$$I = \int_a^b f(x)\,dx = \frac{N_1}{N} \cdot c \cdot (b-a)$$

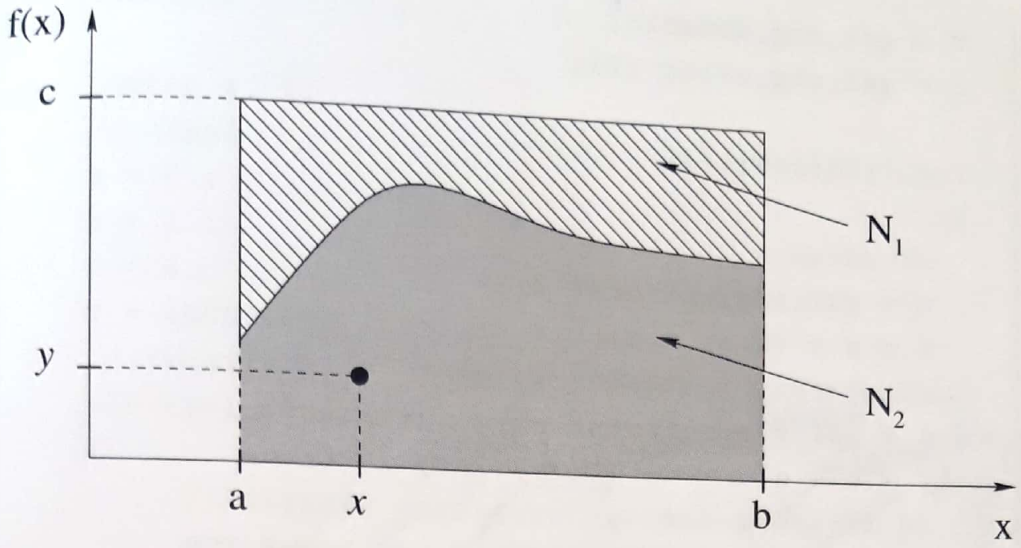

Figura 3.2: Integrare prin metoda Monte-Carlo.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_rng.h>
#include <mpi.h>

// functia de integrat f(x)
double f( double x)
{
  return 4.*sqrt(1.-x*x);
}

// functia care calculeaza integrala
double calculeaza_integrala( double a, double b, double c,
    long N)
{
  long i;
  const gsl_rng_type * T;
  gsl_rng * r;
  double u;
  double x, y;
  long N1=0, N2=0;
  double I;

  gsl_rng_env_setup();
```

```
  T = gsl_rng_default;
  r = gsl_rng_alloc (T);

  for (i=0;i<N;i++)
  {
    // alege x
    u = gsl_rng_uniform( r);
    x = a + (b-a) * u;
    // alege y
    u = gsl_rng_uniform( r);
    y = c * u;
    // verifica f(x)<y
    if ( y < f(x) )
       {
       N1 = N1 + 1;
       }
    else
       {
       N2 = N2 + 1;
       }
  }
  // valoarea integralei
  I = (double)N1/(double)N * (b-a)*c;

  return I;
}

int main( int argc, char **argv)
{
int Np, rank;
long i;
// limitele de integrat
double a, b, c;
double N;
double c1, c2;
long Ni;

MPI_Init( &argc, &argv);
MPI_Comm_size( MPI_COMM_WORLD, &Np);
MPI_Comm_rank( MPI_COMM_WORLD, &rank);

// MASTER -- rang 0
```

```
if (rank==0)
{
    double s, I;
    // input
    a = 0.;
    b = 1.;
    c = 4.;
    N = 100000000;
    // trimite a, b, c catre procesele de tip worker
    for (i=1;i<Np;i++)
    {
        // trimite intervalul de integrare [a,b]
        MPI_Send( &a, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
        MPI_Send( &b, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
        // trimite valoarea maxima c
        MPI_Send( &c, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
        // trimite numarul de puncte
        Ni = (long) ( (double)N/(double)(Np-1) );
        MPI_Send( &Ni, 1, MPI_LONG, i, 0, MPI_COMM_WORLD);
    }
    // primeste rezultatele integralelor si calculeaza
       rezultatul final
    I = 0.;
    for (i=1;i<Np;i++)
    {
        MPI_Recv( &s, 1, MPI_DOUBLE, MPI_ANY_SOURCE, 0,
           MPI_COMM_WORLD,
                  MPI_STATUS_IGNORE);
        I = I + s;
    }
    I = I / (double)(Np-1);
    // afiseaza rezultatul integralei
    printf( "I = %f \n", I);
}

// WORKERS -- rangurile 1, 2, ... ,
else
{
double s;
// primeste intervalul de integrare [a,b]
MPI_Recv( &a, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
   MPI_STATUS_IGNORE);
```

```
MPI_Recv( &b, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
    MPI_STATUS_IGNORE);
// primeste valoarea maxima c
MPI_Recv( &c, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
    MPI_STATUS_IGNORE);
// primeste numarul de puncte Ni
MPI_Recv( &Ni, 1, MPI_LONG, 0, 0, MPI_COMM_WORLD,
    MPI_STATUS_IGNORE);
printf( "%f %f %f %i\n", a, b, c, Ni);
// calculeaza integrala
s = calculeaza_integrala( a, b, c, Ni);
printf("s=%f\n", s);
// trimite rezultatul catre MASTER
MPI_Send( &s, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
}

MPI_Finalize();

return 0;
}
```