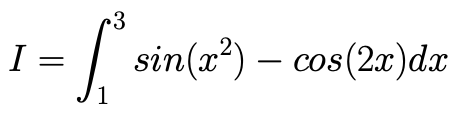
**Integration of Functions**

**“Composite Simpson rule - Romberg’s method - Gauss-Legendre method”**

**Brief description:**

This first part of the work consists of the creation of different codes for the resolution of integrals, in this case demonstrating their usefulness based on the function (Fig. 1).

  
Fig .1: *Integral to be solved*

The codes created have been based on the use of the Fortran programming language, which is a programming language used mainly in mathematics and scientific calculation applications. From this language and the use of various loops and subroutines, the integral calculation has been carried out. The first part is divided into two different codes, the first one combining the composite Simpson rule and Romberg's method. And in the second one, with the help of a calculation subroutine provided by the professors, the Gauss-Legendre method is used.

**Algorithm:**

In this part of the paper, the algorithm (calculation method) used in each of the codes is explained in a general way. The three calculation methods (Composite Simpson rule - Romberg's method - Gauss-Legendre method) used are explained.

* Composite Simpson rule:

The Composite Simpson rule is based on a general procedure which is to break up the length of the integral ([a,b]) into small subintervals, then apply Simpson's rule to each subinterval. Then, sum the results of each calculation to produce an approximation over the entire integral.

However, it is often the case that the function we are trying to integrate is not smooth over the interval. Typically, this means that either the function is highly oscillatory or lacks derivatives at certain points. In these cases, Simpson's rule may give very poor results. One common way of handling this problem is by breaking up the interval [a,b][a,b] into n > 2 n>2small subintervals. Simpson's rule is then applied to each subinterval, with the results being summed to produce an approximation for the integral over the entire interval. This sort of approach is termed the *composite Simpson's rule*.

At the end, it must be said that this formulation splits the interval [a,b][a,b] in subintervals of equal length. In practice, it is often advantageous to use subintervals of different lengths and concentrate the efforts on the places where the integrand is less well-behaved.

* Romberg's method:

Numerical Integration is simply the approximation of integrals and is useful for integrals that cannot be evaluated by the special formulas. Romberg's method, used in numerical analysis, is used to estimate the definite integral. by applying Richardson extrapolation repeatedly on the Composite trapezoidal rule. The estimates generate a triangular array. It is a Newton-Cotes formula, it evaluates the integrand at equally spaced points. The integrand must have continuous derivatives, though fairly good results may be obtained if only a few derivatives exist. It is defined as an extrapolation technique which allows us to take a sequence approximate solution to an integral and calculate a better approximation and It also assumes that the function we are integrating is sufficiently differentiable.

At the end we know that the Romberg integration (Rnn) can be rapidly converged with the increase in n. It is calculated by increasing the number of partitions to double from 2 to N=2n.

The general process is as follows: first, we compute the h, then the first column of the first two rows using CTR is computed, next, the next columns in that row using Romberg’s Formula for i>1 is computed and at the end, we repeat the previous steps for every row until it converges to the needed solution.

* Gauss-Legendre method:

To finish with the explanation of the different methods, we must talk about the Gauss–Legendre method. In general, the Gauss-Legendre methods are defined as a a family of numerical methods for ordinary differential equations, which are implicit Runge–Kutta methods. More specifically, they are collocation methods based on the points of Gauss–Legendre quadrature.

Gauss quadrature deals with integration over a symmetrical range of x from -1 to + 1. The important property of Gauss quadrature is that it yields exact values of integrals for polynomials of degree up to 2n – 1. Gauss quadrature uses the function values evaluated at a number of interior points (hence it is an open quadrature rule) and corresponding weights to approximate the integral by a weighted sum.

**Implementation of the code:**

This final part of the paper discusses the general structure of the code used for each of the methods, their parts, subroutines and also the main variables used. It must be said that the programs have been tested for correct operation.

It must be said that this part of the work is divided into two different codes. The first one combines the Composite Simpson rule and the Romberg method, and the second one is based on the Gauss-Legendre method.

The first code has as main variables the following:

conv 🡪 convergence criteria (can be adjusted from main program).

Areas 🡪 integral calculated using simpson's rule, since integral is a signed area under the curve.

r 🡪 matrix containing successive integral values calculated from romberg's method.

msz 🡪 size of matrix r.

ipos 🡪 a two-value array keeping the position of the converged result.

To explain the general structure, the code is divided into four distinct parts: the subroutine belonging to the Composite Simpson rule, the subroutine belonging to the Romberg's method, the main program where the calculation and global procedure are generated, and finally, the definition of the function to be calculated.

In the first part, we start by doubling the number of sub-intervals from n=2. We then proceed to calculate the increment using the h function defined in the code. The next step is based on two loops to introduce the contribution of the two different types of points (odd and even) and finally, we calculate the error between successive iterations to consider the final result as correct or to remain with the previous result until convergence. The information obtained in each iteration is the following: number of subintervals used (n), integral value ("areas"), iteration number (iter) and the error between two latest consecutive values.

For the second subroutine, the dimensions of the matrix for the results and of the vector of the successive values of h are defined. Then a loop is defined for the creation of the different successive values of h, inside this loop, three other loops are defined: the first one is defined to run the calculation of the following two loops along the values of k, the second one is used to generate the summation to obtain the third value of r(k,1) and finally, the last one allows the calculation of the whole row up to the diagonal. At the end of the subroutine two ifs are used: the first one to evaluate the convergence at each time and the last one to calculate the values of the k-th row.

Then the code of the main program is written, where the two subroutines are simply called and the results for each iteration and the final result in the last iteration are written on the screen. It is necessary to say that a do loop is generated to write the results matrix at the end of the calculation.

In the final part of the code, you can see how the function to be integrated is defined using the "implicit double precision" command, which gives better results.

The second code has as main variables the following:

n 🡪 number of quadrature points.

t 🡪 vector containing Gauss-Legendre zeros (points to employ) for particular n

w 🡪 array/vector containing weights for corresponding points

X1, X2 🡪 bounds of integration.

XM, XL 🡪 parameters (computed) for variables change.

par\_m and par\_c 🡪 corresponding parameters for change of variables.

The general structure of the second code is divided into three distinct parts, the first consists of a subroutine belonging to the Gauss-Legendre method, followed by the definition of the main program where the organization of the calculation is carried out and finally, the function to be integrated in this case is defined.

The only subroutine in the code defines the procedure followed by the Gauss-Legendre method. First the different precisions for the calculation are defined, then the value of m is calculated from the number of quadrature points. The change of variable (XM-XL) is then performed until the loop which performs the calculations inside it along all the roots. Within this, a loop is placed to generate the corresponding Legendre polynomial evaluated at z and the derivative of this is calculated using the lower degree polynomial. Finally, an if is used to evaluate the convergence of the calculation and terminate or recalculate certain parameters.

Then the code corresponding to the main program begins. Here the arbitrary dimensions of two differentiated vectors are defined. Then we define the extremes of the integral and enter a do while loop to call the subprogram.

This loop is used to evaluate the final results according to the error defined in the previous step together with different parameters. If this point is exceeded by the results of any of the calculations, the results are displayed on the screen and the calculation is terminated.

In the final part of the code, you can see how the function to be integrated is defined using the "implicit double precision" command, which gives better results.