

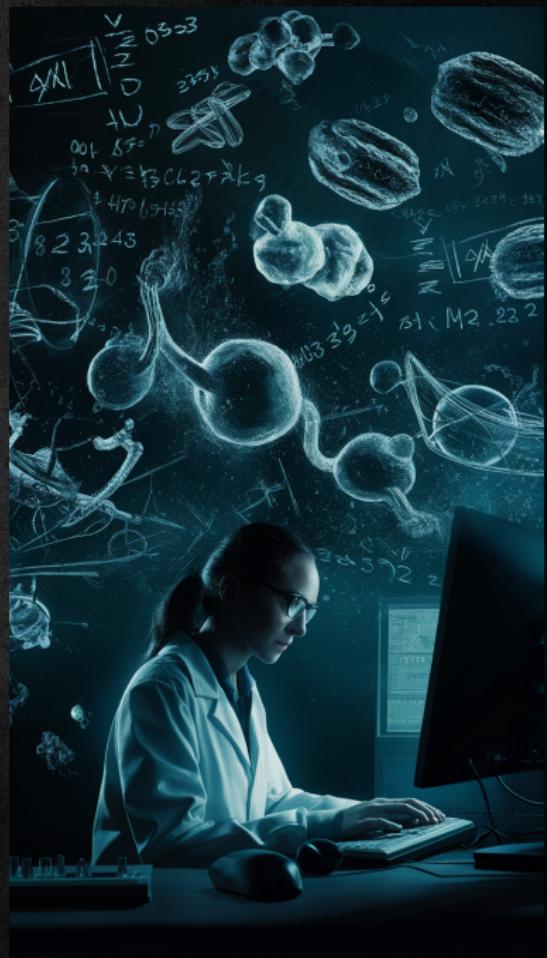
Métodos Computacionales

Agosto 5, 2024

S. Alexis Paz



Departamento de
QUÍMICA TEÓRICA
Y COMPUTACIONAL
Facultad de Ciencias Químicas
Universidad Nacional de Córdoba



hecho con idiogram

Sobre la materia . . .

19 teóricos (24hs) / 9 prácticos (36h)

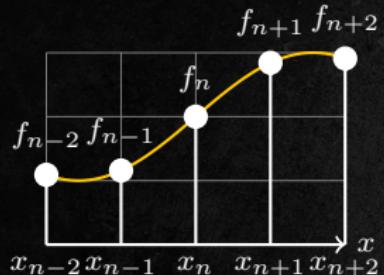
Objetivos: Proveer una **experiencia directa en el uso de computadoras para resolver modelos de sistemas físicos y químicos**. De este modo, se incluye en el mismo el mínimo de técnicas numéricas que son necesarias para la realización de física y química computacional. Si bien la perspectiva del curso es físico-matemática, la creciente difusión de la computación en todas las áreas de la química hará que el mismo redunde en beneficio de las diferentes ramas de la química que se desarrollan en nuestra Facultad. Se provee además una introducción a técnicas de simulación de Dinámica Molecular y Electrónica, Monte Carlo y Cálculos de Alto Rendimiento.

Contenidos: Operaciones matemáticas básicas. Métodos de diferenciación numérica. Cuadratura numérica. Búsqueda de raíces. Extremos de funciones. Ecuaciones diferenciales ordinarias. Métodos simples. Métodos de pasos múltiples e implícitos. Introducción a la dinámica atómica y molecular. Generalidades sobre las simulaciones. Modelos de sistemas y potenciales de interacción. Sistemas atómicos de muchas partículas. Dinámica molecular. Estructura básica de un programa de simulación. Obtención de valores medios. Métodos con valores en el contorno y problemas de autovalores. Métodos de Monte Carlo. La estrategia básica de Monte Carlo.

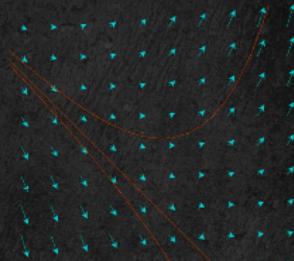
Programa

- Tema 1:** Operaciones matemáticas básicas. Métodos de diferenciación numérica. Cuadratura numérica. Búsqueda de raíces. Extremos de funciones.
- Tema 2:** Ecuaciones diferenciales ordinarias. Métodos simples. Métodos de pasos múltiples e implícitos. Métodos de Runge-Kutta.
- Tema 3:** Métodos con valores en el contorno y problemas de autovalores. El algoritmo de Numerov. Integración directa del problema con valores en el contorno. Solución de la ecuación de Schrödinger para un potencial monodimensional.
- Tema 4:** Introducción Mecánica Estadística, Métodos de Monte Carlo. La estrategia básica de Monte Carlo. Generando variables aleatorias con una distribución especificada. El algoritmo de Metrópolis. El modelo de Ising.
- Tema 5:** Introducción a la dinámica molecular. Generalidades sobre las simulaciones. Modelos de sistemas y potenciales de interacción. Estructura básica de un programa de dinámica molecular. Algoritmos para integrar las ecuaciones de movimiento. Obtención de valores medios.

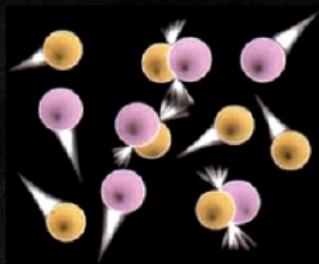
Op. Mat. Basicas



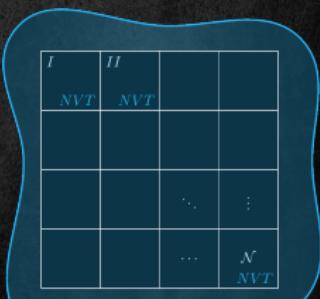
E.D.O.



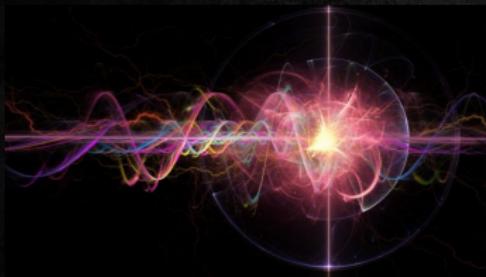
Dinámica Molecular



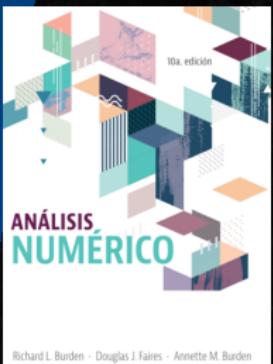
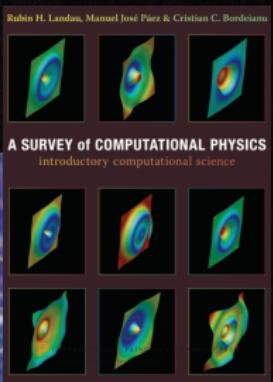
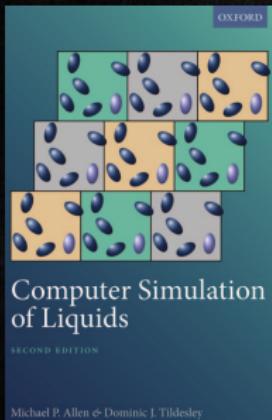
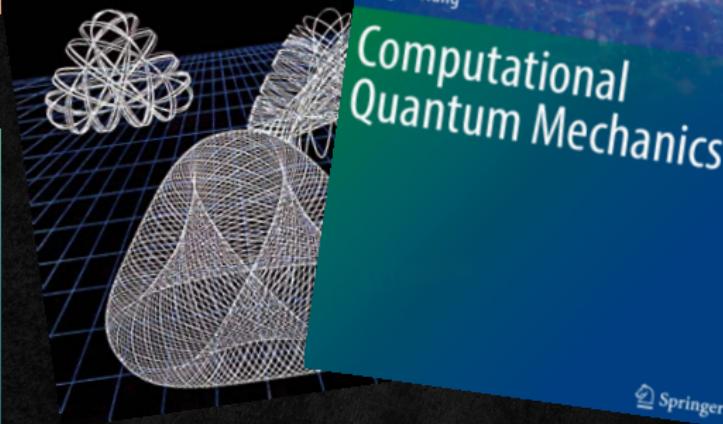
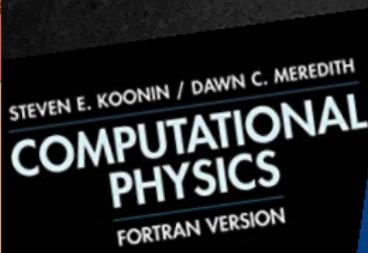
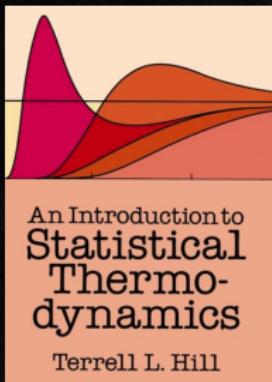
Montecarlo



Cuántica



Bibliografía



Richard L. Burden · Douglas J. Faires · Annette M. Burden

Cronograma

TEÓRICOS	PRÁCTICOS
Lunes Y Miércoles 8:15 a 9:30 Aula DQTC	Jueves 9:00 a 13:00 Lab 209 – Edificio Integrador
05/08 Introducción a la materia	08/08 TP0: Adecuación del entorno virtual
07/08 Operaciones Matemáticas Básicas: raíces	
12/08 Operaciones Matemáticas Básicas: derivada	15/08 TP1: Introducción al lenguaje de programación
14/08 Operaciones Matemáticas Básicas: cuadratura	
19/08 Operaciones Matemáticas Básicas: simplex	22/08
21/08 Ecuaciones Diferenciales Ordinarias	
26/08 Ecuaciones Diferenciales Ordinarias(2)	TP2: Operaciones Matemáticas Básicas
28/08 Métodos con valores en el contorno	29/08
02/09 Métodos con valores en el contorno(2)	
04/09 Métodos con valores en el contorno(3)	05/09 TP3: Ecuaciones Diferenciales Ordinarias
09/09 Problemas de autovalores	12/09 TP4: Autovalores y dinámica cuántica.
11/09 Problemas de autovalores(2)	
16/09 Introducción Mecánica Estadística	19/09 Consulta 1er Parcial
18/09 Consulta 1er Parcial	
1er Parcial	
07/10 Introducción Mecánica Estadística(2)	10/10 TP4: Autovalores y dinámica cuántica.(2)
09/10 Introducción Mecánica Estadística(3)	
14/10 Introducción Mecánica Estadística(4)	17/10 TP5: Generación de Números Aleatorios
16/10 Métodos de Monte Carlo	
21/10 Métodos de Monte Carlo(2)	24/10 TP6: Modelo de Ising
23/10 Métodos de Monte Carlo(3)	
28/10 Introducción a la dinámica molecular	31/10
30/10 Introducción a la dinámica molecular(2)	
04/11 Introducción a la dinámica molecular(3)	TP7: Dinámica Molecular
06/11 Aplicaciones	07/11
11/11 Aplicaciones(2)	
13/11 Consulta 2dº Parcial	14/09 Consulta 1er Parcial
2do Parcial	



Fortran



Puedo reventar la
computadora en 10 líneas.

Python



Necesito una libreria
para hacer $2+2$.

Fortran

```
program ejemplo
implicit none
integer,parameter :: dp=8
real(dp)      :: l(3),r(3),p(3)

!initialize arrays
r(:)=[1._dp, 2._dp, 0._dp]
p(:)=[.4_dp, 1.e-3_dp,3.2_dp]

!compute cross product
l(:)=cross_product(r,p)

print *, l(:)

contains

function cross_product(a,b) result(c)
  real(dp)      :: c(3)
  real(dp),intent(in) :: a(3),b(3)
  c(1)=a(2)*b(3)-a(3)*b(2)
  c(2)=a(3)*b(1)-a(1)*b(3)
  c(3)=a(1)*b(2)-a(2)*b(1)
end function cross_product

end program ejemplo
```

Python

```
#!/usr/bin/env python3
def cross_product(a,b):
    c=[0, 0, 0]
    c[0]=a[1]*b[2]-a[2]*b[1]
    c[1]=a[2]*b[0]-a[0]*b[2]
    c[2]=a[0]*b[1]-a[1]*b[0]
    return c

#initialize arrays
r=[1., 2., 0.]
p=[.4, 1.e-3,3.2]

#compute cross product
l=cross_product(r,p)

print(l)
```

Compilo



```
$ gfortran rxp.f90
```

Ejecuto → \$ time for i in \$(seq 100); do ./a.out >/dev/null; done

```
real    0m0.485s
user    0m0.113s
sys     0m0.390s
```

Ejecuto
s/compilar → \$ time for i in \$(seq 100); do python rxp.py >/dev/null; done

```
real    0m1.701s
user    0m1.378s
sys     0m0.324s
```

Fortran

```
program ejemplo
implicit none
integer,parameter :: dp=8
real(dp)
!initialize
r(:)=[1._d0, 2._d0, 0._d0]
p(:)=[.4_d0, 1.e-3_d0, 3.2_d0]
!compute cross product
l(:)=cross_product(r,p)
print *, l
contains
function cross_product(a,b)
    real(dp), intent(in) :: a(3), b(3)
    real(dp) :: c(3)
    c(1)=a(2)*b(3)-a(3)*b(2)
    c(2)=a(3)*b(1)-a(1)*b(3)
    c(3)=a(1)*b(2)-a(2)*b(1)
    return c
end function
end program ejemplo
```

Python

```
#!/usr/bin/env python3
def cross_product(a,b):
    c=[0, 0, 0]
    c[0]=a[1]*b[2]-a[2]*b[1]
    c[1]=a[2]*b[0]-a[0]*b[2]
    c[2]=a[0]*b[1]-a[1]*b[0]
    return c

print(*, l)

#contains
#initialize arrays
r=[1., 2. ,0.]
p=[.4, 1.e-3,3.2]

#function cross_product
#compute cross product
l=np.cross(r, p)

print(l)
```

NumPy

```
#!/usr/bin/env python3
import numpy as np

#initialize arrays
r = np.array([1., 2. ,0.])
p = np.array([.4, 1.e-3,3.2])

#compute cross product
l = np.cross(r, p)

print(l)
```

Fortran

```
program ejemplo
implicit none
integer,parameter :: dp=8
real(dp)
!initialize
r(:)=[1._d0, 2._d0, 0._d0]
p(:)=[.4_d0, 1.e-3_d0, 3.2_d0]
!compute cross product
l(:)=cross_product(r,p)
print *, l
contains
function cross_product(a,b)
    real(dp)
    real(dp)
    c(1)=a(2)*b(3)-a(3)*b(2)
    c(2)=a(3)*b(1)-a(1)*b(3)
    c(3)=a(1)*b(2)-a(2)*b(1)
    print(l)
end function
end program ejemplo
```

Python

```
#!/usr/bin/env python3
def cross_product(a,b):
    c=[0, 0, 0]
    c[0]=a[1]*b[2]-a[2]*b[1]
    c[1]=a[2]*b[0]-a[0]*b[2]
    c[2]=a[0]*b[1]-a[1]*b[0]
    return c
```

```
#initialize arrays
r=[1., 2. ,0.]
p=[.4, 1.e-3,3.2]
```

```
#compute cross product
l=cross_product(r,p)
```

```
c(1)=a(2)
c(2)=a(3)
c(3)=a(1)
```

NumPy

```
#!/usr/bin/env python3
import numpy as np

#initialize arrays
r = np.array([1., 2. ,0.])
p = np.array([.4, 1.e-3,3.2])

#compute cross product
l = np.cross(r, p)

print(l)
```

	real	0m8.124s
	user	0m6.690s
	sys	0m1.418s

end program ejemplo

Fortran

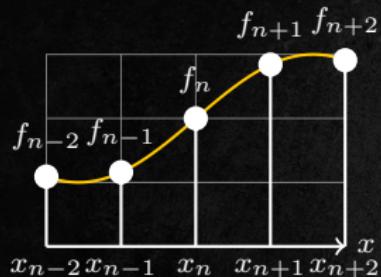


Python

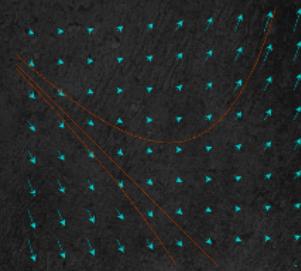


- 1957 (2018).
- Compilado.
- Computación de Alto Rendimiento.
- Fuente de Numpy, Scipy, LAPACK, etc.
- 1991.
- Interpretado.
- Legible, simple.
- Universo de librerías.

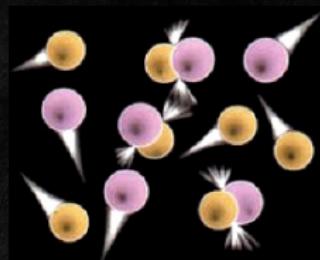
Op. Mat. Basicas



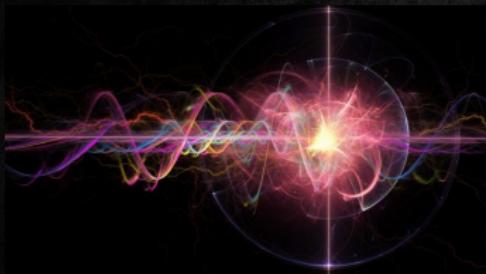
E.D.O.



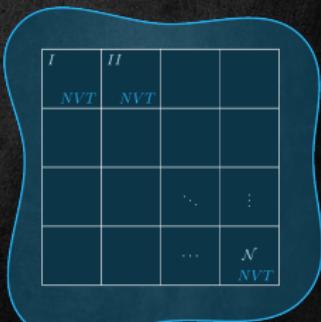
Dinámica Molecular



Cuántica



Montecarlo



Aceleración

