In this code we compute $\int \sin(x^2) - \cos(2x)$ in the interval $[1, 3]$, with a threshold 10^{-8}, because of that we set the variable `lowl`, `upl` and `threshold`, in the starts lines. The function in the argument of the integral is in the `FUNCTION`. So we can change the function only changing that part of the program.

For the methods of integration we only need to call the respective `SUBROUTINE` for the three first ones, the last one needs some changes because of the integration limits, that will be comment in the section 4.

1. Trapezoidal rule

```
1   SUBROUTINE Trapezoid(lowl,upl,threshold)
2   IMPLICIT NONE
3   REAL*8 :: lowl, upl, threshold, e, h, p, s, integra, er
4   INTEGER :: i, n, iter
5
6   !n how many trapezoids, e and er for threshold
7   n = 2; e = 0.d0; er = 1.d0; iter = 0
8   WRITE(*,*) "Integral value", "       error"
9   DO WHILE (er>threshold)          !until converge
10      h = (upl-lowl)/n             !base trapezoid size
11      p = (h/2.)*(f(lowl)+f(upl))
12      s = 0.d0
13      DO i = 1, n-1
14          s = s + h*f(lowl+i*h)
15      ENDDO
16      integra = p+s                !integral value
17      n = 2*n                      !twice the number of trapezoids
18      er = ABS(e - integra)
19      e = integra
20      iter = iter + 1
21      WRITE(*,FMT='(F14.5,F14.5)') integra, er
22  ENDDO
23  WRITE(*,FMT='((A),F8.6,(A),I4,(A))') "The integral value &
24              is: ", integra, " computed with ", iter, " iterations"
25  ENDSUBROUTINE Trapezoid
```

Starting with 2 trapezoids in the variable `n`, the `SUBROUTINE` will duplicate the number of trapezoid until have a difference between the calculations less than the `threshold`. That what express the line 9.

Knowing that in this approximation we need to compute:

$$I = \frac{b-a}{2N}\left[f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i)\right] \tag{1}$$

We start in the iteration 0 (`iter=0`), and to save the value of previous integration using the variables `e` and `er`, now in the loop we compute the `h`, which is equal to $(b-a)N^{-1}$. Also the firsts two terms of 1., then is necessary other loop for compute the sum (3rd part of 1.), therefore the integral is the addition of these terms. Note that we are computing the Equation with $(b-a)(2N)^{-1}$ multiplying every term, and x_i means a plus small step, h.

With the value computed the `SUBROUTINE` duplicate number of trapezoids, calculate the absolute value of the difference between the previous integral and the current integral, finally add 1 to the number of iterations and write in the screen the value with the difference between the previous step.

Additionally the `SUBROUTINE` write the final value with the number of iterations needed to get the value.

Is also possible compute the factor of the error in this approximation, knowing the error is:

$$Error = -N\frac{h^3}{12}f^2(\xi) \tag{2}$$

However we cannot know the value of ξ, all we know is $\xi \in [a, b]$, so is not very useful the knowledge of these factor of $f^2(\xi)$.

2. Simpson rule

```
1   SUBROUTINE Simpson(lowl,upl,threshold)
2   IMPLICIT NONE
3   REAL*8 :: lowl, upl, threshold, e, h, p, s1, s2, integra, er
4   INTEGER :: i, n
5
6   !n how many intervals, e and er for threshold
7   n = 2; e = 1.d0; er = 1.d0
8   WRITE(*,*) "Integral value", "       error"
9   DO WHILE (er>threshold)
10      h = (upl-lowl)/n
11      p = f(lowl)+f(upl)
12      s1 = 0.d0;  s2 = 0.d0
13      DO i = 2, n-2, 2
14          s1 = s1 + f(lowl+i*h)
15      ENDDO
16      DO i = 1, n-1, 2
17          s2 = s2 + f(lowl+i*h)
18      ENDDO
19      integra = (h/3.d0)*(p + 2.d0*s1 + 4.d0*s2)
20      n = 2*n
21      er = ABS(e - integra)
22      e = integra
23      WRITE(*,FMT='(F14.5,F14.5)') integra, er
24  ENDDO
25  WRITE(*,FMT='((A),F8.6,(A),I4,(A))') "The integral value &
26              is: ", integra, " computed with ", n, " points"
27  ENDSUBROUTINE Simpson
```

For this approximation we need to compute the next expression:

$$I = \frac{b-a}{3N}\left[f(a) + f(b) + 2\sum_{\substack{even\\ i=2}}^{N-2} f(x_i) + 4\sum_{\substack{odd\\ i=1}}^{N-1} f(x_i)\right] \tag{3}$$

As in the `SUBROUTINE Trapezoid` we start with `n=2` (how many intervals), `e` and `er` to save the values for error.

Starting with the loop until the convergence criteria, we compute the auxiliary variable `h`, and the first two terms of Equation 3. Continuing with the two sum, one for the even and the other one for odd terms, that is the reason for the last `2` in lines 13 and 16, meaning that the loops are skipping one of the terms.

Finally, is only necessary add the three terms and multiply by `h/3` to get the value of integral. Then the program duplicate the number of intervals, compute the difference between the current value and the previous one, besides saving this current value.

The program also write the final value with the number of necessary points, additionally of the value table before.

3. Rombergs method

```
    SUBROUTINE Romberg(R,d,lowl,upl,threshold)
    IMPLICIT NONE
    REAL*8 :: lowl, upl, threshold, e, er, s, aux1, aux2
    INTEGER :: i, j, k, n, d
    REAL*8, DIMENSION(d) :: h
    REAL*8, DIMENSION(d,d) :: R
    INTEGER, DIMENSION(2,2) :: flag
    CHARACTER(LEN=60) :: strfor, str

    DO k = 1, d
        h(k) = (upl - lowl)/(2.d0**(k-1))
    ENDDO

    R(1,1) = (h(1)/2.d0)*(f(lowl)+f(upl))

    e = 1.d0; er = 1.d0
    DO k = 2, d           !R_{k,1}   k>1
        s = 0.d0
        DO i = 1, (2**(k-2))
            s = s + f(lowl+(2*i -1)*h(k))
        ENDDO
        R(k,1) = 0.5d0*(R(k-1,1) + h(k-1)*s)
        er = ABS(e - R(k,1))
        e = R(k,1)
        IF (er > threshold) THEN
            flag(1,1) = k+1; flag(1,2) = 1
        ENDIF
    ENDDO

    e = 1.d0; er = 1.d0
    j = 2; k = j          !R_{k,j}    k>=j>1
    DO WHILE ( (k <= d).AND.(j <= d) )
        aux1 = R(k,j-1) - R(k-1,j-1)
        aux2 = 4.d0**(j-1) - 1.
        R(k,j) = R(k,j-1) + aux1/aux2
        er = ABS(e - R(k,j))
        e = R(k,j)
        IF (er > threshold) THEN
            flag(2,1) = k+1; flag(2,2) = j
        ENDIF
        IF (k == d) THEN
            j = j + 1
            k = j
        ELSE
            k = k + 1
        ENDIF
    ENDDO

    WRITE(*,FMT='(A,F8.6)') 'The integral value is: ',R(flag(1,1),flag(1,2))
    WRITE(*,*) 'And is, at least, in the positions: '
    DO i = 1, 2
        WRITE(*,FMT='(A3,I2,A1,I2,A1)') 'R_{',flag(i,1),',',flag(i,2),'}'
    ENDDO

    WRITE(*,*)
    WRITE(*,*) 'in: '
    WRITE(*,FMT='(A5,*(A5,I2,A2))') '*****', (' R_{k,',i,'} ', i=1,10)
    DO i = 1, 10
        strfor = '(A2,I2,'//TRIM(str(i))//'(F9.5))'
        WRITE(*,FMT=strfor) 'k=',i,(R(i,j), j=1,i)
    ENDDO
    ENDSUBROUTINE Romberg
```

In this numerical method we need more steps, starting with a array for the h_k, this `SUBROUTINE` compute until `k=20`, however only write in the screen the first ten.

Then we compute the first R term, necessary to get the next approximations. With the formula:

$$R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right] \tag{4}$$

After that we can continue with the $R_{k,1}$, setting the variables `e` and `er` for the convergence criterion. Start the loop until `k=20` does not matter if we pass the convergence criterion the next formula:

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right] \tag{5}$$

where the variable `s` is to compute the sum, and since `R` is a matrix, we can easily save the values. As the previous `SUBROUTINE`'s, `er` has the value of the difference between the current and the previous integral value, and `e` saves the current value.

Also we have a condition, until we are not at least in the convergence criteria the program is rewriting the coordinates of the matrix where the we are. Therefore, when we are in the convergence criteria `flag` will get where in `R` has the convergence criteria.

With that values we can compute the other columns in R, with the next formula:

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1} \tag{6}$$

for that we use a `DO WHILE` with the dimension of the matrix R (`d`) as criteria, starting in the second column and second row.

The numerator and denominator in 6, are computed in the variables `axu1` and `aux2`, respectively. As before, the convergence criteria are in the variables `er` and `e` and when we get the convergence criteria satisfied. The second row in `flag` will get where in R is satisfied the convergence criteria. We plus one in `k` until the dimension of R, in that case we go to the next column (lines 43 to 48).

After computing R the program write in screen the integral value and the position in R, using the coordinates in `flag`, also write R until $R_{10,10}$. For that we use the `FUNCTION str` included in this program to convert a number into a character.

4. Gauss-Legendre method

```
1  aux1 = (upl-lowl)/2.d0
2  aux2 = (upl+lowl)/2.d0
3  n=2
4  e = 0.d0; er = 1.d0
5  DO WHILE ( (er > threshold).AND.(n<=10) )
6      ALLOCATE(t(n),w(n))
7      CALL sub_GauLeg(-1.d0,1.d0,t,w,n)
8      s = 0.
9      DO i = 1, n
10         s = s + w(i)*f(aux1*t(i) + aux2)
11     ENDDO
12     integra = aux1*s
13     er = ABS(integra - e)
14     e = integra
15     DEALLOCATE(t,w)
16     n = n+1
17 ENDDO
18
19 WRITE(*,FMT='(A,F8.6,A,I2)') "The integral value is: ", integra,&
20         " with quadrature ",n-1
```

In this part of the program we use the `SUBROUTINE sub_GauLeg` given by the teachers. Since the Gauss-Legendre method is only for integrals in the interval $[-1, 1]$ we need to change the limits, that is really easy knowing:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}u + \frac{a+b}{2}\right) du \tag{7}$$

The fractions are in the variables `axu1` and `aux2`, then start the `DO WHILE` with the condition of the convergence criteria and the maximum number of points to quadrature (starts in `n=2`). As all the previous one the variables `e` and `er` is to save the previous value and get the difference between the current value and the past one.

To save the value of the `SUBROUTINE` we `ALLOCATE`: `t` and `w`, not forgetting `DEALLOCATE` at the final of the loop. After calling the `SUBROUTINE` we do the sum with the values of `t` and `w` gotten by the `SUBROUTINE`, not forgetting what since we are changing the limits is necessary use the auxiliary variables `aux1` and `aux2`.

Finally write in the screen the value of the integral with the quadrature number.