PRIMA PROVA SCRITTA: ELENCO DOMANDE

1) Descrivere la propria esperienza acquisita sullo sviluppo e l'uso di modelli numerici applicati alla geofisica
   Describe your experience acquired in the development and use of numerical models applied to geophysics

2) Descrivere la propria esperienza di utilizzo di sistemi Unix/Linux e calcolo numerico
   Describe your experience in the use of Unix/Linux systems and numerical calculus

3) Descrivere la propria esperienza di sviluppo di modelli numerici e calcolo parallelo
   Describe your experience in the development of numerical models and parallel calculus

SECONDA PROVA SCRITTA

1 Traccia

Questo modulo in Python calcola un numero. Scrivi un modulo in un
altro liguaggio a tua scelta (per esempio Fortran) per calcolare lo stesso numero.

This module in Python computes a number. Write a module in another
language (up to you, e.g. Fortran) to compute the same number.

```
--------------------------------------------------------------------------------

#!/usr/bin/env  python3

import numpy as np

def hard_function(x):
    return((1/np.sqrt(2*np.pi))*np.exp(-(x**2)/2))


def integrate(x1,x2,func,n=100000):
    X=np.linspace(x1,x2,1000)
    y1=0
    y2=max((func(X)))+1
    print(x1,x2,y1,y2)
    area=(x2-x1)*(y2-y1)
    check=[ ]
    xs=[ ]
    ys=[ ]
    for i in range(n):
        x=np.random.uniform(x1,x2,1)
        xs.append(x)
        y=np.random.uniform(y1,y2,1)
        ys.append(y)
        if abs(y)>abs(func(x)) or y<0:
            check.append(0)
        else:
            check.append(1)
```

```
    return(np.mean(check)*area,xs,ys,check)

print(integrate(0.3,2.5,hard_function)[0])
```

2 Traccia

Questo modulo in Fortran90 calcola un numero. Scrivi un modulo in un altro liguaggio a tua scelta (per esempio Python) per calcolare lo stesso numero.

This module in Fortran90 computes a number. Write a module in another language (up to you, e.g. Python) to compute the same number.

```
 Subroutine simpson(f,a,b,integral,n)
!=====================================================
! Integration of f(x) on [a,b]
! Method: Simpson rule for n intervals
! written by: Alex Godunov (October 2009)
!-----------------------------------------------------
! IN:
! f   - Function to integrate (supplied by a user)
! a         - Lower limit of integration
! b         - Upper limit of integration
! n   - number of intervals
```

```
! OUT:
! integral - Result of integration
!=======================================================
implicit none
double precision f, a, b, integral,s
double precision h, x
integer nint
integer n, i

! if n is odd we add +1 to make it even
if((n/2)*2.ne.n) n=n+1

! loop over n (number of intervals)
s = 0.0
h = (b-a)/dfloat(n)
do i=2, n-2, 2
   x   = a+dfloat(i)*h
   s = s + 2.0*f(x) + 4.0*f(x+h)
end do
integral = (s + f(a) + f(b) + 4.0*f(a+h))*h/3.0
return
end subroutine simpson
```

3 Traccia

Questo modulo in Fortran90 calcola un numero. Scrivi un modulo in un
altro liguaggio a tua scelta (per esempio Python) per calcolare lo stesso numero.

This module in Fortran90 computes a number. Write a module in another language (up to you, e.g. Python) to compute the same number.

```
module newton_raphson

    implicit none

contains

subroutine find_root( f, xinit, tol, maxiter, result, success )

    interface
        real function f(x)
            real, intent(in) :: x
        end function f
    end interface

    real, intent(in)     :: xinit
    real, intent(in)     :: tol
    integer, intent(in)  :: maxiter
    real, intent(out)    :: result
    logical, intent(out) :: success
```

```
real              :: eps = 1.0e-4
real              :: fx1
real              :: fx2
real              :: fprime
real              :: x
real              :: xnew
integer             :: i

result  = 0.0
success = .false.

x = xinit
do i = 1,max(1,maxiter)
   fx1    = f(x)
   fx2    = f(x+eps)
   write(*,*) i, fx1, fx2, eps
   fprime = (fx2 - fx1) / eps

   xnew   = x - fx1 / fprime

   if ( abs(xnew-x) <= tol ) then
      success = .true.
      result  = xnew
      exit
   endif

   x = xnew
   write(*,*) i, x
```

```
    enddo
end subroutine find_root
end module
```