WTPC2016: Dinámica

Celleri Humberto, Lugones Rodrigo, Pinto Sebastián

Universidad Nacional de Tucumán
Universidad de Buenos Aires
Universidad Nacional de Quilmes

Contenido

1) Introducción al problema
2) Análisis de estrategias y acuerdo de forma de trabajo.
3) Aplicación de temas curso
4) Inconvenientes destacados y experiencia

Introducción al problema

- ¿Con qué empezamos?

  Programa de **dinámica molecular** escrito completamente en C.

- Problemas:

  Muy difícil de leer y entender.

  Poco amigable para adaptarlo a otras funcionalidades.

- Objetivos:

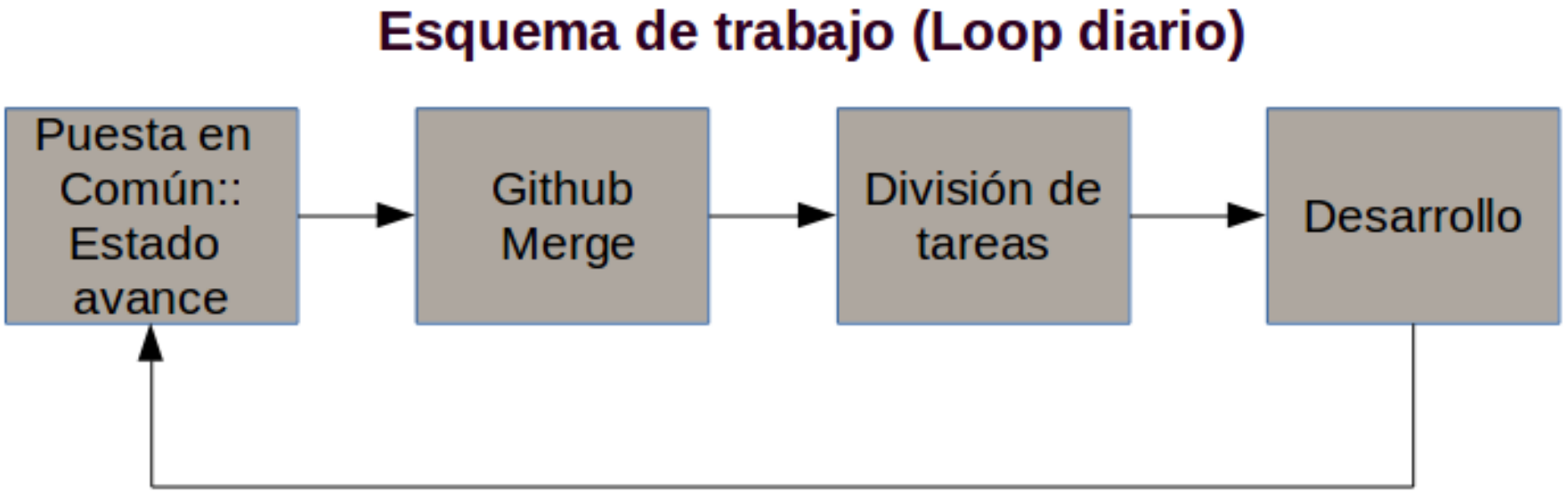
  Hacer un **programa de dinámica molecular que sea flexible**, aprovechando el código desarrollado, de eficiencia probada.

Análisis de estrategias

Como medio de comunicación constante se emplearon:

1) Grupo de Whatsapp para consulta.
2) Aplicación de manejo de proyectos (Trello).
3) Reuniones diarias de discusión y puesta en común.

Análisis de estrategias



Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicación: Github

Todas los archivos fueron subidos al repositorio de **github**, el cual:

1) permitió **trabajar de manera colaborativa** entre los integrantes,
2) y **evitar problemas de versionado**:

https://github.com/HumbertoCelleri/ljmd

Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicación: Python

Wrapper C/Python: Ctypes!

Estructura en C

```c
/* structure to hold the complete information
 * about the MD system */
struct _mdsys {

    double dt, mass, epsilon, sigma, box, rcut;

    ...

};
typedef struct _mdsys mdsys_t;
```

Clase en Python

```python
class mdsys_t(C.Structure):
    _fields_ = [ ('dt', C.c_double),
                 ('mass', C.c_double),
                 ('epsilon', C.c_double),
                 ('sigma', C.c_double),
                 ('box', C.c_double),
                 ('rcut', C.c_double),

                          ...
```

Con muchos mas atributos, por supuesto…

Más métodos...

Aplicación: Python

Funciones del código C que nos interesan:

```c
/* build and update cell list */
static void updcells(mdsys_t *sys)

/* compute forces */
static void force(mdsys_t *sys)

/* velocity verlet */
static void velverlet(mdsys_t *sys)

/* compute kinetic energy */
static void ekin(mdsys_t *sys)
```

Libreria dinámica: libc.so

Dentro de Python:
Ej: Medidor

```python
CLIB = C.CDLL('./libc.so')

class Medidor(object):

    def kinetic_energy(self, sys):

        self.CLIB.ekin(C.byref(sys))

        return sys.ekin
```

Llamado a la función escrita en C

Aplicación: Python

En nuestro código Python vamos a ver cosas como:

```
# Creo e inicializo mi sistema con determinados parametros

system = mdsys.mdsys_t()
system.input(parameters)

# Inicializo el medidor
med = medidor.Medidor()

for i in range(0, nsteps):

        # Metodo de evolucion del sistema
        sistem.evolution()

        if i % nprint == 0:

            # Cada nprint pasos el medidor mide observables del sistema
            ekin = med.kinetic_energy(system)
            epot = med.potencial_energy(system)
            etot = ekin + epot
            temp = med.temperature(system)

            print i, temp, ekin, epot, etot
```

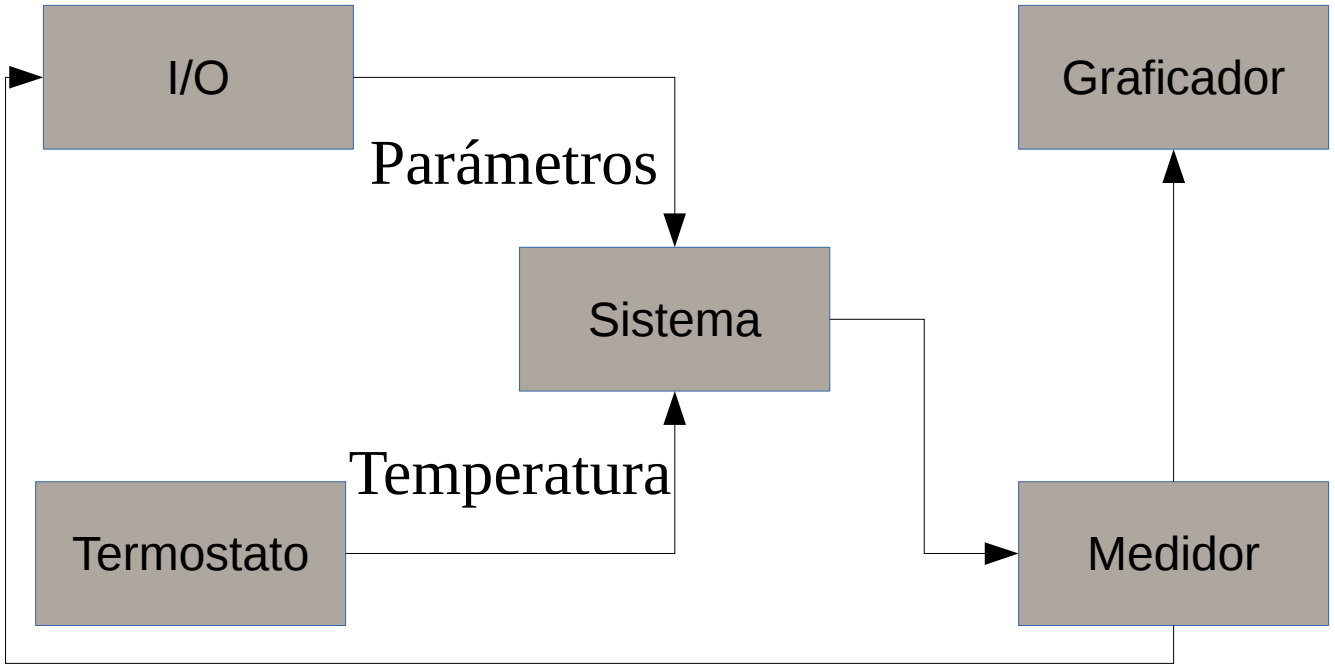
Aplicación: Python

Con este nuevo esquema de trabajo se **pudo implementar** de forma inmediata:

- Un **nuevo potencial de interacción** entre partículas
- Un **termostato**

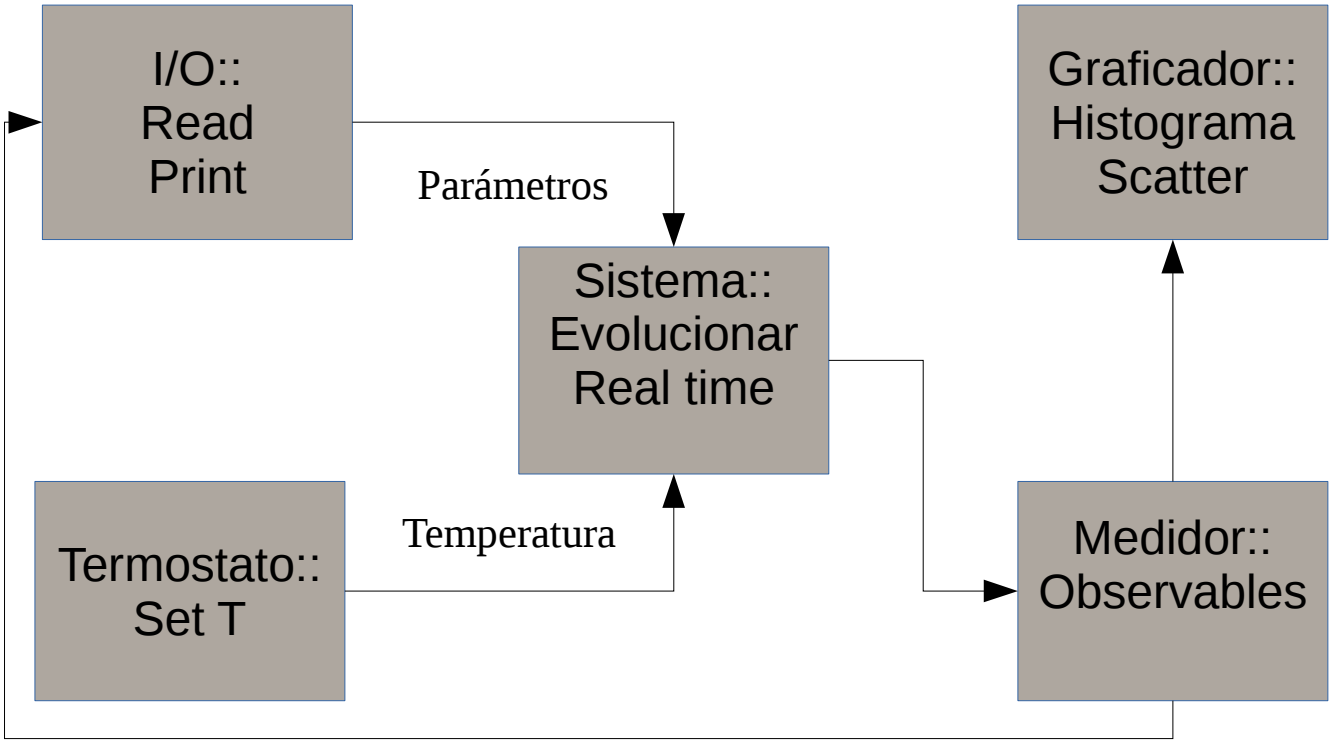
Aplicación: Python OOP

Estructura



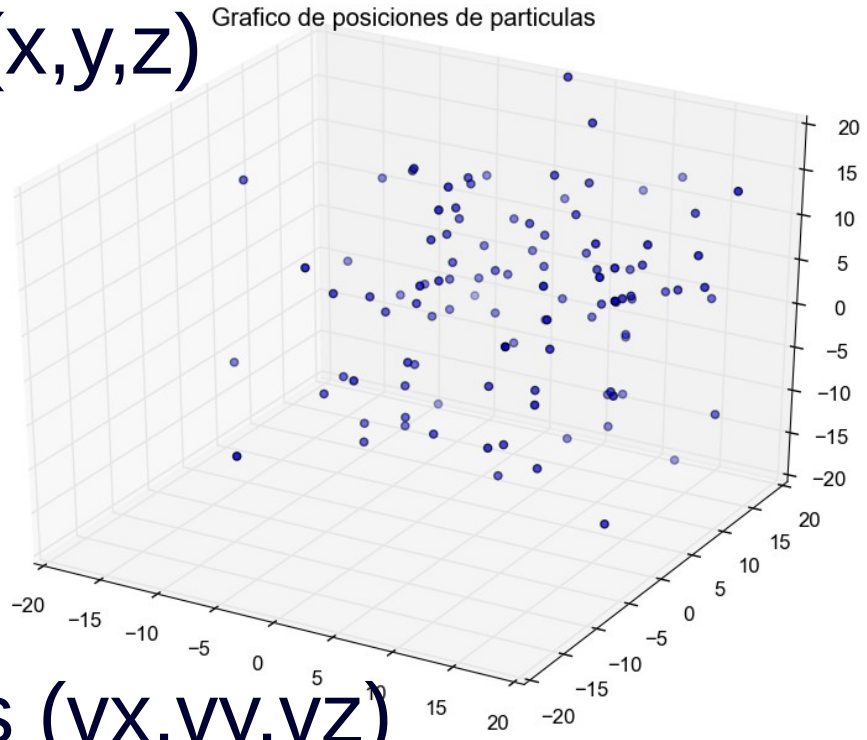
Aplicación: Python OOP

Estructura Extendida

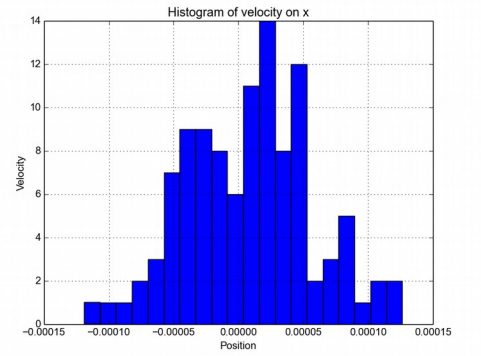


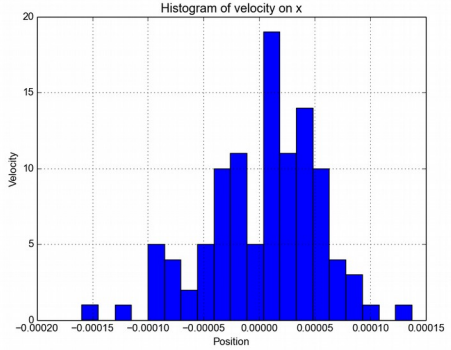
Resultados: Matplotlib

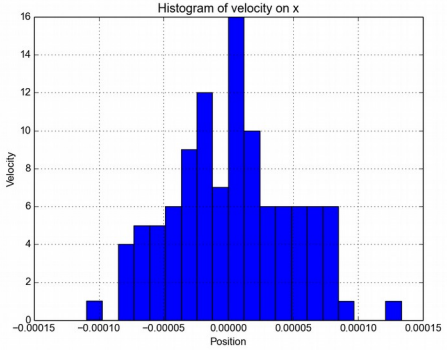
1) Distribución de posiciones(x,y,z)



2) Histograma de velocidades (vx,vy,vz)







Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicación: Debbuging

Casos de interés:

Aplicación: Debbuging

Casos de interés:

1) Static function in C code: Al compilar: ERROR

File "/home/wtpc-16/Documentos/ljmd/src/PYTHON/medidor.py", line 22, in potencial_energy

CLIB.force(C.byref(sys))

File "/usr/lib/python2.7/ctypes/__init__.py", line 378, in __getattr__

func = self.__getitem__(name)

File "/usr/lib/python2.7/ctypes/__init__.py", line 383, in __getitem__

func = self._FuncPtr((name_or_ordinal, self))

AttributeError: ./libc.so: undefined symbol: force

Aplicación: Debbuging

Casos de interés:

1) Static function in C code: nm libc.so

```
                 U fgets@@GLIBC_2.2.5
0000000000003a54 T _fini
                 U floor
                 U fopen@@GLIBC_2.2.5
0000000000001d7c t force
                 U fprintf@@GLIBC_2.2.5
0000000000000da0 t frame_dummy
```

Aplicación: Debbuging

Casos de interés:

1) Static function in C code
2) GitHub: New Code Conflict when Merge:

<<<<<<<<<<<<<<< User1

New Code

==========

Old Code

>>>>>>>>>>>>>>>>User2

Aplicación: Debbuging

Casos de interés:

1) Static function in C code
2) GitHub: New Code Conflict when Merge:

<<<<<<<<<<<<<<< User1

New Code

==========

Old Code

>>>>>>>>>>>>>>>>User2

Permite solucionarse editando el documento y corrigiendo las inconsistencias.

Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicacion: Profiling

Se realizó un **estudio comparativo** de tiempo de ejecución de:

1) Código en C con OpenMP
2) Python + Wrapper(C) + OpenMP

Aplicacion: Profiling

Se realizó un estudio comparativo de tiempo de ejecución del código en C y en Python + Wrapper(C)

1) C: perf stat
2) Python:

   tic-toc, cProfiler

Sistema:

7000 partículas

30 pasos

Intel Xeon E5-2665 x2, 32Gb Ram, Ubuntu 14.04.3 (64bits), gcc 4.8.4, Python 2.7.

Aplicacion: Profiling

Se realizó un estudio comparativo de tiempo de ejecución del código en C y en Python + Wrapper(C)

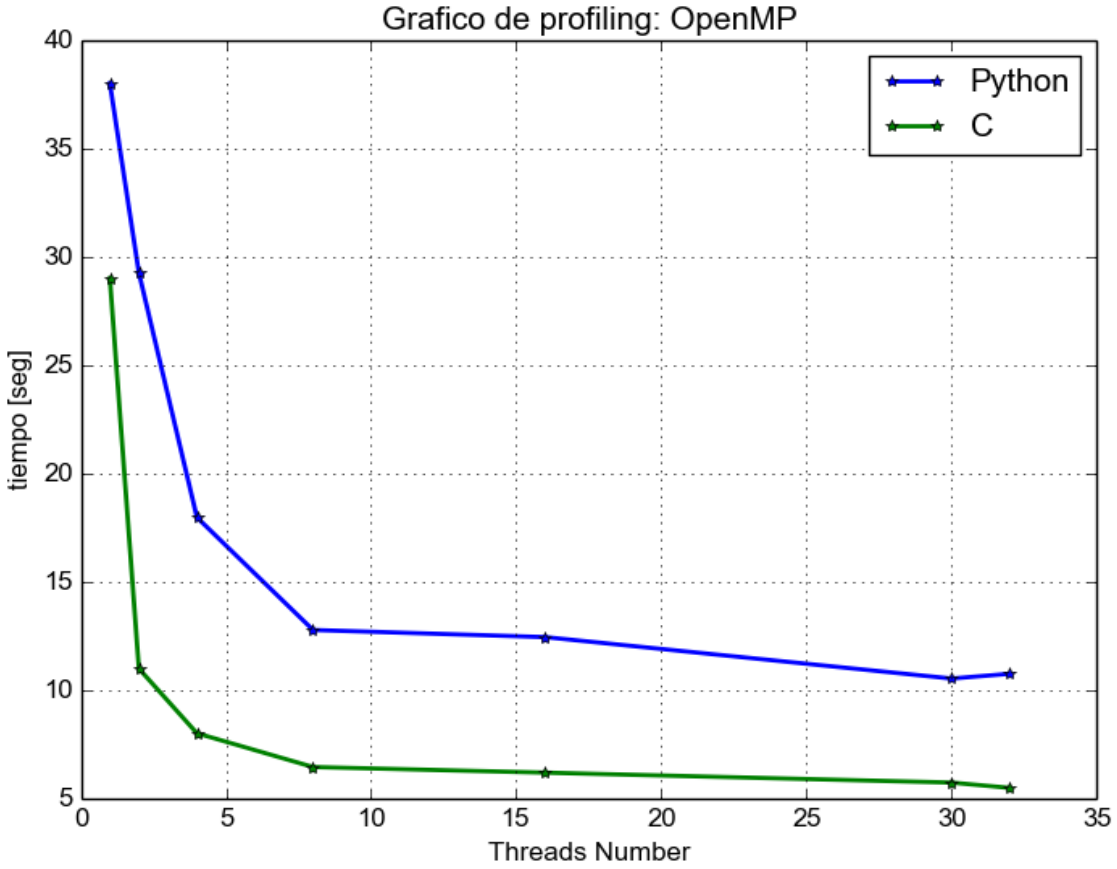
1) C: perf stat
2) Python:

   tic-toc, cProfiler

<u>Sistema</u>:

7000 partículas

30 pasos



Intel Xeon E5-2665 x2, 32Gb Ram, Ubuntu 14.04.3 (64bits), gcc 4.8.4, Python 2.7.

Aplicación de Contenidos

1) Github como plataforma
2) Python
    1) Librerías
        1) Linkeo de python con C
        2) Matplotlib
    2) Programación orientada a objetos
3) Debbuging
4) Profiling
5) Documentación

Aplicación: Documentación

1) Se utilizó **doxygen**
   1) Permite utilizar varios lenguajes (C,Python)
   2) Es relativamente sencillo y automático (Si se conoce la sintaxis)

      # Genera documento de configuración.

      $ doxygen -g <config.file>

      El cual se puede moficar a gusto, agregando archivos *.doc con descripción.
   3) Posibilidad de obtener diferentes formatos salida (html,latex).

Aplicación: Documentación

1) Se utilizó **doxygen**

2) Mostrar **documentación en html**

Destacados y experiencias

Trabajo Colaborativo

+

Separación de Tareas

Conclusiones

1) Se **cumplieron los objetivos** principales:
    1) Aplicando lo visto en el workshop.
    2) Contra adversidades y abandono.

Conclusiones

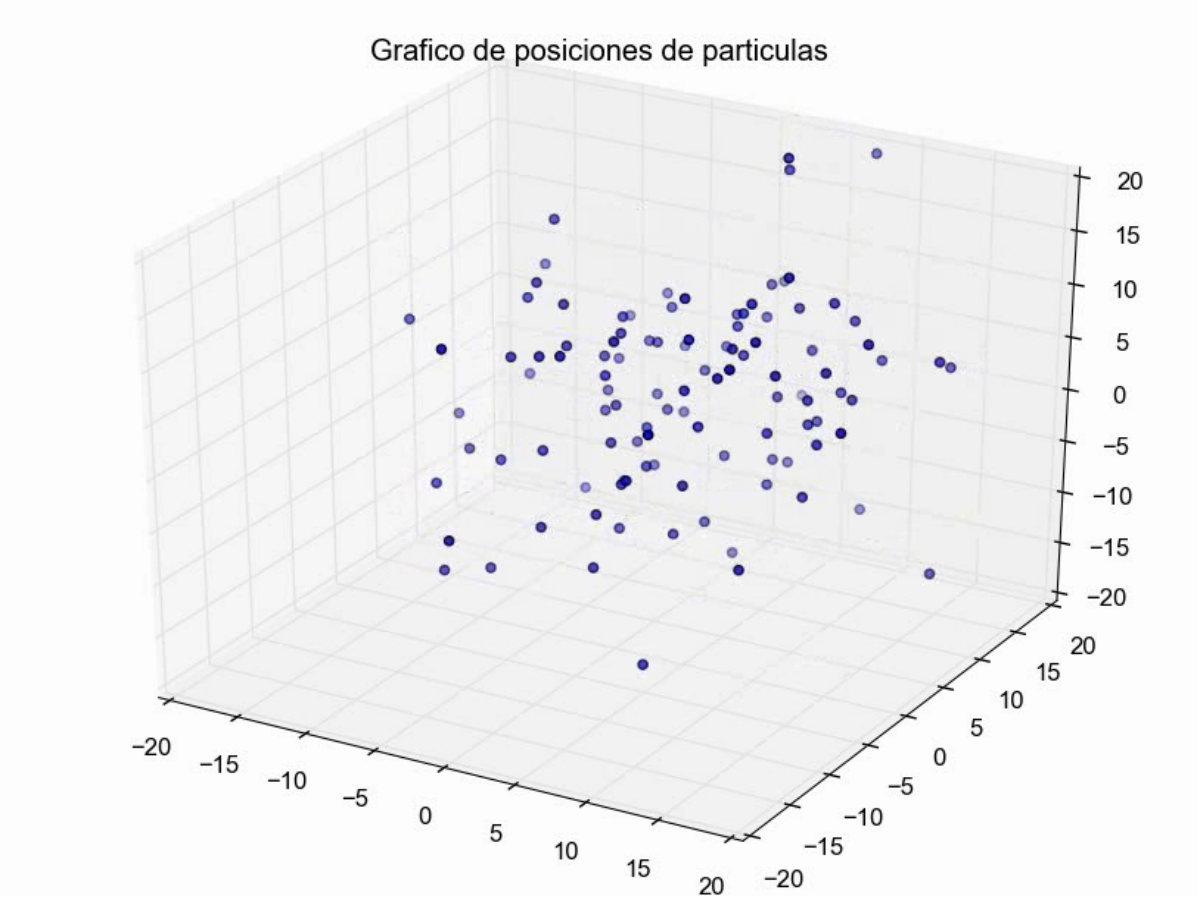
1) Se **cumplieron los objetivos** principales:
    1) Aplicando lo visto en el workshop
    2) Contra adversidades y abandono**s** (Reiterados).

Conclusiones

1) Se cumplieron los objetivos principales:
    1) Aplicando lo visto en el workshop
    2) Contra adversidades y abandonos (Reiterados).
2) Esto **incluye**:
    1) La **aplicación de ctypes** para wrappear Python y C.
    2) Un **nuevo potencial** de interacción de partículas
    3) La implementación de un **termostato**.

WTPC
workshop en técnicas de programación científica

Muchas Gracias!!



Rodrigo Lugones

Pinto Sebastián

Celleri Humberto

WTPC
workshop en técnicas de programación científica

Muchas Gracias!!

Rodrigo Lugones, Pinto Sebastián, Celleri Humberto

Pablo Alcain's (Dream) Team