Melih Kurtaran
24186

CS409 Project #2 Report

I developed a recursive adaptive integration MATLAB code that calculates the integral over an equilateral triangular domain.

The purpose of adaptive methods:

In Simpson's method, choosing the discretization steps is not easy because if it is chosen as too large, the calculated integration value will be inaccurate due to fast oscillation areas. However, if it is chosen as too small the result will be more accurate, but the amount of computation will increase, and we may do some unnecessary calculations for smooth areas. The motivation of adaptive methods is that different discretization will be selected at different areas of the function. For example, it will be small for oscillatory areas but will be large for smooth areas.

My code calculates the integration for different discretization steps for different areas in triangular domain.

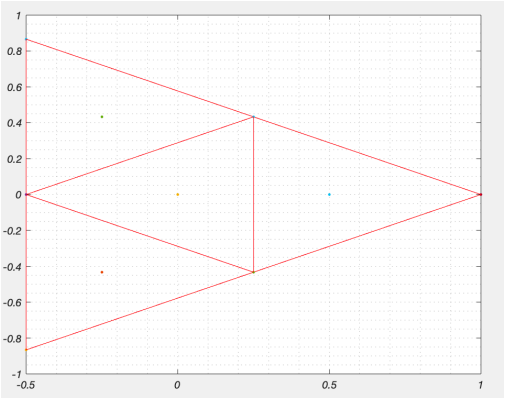
The inputs:

- f(x,y): the function to evaluate integral
- h: determines the size of the triangle
- tolerance: the minimum error which we demand

The outputs:

- The value of integration
- The graph to observe which part of the triangle called more

Simpson's method is used to calculate the integration, I divided the triangle to four equilateral triangles as the figure below.



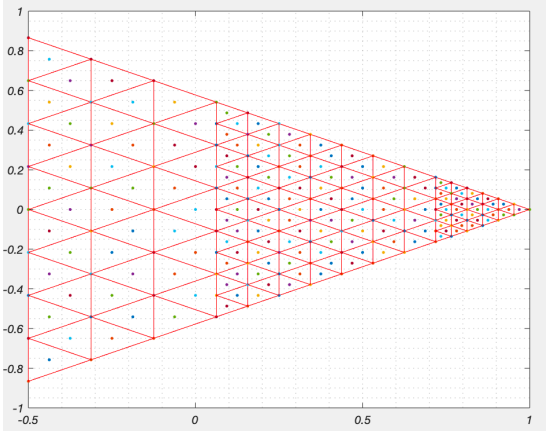
Melih Kurtaran
24186

The total area has been called as S1 and smaller triangles called as S21, S22, S23 and S24. I calculated the error as

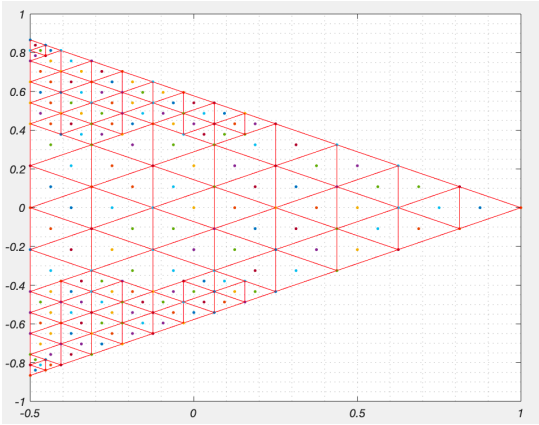
$$\text{Error} = S1 - (S21+S22+S23+S24)/4$$

For every iteration the error and the tolerance has been compared and also the tolerance is divided by 4. If the value of error is bigger than the tolerance the function calls itself for smaller triangles until when the value of the error reached smaller value than the value of tolerance.

The function calls itself for every 4 equilateral triangles. For some triangles it converges faster but for some triangles it is slower hence the function calls itself more which causes that triangle divided into more triangles which represented as below.



The triangle for the function e^x



The triangle for the function $e^{(y\hat{}2)}$

Sometimes, the function does not converge so the tolerance value which we divide into 4 for every iteration get smaller faster. To solve this problem, I defined a minTolerance variable which equals to square of the tolerance. I designed the recurrence not to decrease the tolerance than minTolerance, if tolerance reached minTolerance the function does not keep calling itself, it saves the function from infinite recursion call.

Melih Kurtaran
24186

Correction Analysis with Analytical Results

1) $f(x,y) = e^x$, h=1, tolerance=10^{-3}

Obtained Result:

```
integration =

   1.387898343640317
```

Analytical Result:

$$\int_{-\frac{1}{2}}^{1}\int_{\frac{x-1}{\sqrt{3}}}^{\frac{1-x}{\sqrt{3}}} e^x \, dy\, dx = \frac{2e^{3/2}-5}{\sqrt{3e}} \approx 1.387898292503944734671151663569325$$

We reached 6 digits accuracy.

If we changed our tolerance to 10^{-4}, the result is

```
integration =

   1.387898293249858
```

We reached 8 digits accuracy.

2) $f(x,y)=x^2.y^2$, h=2, tolerance=10^{-3}

Obtained Result:

```
integration =

   1.039230577455772
```

Analytical Result:

$$\int_{-1}^{2}\int_{\frac{x-2}{\sqrt{3}}}^{\frac{2-x}{\sqrt{3}}} x^2\, y^2 \, dy\, dx = \frac{3\sqrt{3}}{5} \approx 1.039230484541326376116467804903 5$$

We reached 6 digits accuracy.

If we changed our tolerance to 10^{-4}, the result is

```
integration =

   1.039230484904273
```

We reached 9 digits accuracy.

3) $f(x,y)=\sin(x+y)$, h=1, tolerance=10^{-2}

Obtained Result:

```
integration =

   0.010125163123335
```

Analytical Result:

$$\int_{-\frac{1}{2}}^{1}\int_{\frac{x-1}{\sqrt{3}}}^{\frac{1-x}{\sqrt{3}}} \sin(x+y)\, dy\, dx = 0.0100758023027237026'$$

We reached 3 digits accuracy.

If we changed our tolerance to 10^{-3}, the result is

```
integration =

   0.010075836263821
```

We reached 7 digits accuracy.

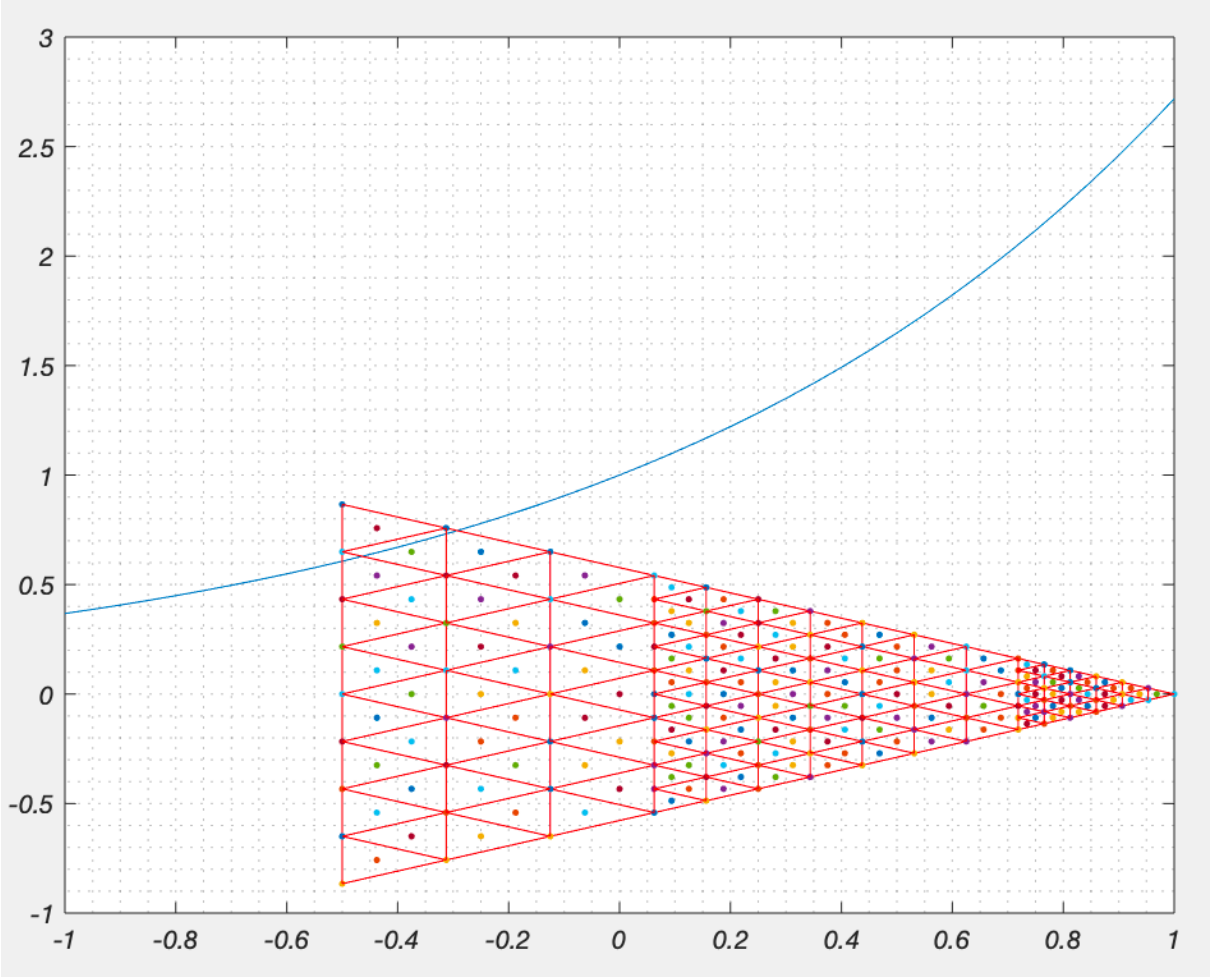
Melih Kurtaran
24186

Examples with their visualizations

1) $f(x,y)=e^x$

   h=1

   tolerance=10^{-3}



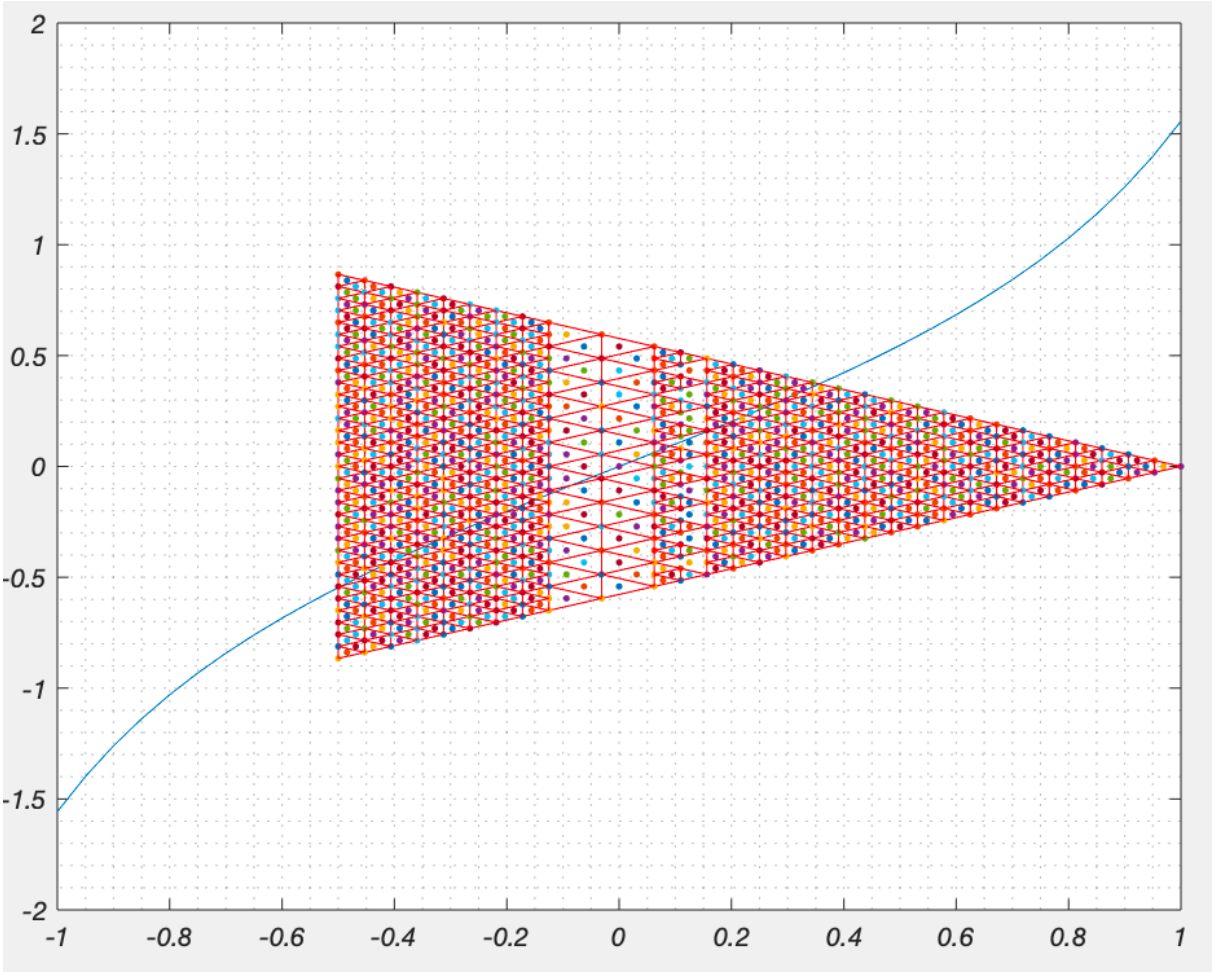
We can observe that when slope increases over x axes, we need smaller discretization steps to have more accurate result. The triangle is divided into more pieces at the right, but it is not divided into more pieces in the left because left part of the function is smoother than right.

Melih Kurtaran
24186

2) f(x,y)=tan(x)

h=1

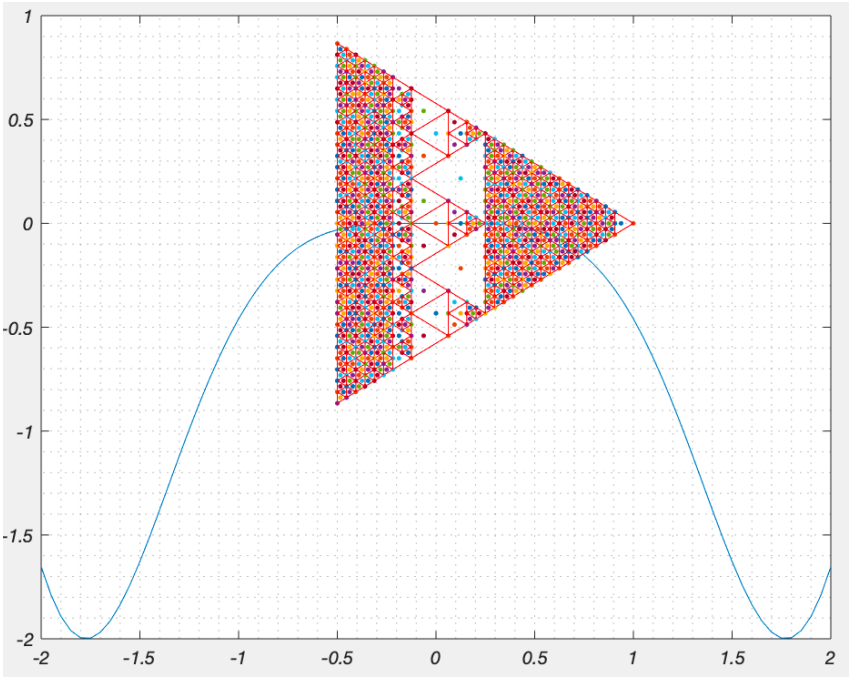
tolerance=10^{-3}



Since the tan(x) is smoother at the center, we have less division of triangles at the center around 0. However, we need smaller discretization steps while we are moving to 1 or -1. That's why triangle is divided into more pieces when it gets away from 0.

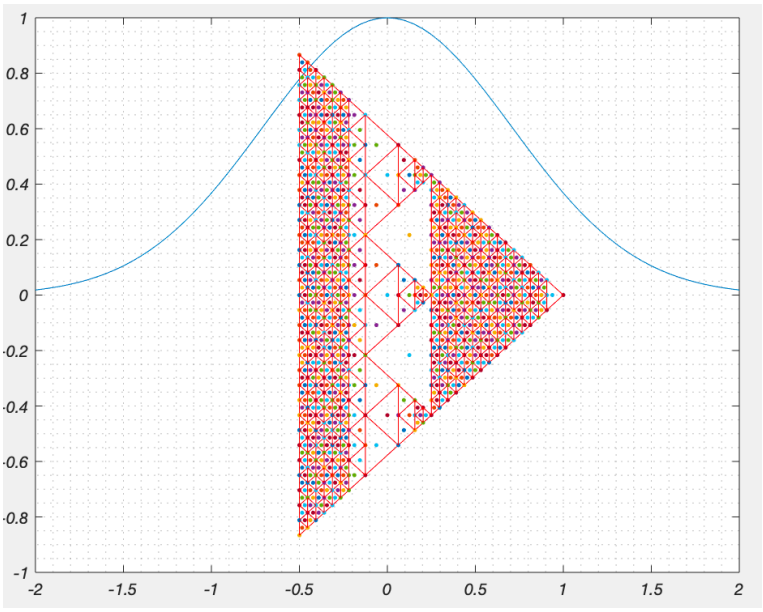
Melih Kurtaran
24186

3) $f(x,y)=\cos(x^2)$, h=1, tolerance=10^{-3}



We can observe that near to 0, the function is smoother so we don't need smaller discretization steps recursion call will be less. When it gets away from 0, it will need smaller discretization steps to calculate the integral more accurate, hence it will do more recursion calls and the triangle will be divided more pieces.

4) $f(x,y)=e^{-x\wedge 2}$, h=1, tolerance=10^{-3}



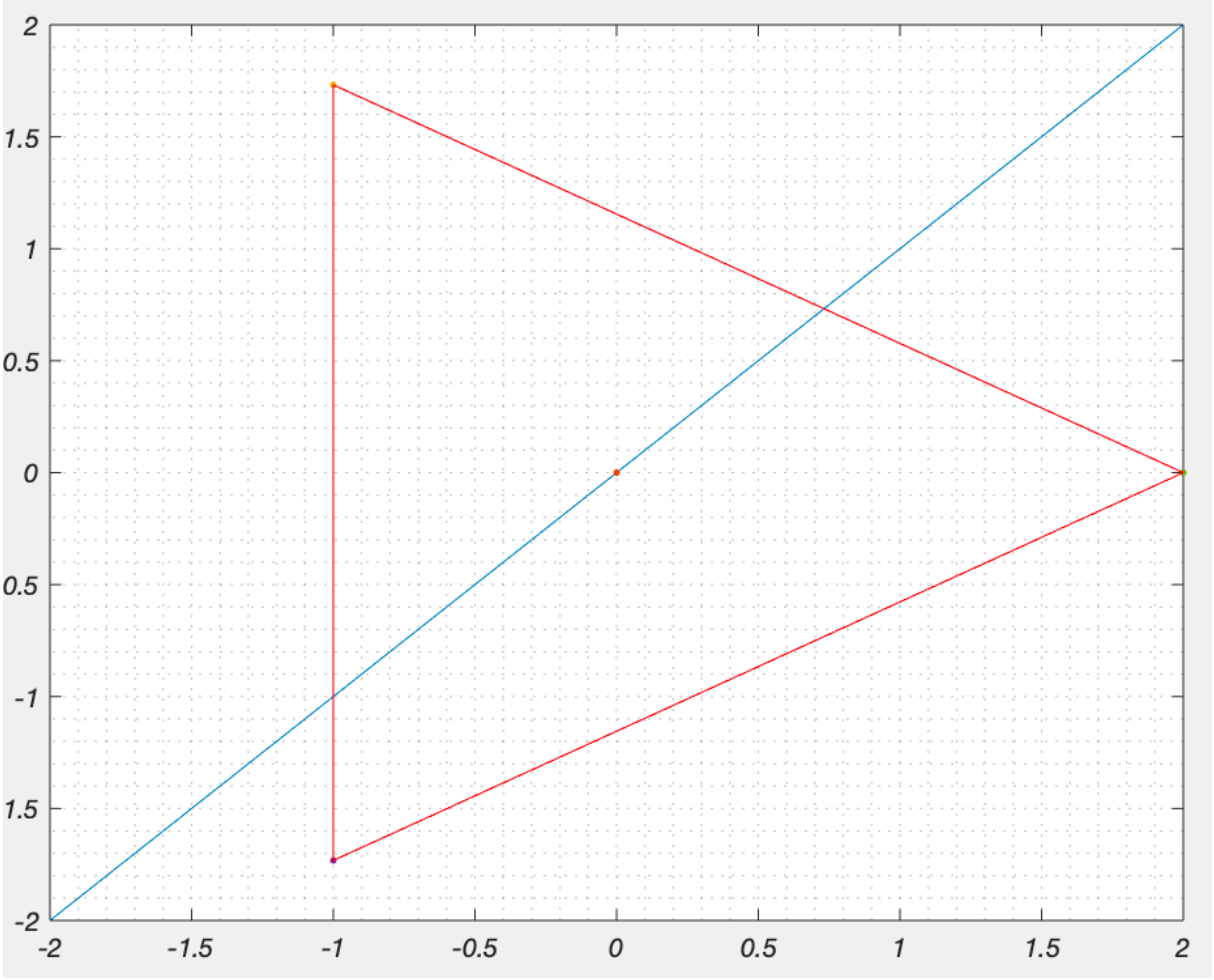
Another similar example

Melih Kurtaran
24186

5) f(x,y)= x

h=2

tolerance=10^{-8}



When we have f(x,y)=x, there is no oscillation and calculating the integral is easy. Even though, I choose tolerance as 10^{-8} , it does not do any recursive call and not divide to triangle. Because a large discretization step is good enough to calculate the integral as we demanded tolerance.