* 10-19 x7 - 1.412 \* 10-8 x6 - 2.527 \* 10-18 x5 + 0.0122 x4 + 0 x3 - 0.3042 x2 + 1.675 \*10-16 x + 0.8324

| **x** | **cos(x)** | **g(x) for n = 6** | **g(x) for n = 11** |
| --- | --- | --- | --- |
| -10 | -0.83907 | 82.918071 | 92.33479 |
| -9 | -0.911130 | 50.019615 | 56.213106 |
| -8 | -0.1455 | 27.463889 | 31.328969 |
| -7 | 0.7539 | 12.957562 | 15.217627 |
| -6 | 0.96017 | 4.4771514 | 5.6921916 |
| -5 | 0.28366 | 0.26900468 | 0.85234475 |
| -4 | -0.6536 | -1.1507051 | -0.91162177 |
| -3 | -0.9899 | -0.99599279 | -0.91720603 |
| -2 | -0.4161 | -0.2110708 | -0.18920072 |
| -1 | 0.5403 | 0.52964003 | 0.54039999 |
| 0 | 1 | 0.8215 | 0.8324 |
| 1 | 0.5403 | 0.52963997 | 0.54039999 |
| 2 | -0.4161 | -0.2110492 | -0.18920072 |
| 3 | -0.9899 | -0.99592721 | -0.91720603 |
| 4 | -0.6536 | -1.1506149 | -0.91162177 |
| 5 | 0.25366 | 0.26899532 | 0.85234475 |
| 6 | 0.96017 | 4.4767286 | 5.6921916 |
| 7 | 0.75390 | 12.956118 | 15.217627 |
| 8 | -0.1455 | 27.460391 | 31.328969 |
| 9 | -0.9111 | 50.012465 | 56.213106 |
| 10 | -0.8390 | 82.904929 | 92.33179 |

**Graph for g(x) for n = 6**

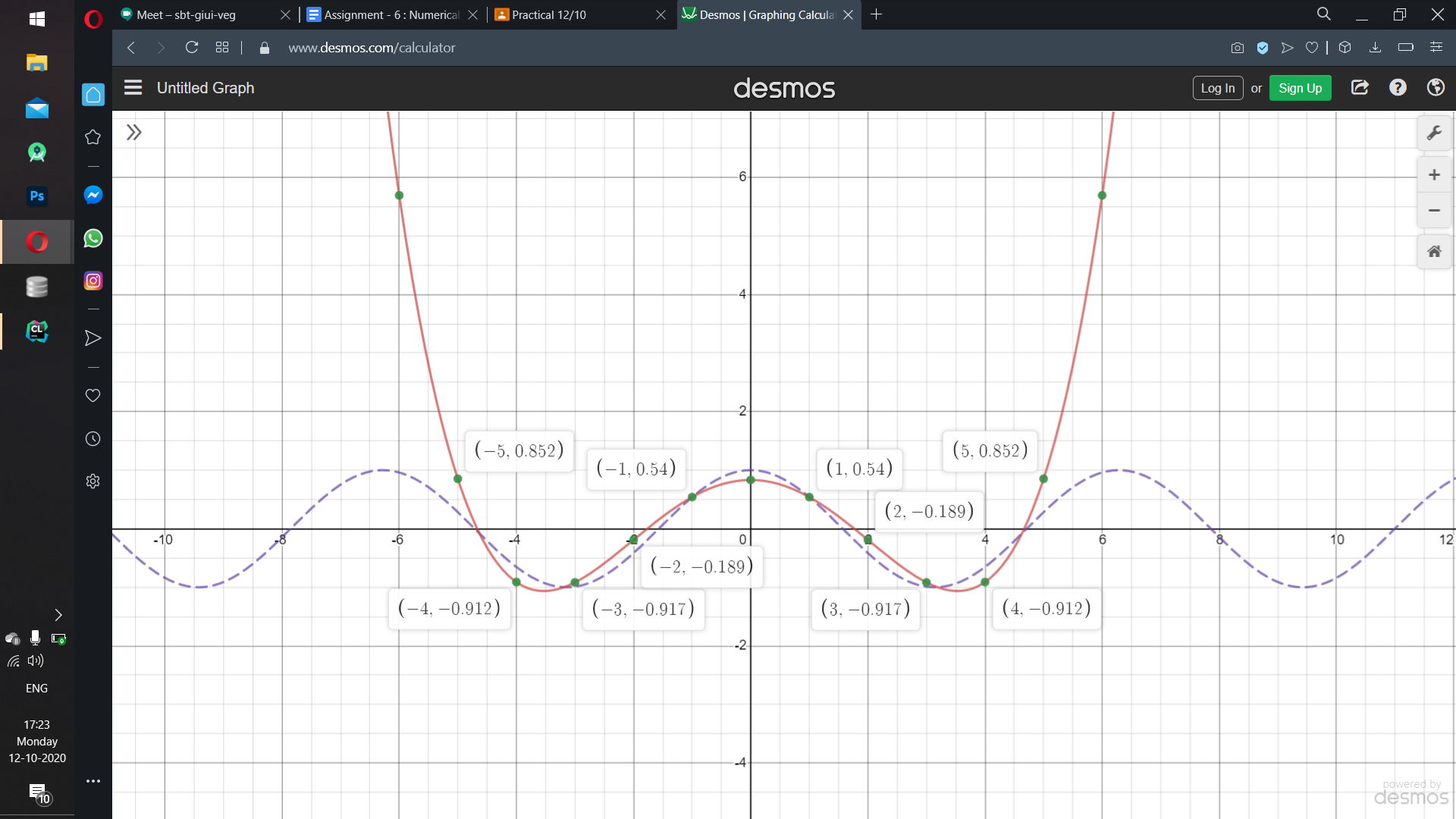
Violet line shows f(x) = cos(x) while red line shows our interpolation polynomial g(x) = -8.799 \* 10-8 x5 + 0.01124 x4 + 2.25 \* 10-6 x3 - 0.3031 x2 - 2.192 \*10-6 x + 0.8215



The graph shows a similar trend as cos x in the range [-5,5] but outside that range, it increases exponentially as we go away from the origin.

**Graph for g(x) for n = 11:**

Violet line shows f(x) = cos(x) while red line shows our interpolation polynomial g(x) = -1.411 \* 10-11 x10 - 1.2412 \* 10-21 x9 + 7.761 \* 10-10 x8 + 1.250 \* 10-19 x7 - 1.412 \* 10-8 x6 - 2.527 \* 10-18 x5 + 0.0122 x4 + 0 x3 - 0.3042 x2 + 1.675 \*10-16 x + 0.8324



The graph shows a similar trend as cos x in the range [-5,5] but outside that range, it increases exponentially as we go away from the origin.

**For f(x) = and n = 6**

**Program (using C):**

#include <bits/stdc++.h>

using namespace std;

float proterm(int i, float value, float x[])

{

float pro = 1;

for (int j = 0; j < i; j++) {

pro = pro \* (value - x[j]);

}

return pro;

}

void dividedDiffTable(float x[], float y[][10], int n)

{

for (int i = 1; i < n; i++) {

for (int j = 0; j < n - i; j++) {

y[j][i] = (y[j][i - 1] - y[j + 1]

[i - 1]) / (x[j] - x[i + j]);

}

}

}

float applyFormula(float value, float x[],

float y[][10], int n)

{

float sum = y[0][0];

for (int i = 1; i < n; i++) {

sum = sum + (proterm(i, value, x) \* y[0][i]);

}

return sum;

}

void printDiffTable(float y[][10],int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n - i; j++) {

cout << setprecision(4) <<

y[i][j] << " ";

}

cout << "\n\n";

}

}

int main()

{

int n = 6;

float value, sum, y[10][10];

float x[] = {-5, -3, -1, 1, 3, 5};

y[0][0] = 0.03846 ;

y[1][0] = 0.1;

y[2][0] = 0.5;

y[3][0] = 0.5;

y[4][0] = 0.1 ;

y[5][0] = 0.03846; ;

dividedDiffTable(x, y, n);

printDiffTable(y,n);

return 0;

}

**For f(x) = and n = 11**

**Program (using C):**

#include <bits/stdc++.h>

using namespace std;

float proterm(int i, float value, float x[])

{

float pro = 1;

for (int j = 0; j < i; j++) {

pro = pro \* (value - x[j]);

}

return pro;

}

void dividedDiffTable(float x[], float y[][10], int n)

{

for (int i = 1; i < n; i++) {

for (int j = 0; j < n - i; j++) {

y[j][i] = (y[j][i - 1] - y[j + 1]

[i - 1]) / (x[j] - x[i + j]);

}

}

}

float applyFormula(float value, float x[],

float y[][10], int n)

{

float sum = y[0][0];

for (int i = 1; i < n; i++) {

sum = sum + (proterm(i, value, x) \* y[0][i]);

}

return sum;

}

void printDiffTable(float y[][10],int n)

{

for (int i = 0; i < n; i++) {

for (int j = 0; j < n - i; j++) {

cout << setprecision(4) <<

y[i][j] << " ";

}

cout << "\n\n";

}

}

int main()

{

int n = 11;

float value, sum, y[10][10];

float x[] = {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5};

y[0][0] = 0.03846 ;

y[1][0] = 0.0588;

y[2][0] = 0.1;

y[3][0] = 0.2;

y[4][0] = 0.5;

y[5][0] = 1 ;

y[6][0] = 0.5 ;

y[7][0] = 0.2;

y[8][0] = 0.1;

y[9][0] = 0.0588;

y[10][0] = 0.03846 ;

dividedDiffTable(x, y, n);

printDiffTable(y,n);

return 0;

}

So, our interpolating polynomials are :

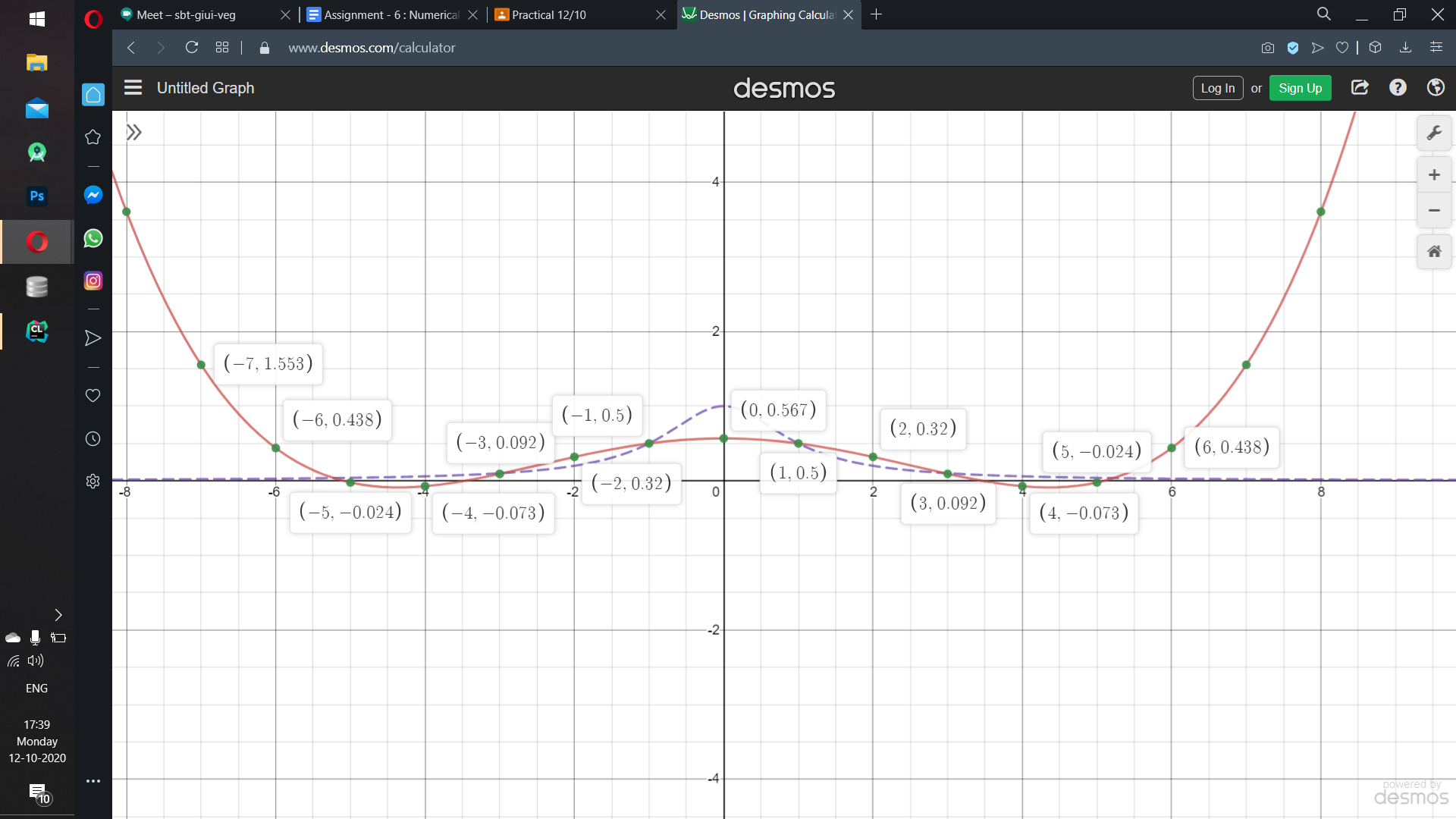
**g(x) {for n=6}:** 0\*x5 + 0.001823x4 + 0\*x3 - 0.069213x2 + 0\*x + 0.567318

**g(x) {for n = 11}:** -0.00002242 x 10 + 0\* x9 + 0.00125 x8 + 0 x7 - 0.024 x6 + 0 \* x5 + 0.1971 x4 + 0 \* x3 - 0.6742 x2 - 5.5541 \* 10-17 x + 1.0076

| **x** | **f(x) =** | **g(x) for n = 6** | **g(x) for n = 11** |
| --- | --- | --- | --- |
| -10 | 0.00990 | 11.876018 | -121295.41 |
| -9 | 0.0121 | 6.921768 | -35880.319 |
| -8 | 0.01538 | 3.604694 | -8628.0473 |
| -7 | 0.02 | 1.552904 | -1509.4609 |
| -6 | 0.02702 | 0.438258 | -143.69767 |
| -5 | 0.03846 | -0.023632 | 1.6760375 |
| -4 | 0.058823 | -0.073402 | 0.78492608 |
| -3 | 0.1 | 0.092064 | 0.2867142 |
| -2 | 0.2 | 0.319634 | 0.22544192 |
| -1 | 0.5 | 0.499928 | 0.50772758 |
| 0 | 1 | 0.567318 | 1.0076 |
| 1 | 0.5 | 0.499928 | 0.50772758 |
| 2 | 0.2 | 0.319634 | 0.22544192 |
| 3 | 0.1 | 0.092064 | 0.28627142 |
| 4 | 0.058823 | -0.073402 | 0.78492608 |
| 5 | 0.03846 | -0.023632 | 1.6760375 |
| 6 | 0.02702 | 0.438258 | -143.69767 |
| 7 | 0.02 | 1.552901 | -1509.4609 |
| 8 | 0.0153846 | 3.604694 | -8628.0473 |
| 9 | 0.01219 | 6.21768 | -35880.319 |
| 10 | 0.00990 | 11.876018 | -121295.41 |

**Graph for g(x) for n = 6:**

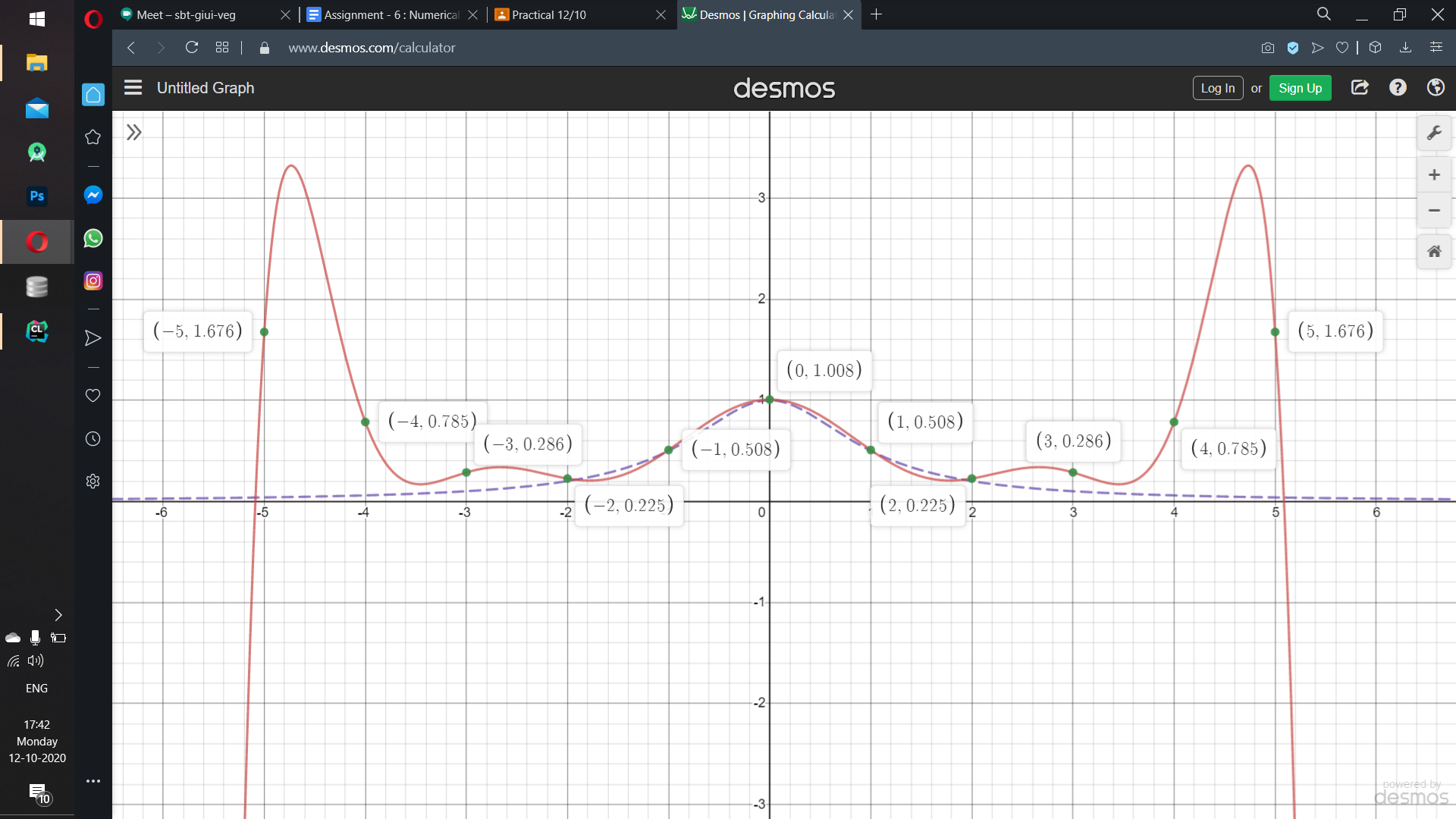
Violet line shows our f(x) **=** and red line shows our interpolating polynomial g(x) = 0\*x5 + 0.001823x4 + 0\*x3 - 0.069213x2 + 0\*x + 0.567318



**Conclusion:** In the above graph, the g(x) function shows a similar trend as 1/1+x2 in the range of [-5,5] but outside that range, the graph diverges and error increases continuously as we go away from the origin.

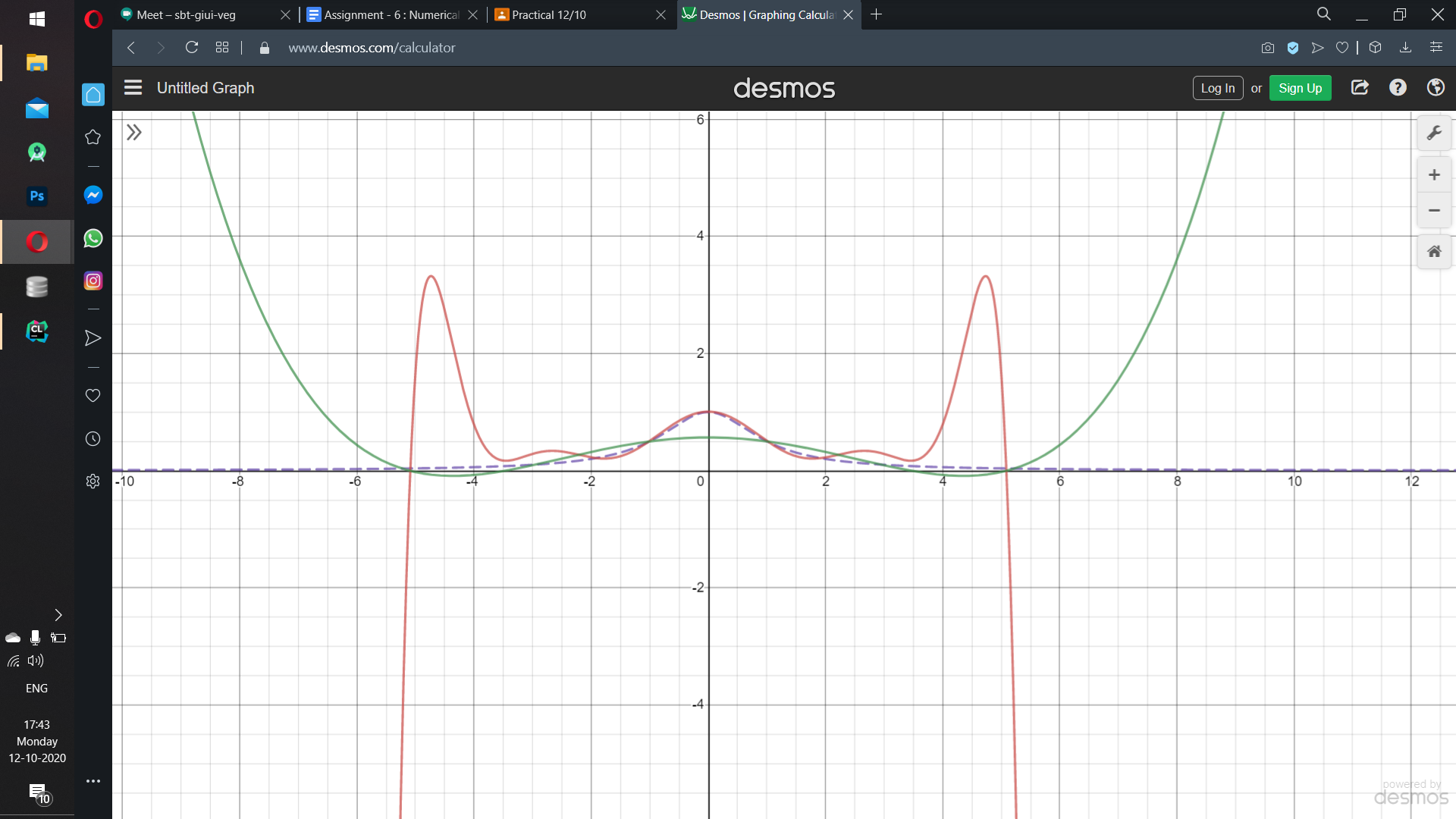
**Graph for g(x) for n = 11:**

Green line shows our f(x) **=** and orange points shows our interpolating polynomial g(x) = -0.00002242 x 10 + 0\* x9 + 0.00125 x8 + 0 x7 - 0.024 x6 + 0 \* x5 + 0.1971 x4 + 0 \* x3 - 0.6742 x2 - 5.5541 \* 10-17 x + 1.0076



**Conclusion:** In the above the graph, the plot follows a similar trend as the original function 1/1+x2 in the range of [-5,5] but continues to diverge as we go away from the origin.

**Combined graph for our interpolating polynomial g(x) for n = 6, 11 respectively:**



**Conclusion:** In the above graph, the violet line is of f(x) =**,** red lines are of g(x) {for n=11} and green dots are of g(x){for n=6}.

Both the graphs show almost similar trend in the range [-4,4] but outside that range, g(x) (for n =6) continues to increase i.e. error increases while g(x) (for n=11) continues to decrease i.e. negative error increases as one goes away from the origin.