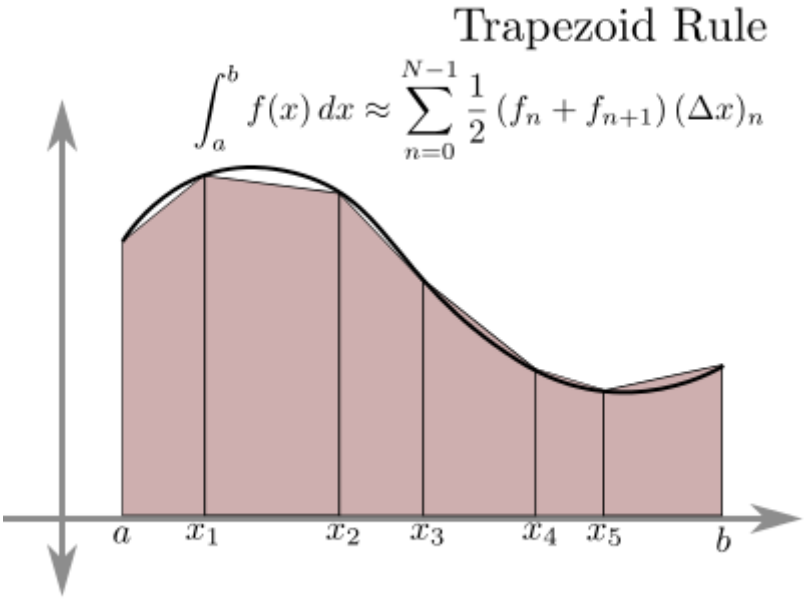
Trapezoidal Rule is a Numerical technique to find the definite integral of a function.



The function is divided into many sub-intervals and each interval is approximated by a Trapezium. Then the area of trapeziums is calculated to find the integral which is basically the area under the curve. THe more is the number of trapeziums used, the better is the approximation.

Formula:

$$\int_a^b f(x)dx \approx .\frac{h}{2}(f(a)+2f(a+h)+2f(a+2h)+....+f(b))$$

The following is a simple C Program that uses the trapezoidal rule to find the definite integral of a function.
Users, will have to change the function f in the following program to the function whose integral they want to find.

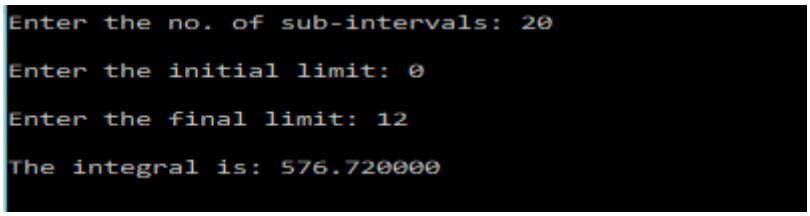
PROGRAM (SIMPLE VERSION):

```
/*********************************
 ********TRAPEZOIDAL RULE*********
  2017 (c) Manas Sharma - http://bragitoff.com
 *********************************/
#include<stdio.h>
#include<math.h>

/* Define the function to be integrated here: */
double f(double x){
  return x*x;
}

/*Program begins*/
main(){
  int n,i;
  double a,b,h,x,sum=0,integral;
  /*Ask the user for necessary input */
  printf("\nEnter the no. of sub-intervals: ");
  scanf("%d",&n);
  printf("\nEnter the initial limit: ");
  scanf("%lf",&a);
  printf("\nEnter the final limit: ");
  scanf("%lf",&b);
  /*Begin Trapezoidal Method: */
  h=fabs(b-a)/n;
  for(i=1;i<n;i++){
    x=a+i*h;
    sum=sum+f(x);
  }
  integral=(h/2)*(f(a)+f(b)+2*sum);
  /*Print the answer */
  printf("\nThe integral is: %lf\n",integral);
}
```

OUTPUT:



The above program returns a better approximation to the integral as the number of sub-intervals is increased. This might work for some applications, however, sometimes one might not want to deal with the number of sub-intervals, but rather the accuracy upto a certain decimal places. What I mean by accuracy is that sometimes you might just want the approximate value of integral upto a few decimal place. So you will have to keep on increasing the number of sub-intervals and check the value of the integral. If the integral for two subsequent no. of sub-intervals is within the accuracy/tolerance limit given by the user(or set by you) then the integral should be printed out.

The following program illustrates the process of achieving what I just explained and also uses a function called ‘trapezoidal’ that handles the integration part.

PROGRAM (Better Version):

```
/****************************************
 ****TRAPEZOIDAL RULE USING FUNCTION*******
  2017 (c) Manas Sharma - http://bragitoff.com
 ****************************************/
#include<stdio.h>
#include<math.h>

/* Define the function to be integrated here: */
double f(double x){
  return x*x;
}

/*Function definition to perform integration by Trapezoidal Rule */
double trapezoidal(double f(double x), double a, double b, int n){
  double x,h,sum=0,integral;
  int i;
  h=fabs(b-a)/n;
  for(i=1;i<n;i++){
    x=a+i*h;
    sum=sum+f(x);
  }
  integral=(h/2)*(f(a)+f(b)+2*sum);
  return integral;
}

/*Program begins*/
main(){
  int n,i=2;
  double a,b,h,eps,sum=0,integral,integral_new;

  /*Ask the user for necessary input */
  printf("\nEnter the initial limit: ");
  scanf("%lf",&a);
  printf("\nEnter the final limit: ");
  scanf("%lf",&b);
  printf("\nEnter the desired accuracy: ");
  scanf("%lf",&eps);
  integral_new=trapezoidal(f,a,b,i);

  /* Perform integration by trapezoidal rule for different number of sub-intervals
until they converge to the given accuracy:*/
  do{
    integral=integral_new;
    i++;
    integral_new=trapezoidal(f,a,b,i);
  }while(fabs(integral_new-integral)>=eps);

  /*Print the answer */
  printf("The integral is: %lf\n with %d intervals",integral_new,i);
}
```

OUPTUT:

```
Enter the initial limit: 5

Enter the final limit: 10

Enter the desired accuracy: 0.000001
The integral is: 0.142222
 with 43 intervals
```

Web App

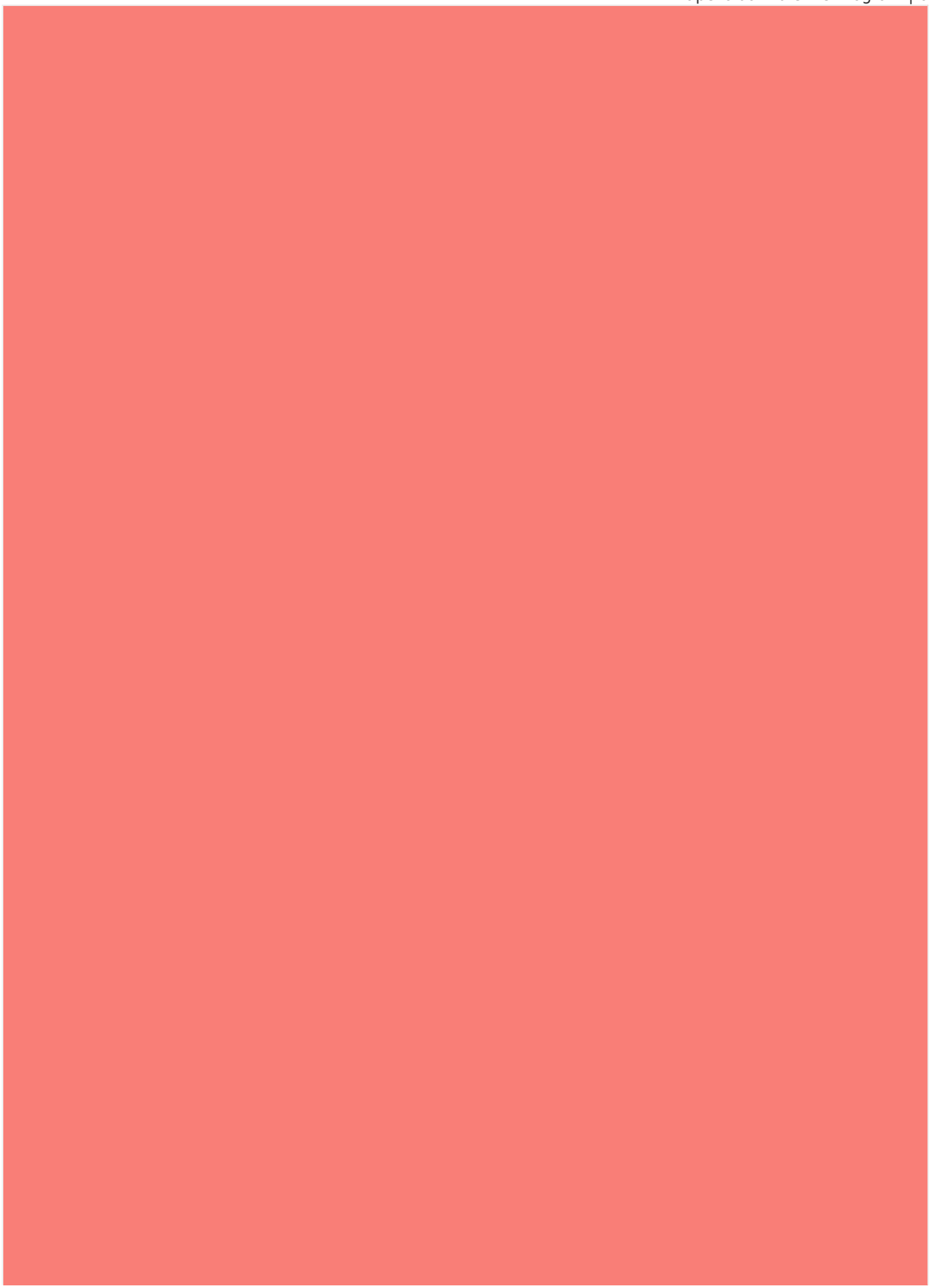
I have also created this web app so that you can try out the different numerical integration techniques online. You can also open this link to go full screen.

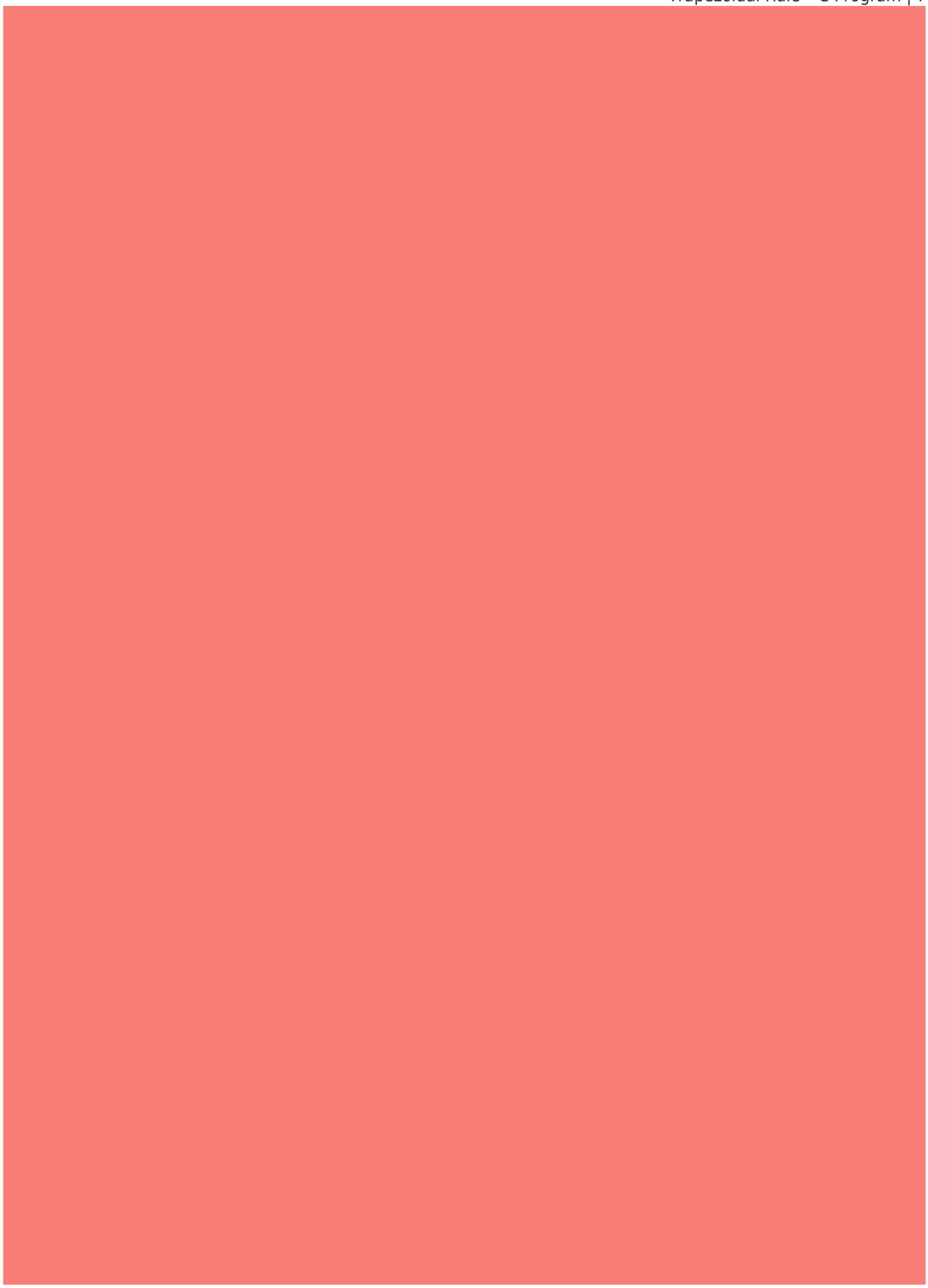
Your browser does not support iframes.

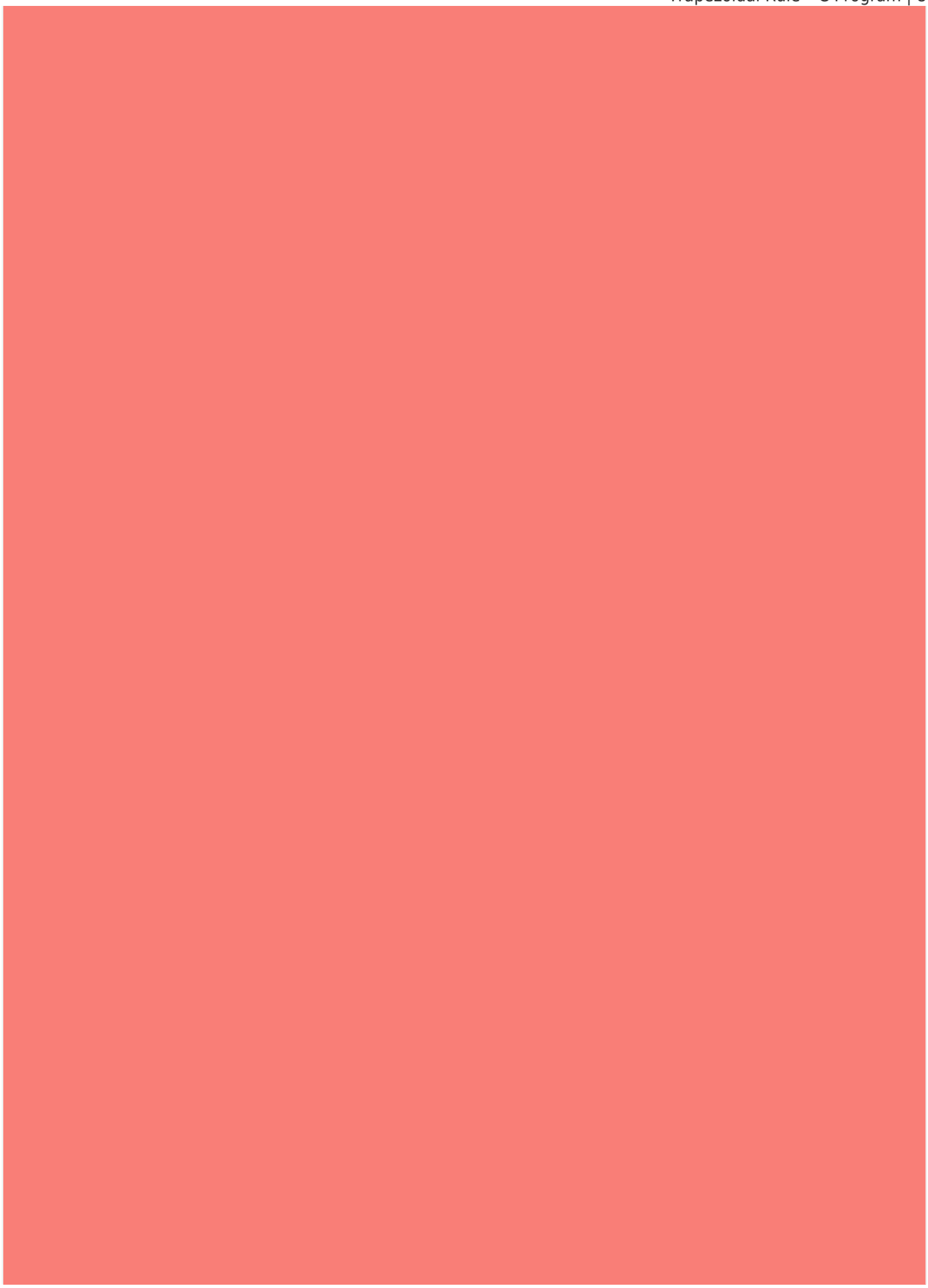
Manas Sharma

I'm a physicist specializing in computational material science with a PhD in Physics from Friedrich-Schiller University Jena, Germany. I write efficient codes for simulating light-matter interactions at atomic scales. I like to develop Physics, DFT, and Machine Learning related apps and software from time to time. Can code in most of the popular languages. I like to share my knowledge in Physics and applications using this Blog and a YouTube channel.

manas.bragitoff.com/







Share this:

Click to share on Facebook (Opens in new window)

Click to share on Twitter (Opens in new window)

Click to share on WhatsApp (Opens in new window)

Click to share on Pinterest (Opens in new window)

Click to share on Reddit (Opens in new window)

Click to share on LinkedIn (Opens in new window)

Click to email a link to a friend (Opens in new window)

Click to print (Opens in new window)

Click to share on Tumblr (Opens in new window)

Click to share on Pocket (Opens in new window)

Click to share on Telegram (Opens in new window)

[wpedon id="7041" align="center"]