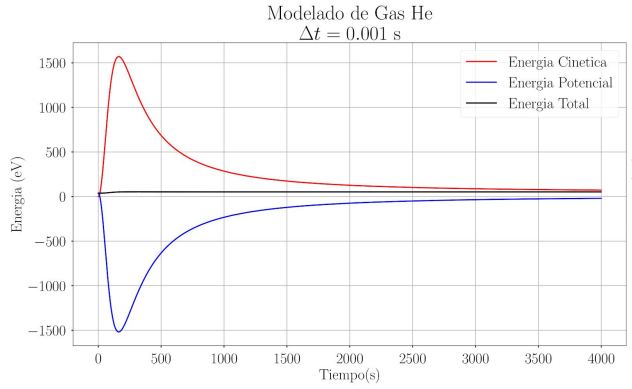
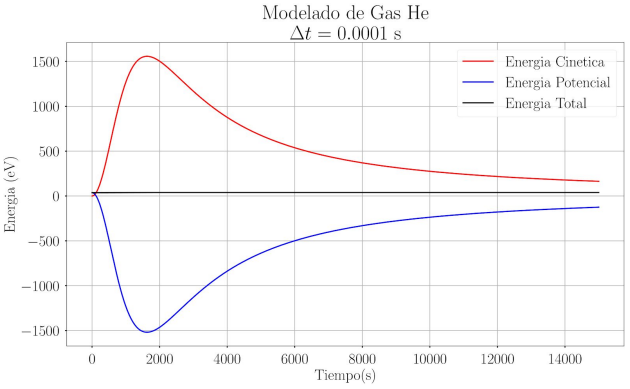
**MODELADO MOLECULAR II, TAREA 02**

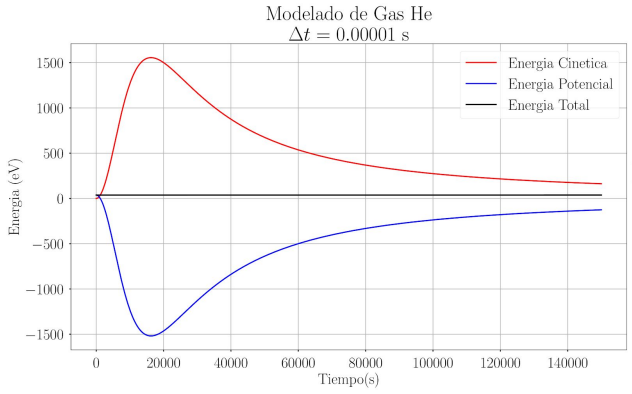
**Integrantes: Karina Chiñas Fuentes (NT70484), Natalia González Vázquez (NT703441).**

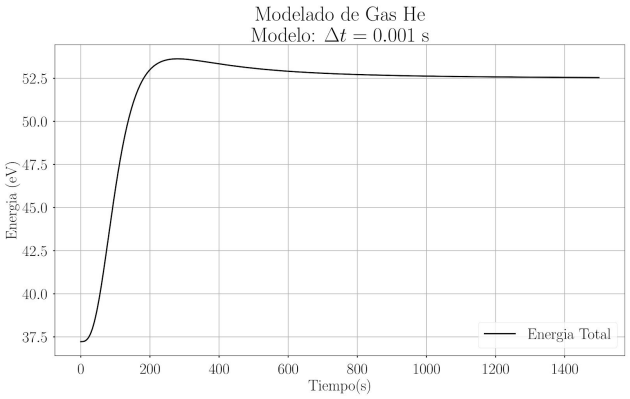
*1. A) Utilizando la última versión del programa desarrollado en clase, calcule las energías cinética, potencial y total como función del tiempo, para tres diferentes valores del paso temporal: Dt = 0.001, 0.0001 y 0.00001. b) Elabore un gráfico que contenga las tres energías como función del tiempo para cada uno de los valores de Dt indicados. c) Elabore un gráfico que contenga la energía total como función del tiempo para cada uno de los Dt indicados (un total de tres gráficos, uno para cada valor de Dt). d) Incorpore los tres primeros gráficos obtenidos en una primera sección de un documento y comente en relación a estos gráficos; incorpore los tres últimos gráficos en una segunda sección de este documento y comente igualmente en relación a estos gráficos.*

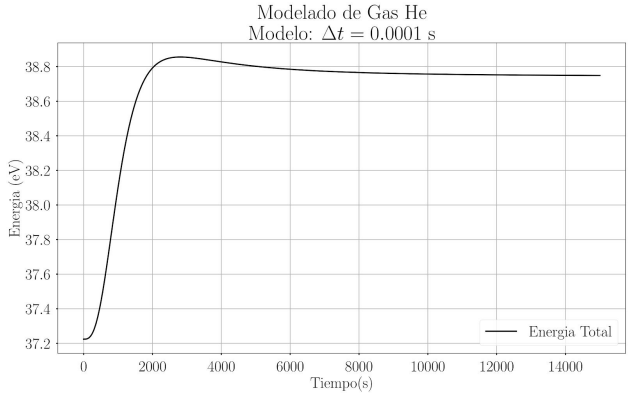
RESULTADOS

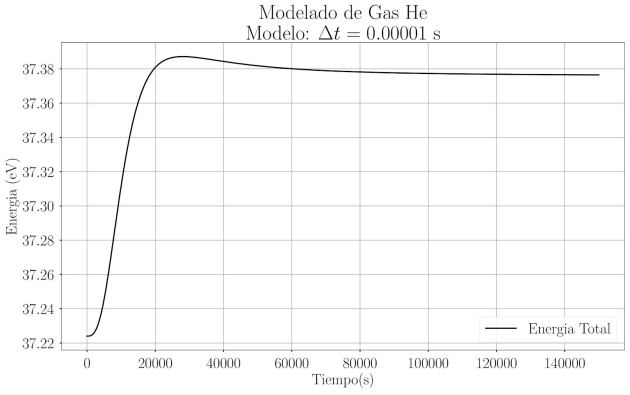


COMENTARIOS

Respecto a las primeras tres gráficas, las cuales muestran una variación en el intervalo de tiempo, con todas las componentes de las energías, es posible apreciar que en el máximo de energía cinética, como el mínimo en energía potencial, el decaimiento se suaviza conforme disminuimos el intervalo del tiempo. En otras palabras, hay un cambio menos abrupto en la caída de la curva de energía cinética y energía potencial al minimizar el intervalo del tiempo en el código. Respecto a los valores en energía cinética y potencial, no hay cambios considerables.

Por otra parte, las últimas tres gráficas muestran la energía total como función del tiempo para las diferentes variaciones en el intervalo del tiempo, mostradas anteriormente. Es posible apreciar que conforme aumenta el intervalo del tiempo, también aumenta el valor máximo que puede tomar la energía total; además, es importante notar que, tanto en el intervalo de tiempo igual a 0.0001 s y igual a 0.00001 s, la curva empieza a decaer aproximadamente en 2000 s y 20000 s, respectivamente y la energía toma valores de 38.8 eV y 37.8 eV, respectivamente. Mientras que para un intervalo de tiempo igual a 0.001, la energía se estabiliza en 52.2 eV cuando el tiempo empieza a tomar valores mayores a 200 s.

*2. Modificar el código.*

El código modificado queda de la siguiente forma:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define N 512
#define L 2*2*10.0
#define NL 8 /*Cubo de N*/

#define Nt 2000
double Dt = 0.001;   /*Tiempo en picosegundos*/

double T = 300.0;
double tao = 100.0;   /*Declaración de constantes*/
double mass = 39.948; /*1.66E-27;
double sigma = (3.4E-10)*(1.0/1.0E-10);
double epsilon = (1.65E-21)*(1.0/1.66E-23);
double rx[N], ry[N], rz[N];
double vx[N], vy[N], vz[N];

double Upot (double r);
void calculaFij (double r, int i, int j, double Fij[], int ipx, int ipy,
int ipz);
void cond_ini(void);

int main(void)
```

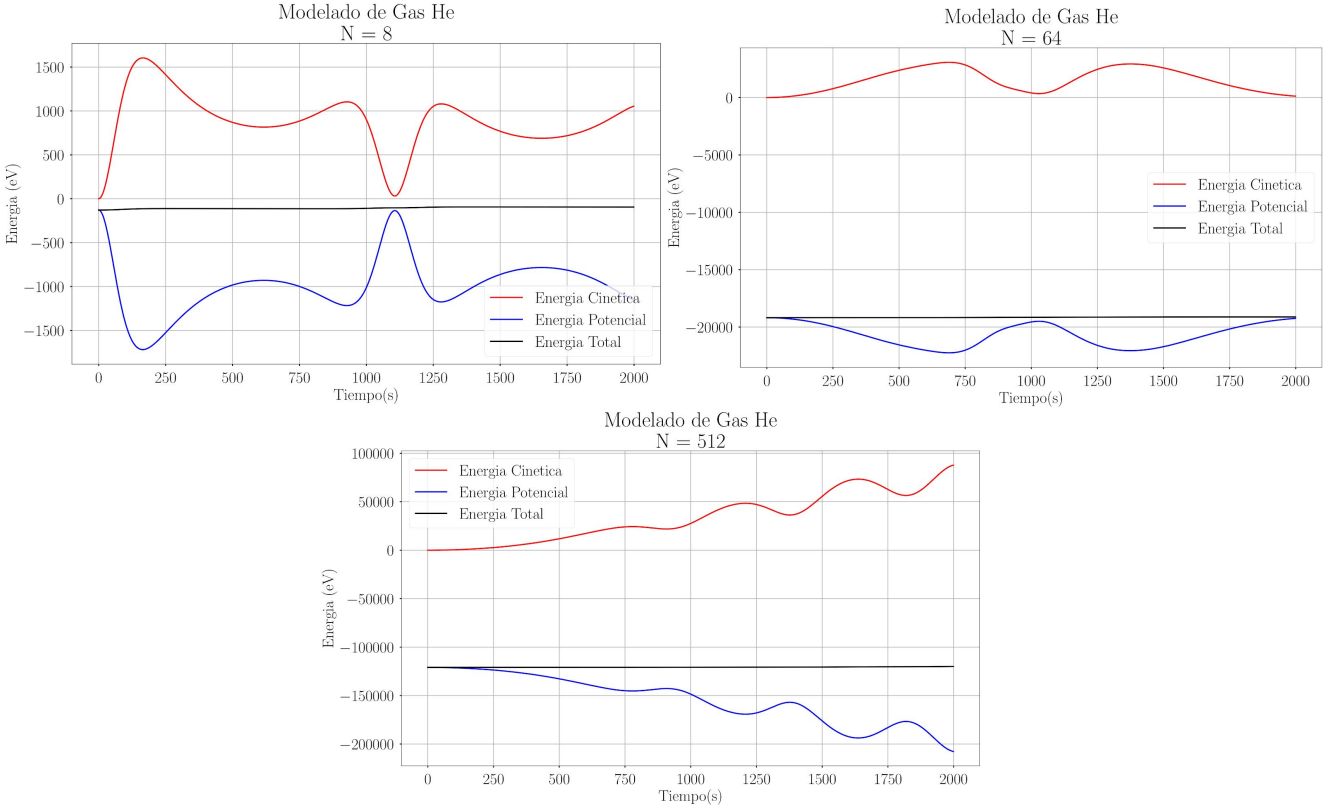
```
{
    int i, j, it, timeprint;
    int ipx, ipy, ipz;
    double Fij[3], Fi[3], Ecin, Ui, Epot;
    double r;
    double newrx[N], newry[N], newrz[N];
    double newvx[N], newvy[N], newvz[N];

    timeprint = 1;
    tao = tao*Dt;

    cond_ini();

    for( it=1; it<Nt; it++)
    {
       Ecin = 0.0;
       Epot = 0.0;

       for (i = 0; i < N; i++)
       {
             /* Calculo de la fuerza sobre la particula i*/
             Fi[0] = 0.0;
             Fi[1] = 0.0;
             Fi[2] = 0.0;

             Ui = 0.0;

       for (ipx = -1; ipx <= 1; ipx++){
             for (ipy = -1; ipy <= 1; ipy++){
                   for (j = 0; j < N; j++){
                         for (j = 0; j < N; j++){
                               if (j != i){
                                     r = (rx[j] + ipx*L - rx[i])*(rx[j] +
ipx*L - rx[i]);
                                      r   =   r   +   (ry[j]   +   ipy*L   -
ry[i])*(ry[j] + ipy*L - ry[i]);
                                      r   =   r   +   (rz[j]   +   ipz*L   -
rz[i])*(rz[j] + ipz*L - rz[i]);
                                     r = sqrt(r);

                                      calculaFij(r,  i,  j,  Fij,  ipx,  ipy,
ipz);

                                     Fi[0] = Fi[0] + Fij[0];
                                     Fi[1] = Fi[1] + Fij[1];
                                     Fi[2] = Fi[2] + Fij[2];
```

```
                                    Ui = Ui + Upot(r);
                              }
                        }
                  }
            }
      }
            /* Calculo de las nuevas velocidades*/
            newvx[i] = vx[i] + Fi[0] / mass * Dt;
            newvy[i] = vy[i] + Fi[1] / mass * Dt;
            newvz[i] = vz[i] + Fi[2] / mass * Dt;


            /*Movimiento Rectilinio uniforme: aceleraci\'on constante*/

                newrx[i]    =    rx[i]    +    vx[i]    *    Dt    +
(1.0/2.0)*(Fi[0]/mass)*(Dt*Dt);
                newry[i]    =    ry[i]    +    vy[i]    *    Dt    +
(1.0/2.0)*(Fi[1]/mass)*(Dt*Dt);
                newrz[i]    =    rz[i]    +    vz[i]    *    Dt    +
(1.0/2.0)*(Fi[2]/mass)*(Dt*Dt);

            if (newrx[i]<0.0) newrx[i] = newrx[i] + L;
            if (newrx[i]> L ) newrx[i] = newrx[i] - L;
            if (newry[i]<0.0) newry[i] = newry[i] + L;
            if (newry[i]> L ) newry[i] = newry[i] - L;
            if (newrz[i]<0.0) newrz[i] = newrz[i] + L;
            if (newrz[i]> L ) newrz[i] = newrz[i] - L;

            /*printf("\n %lf, %lf, %lf", newrx[i], newry[i], newrz[i]);*/

             Ecin = Ecin + 1.0 / 2.0 * mass * (vx[i] * vx[i] + vy[i] *
vy[i] + vz[i] * vz[i]);
            Epot = Epot + Ui/2.0;
      }

      if ((it % timeprint) == 0) printf("%lf %lf %lf\n", Ecin,Epot,Ecin +
Epot);

      for (i = 0; i < N; i++)
      {
            vx[i] = newvx[i];
            vy[i] = newvy[i];
            vz[i] = newvz[i];

            rx[i] = newrx[i];
```

```
            ry[i] = newry[i];
            rz[i] = newrz[i];
       }
    }

    printf("\n");

    return(0);
}


double Upot(double r)
{
    return(4.0 * epsilon * (pow(sigma / r, 12) - pow(sigma / r, 6)));
}

void calculaFij(double r, int i, int j, double Fij[], int ipx, int ipy,
int ipz)
{
    double fij;

     fij = -24.0 * epsilon / (sigma * sigma) * pow(sigma / r, 8) * (2.0 *
pow(sigma / r, 6) - 1.0);
    Fij[0] = fij * (rx[j] + ipx*L - rx[i]);
    Fij[1] = fij * (ry[j] + ipy*L - ry[i]);
    Fij[2] = fij * (rz[j] + ipz*L - rz[i]);

}

void cond_ini(void)
{
    int i, ix, iy, iz;

    for (i = 0; i < N; i++){
       vx[i] = 0.0;
       vy[i] = 0.0;
       vz[i] = 0.0;
    }

    for (iz = 0; iz < NL ; iz++){
       for (iy = 0; iy < NL ; iy++){
            for (ix = 0; ix < NL ; ix++){
                  rx[iz * NL * NL + iy * NL + ix] = L / (NL + 1.0) * (ix
+ 1.0);
                  ry[iz * NL * NL + iy * NL + ix] = L / (NL + 1.0) * (iy
+ 1.0);
```

```
                    rz[iz * NL * NL + iy * NL + ix] = L / (NL + 1.0) * (iz
+ 1.0);
                }
            }
        }
}
```

*3. Utilice el código desarrollado en el ejercicio anterior para elaborar tres corridas para tres diferentes números de partículas: N = 8, 64 y 512. Utilice en los tres casos un valor de Dt = 0.001 y un número de pasos totales de 2000 (no olvide escalar el valor de NL y de L acorde al número de partículas). Elabore tres gráficos, uno para cada valor de N, en los cuales se muestran las tres energías del sistema de partículas (energía cinética, energía potencial y energía total) como función del tiempo.*




COMENTARIOS

Es claro que la estabilidad del programa se empieza a perder conforme se aumenta el número de partículas (incluso cuidando factores como NL y L, del código). Cabe mencionar que para el programa de N = 64 partículas, mi computadora tardó aproximadamente 3 minutos para correr el programa. Mientras que para el programa de N = 512 partículas, el programa tardó más de media hora para completar la corrida. No obstante, vale la pena resaltar que el comportamiento esperado (como se muestra en la tarea), sí se logró para el caso de N = 8 partículas.