Cálculo con software libre: Octave y SageMath

Francisco Pena

Departamento de Matemática Aplicada. Universidade de Santiago de Compostela

Software Libre Científico. Edición 2019

Índice I

Cálculo simbólico vs. numérico

- El cálculo simbólico trabaja con *variables simbólicas* y *expresiones simbólicas*.

$$\int x^2 \, \mathrm{d}x = \frac{x^3}{3} + c.$$

- El cálculo numérico trabaja con *números máquina* y *algoritmos*.

$$\int_0^1 x^2 \, \mathrm{d}x = 0.333...3 \text{ (16 dígitos)}.$$

Lenguajes de cálculo científico

- Cálculo simbólico, principalmente:
  - Mathematica
  - Maple
  - maxima
  - sympy (Python)
  - SageMath
- Cálculo numérico, principalmente:
  - Matlab
  - numpy (Python)
  - Octave

Motivación

- Motivación para Octave:
  - Clon de Matlab.
  - *Kernel* vs *toolboxes*, como `symbolic`.
  - Paquetes disponibles.
- Motivación para SageMath:
  - Python para cálculo científico, sin `imports`.
  - Aplicaciones resumidas.

Motivación II

- Why isn't Sage just part of Octave? (12 Dic 2007)

  Octave doesn't do anything at all that is useful for any of my personal research, at least the research I was doing when I started Sage. **I am a number theorist / arithmetic geometer, who works with elliptic curves, modular forms, prime numbers, arbitrary precision arithmetic, and high precision numerical computation**. *There is almost zero overlap between anything I personally wanted to compute and anything Octave does, or even supports at a basic level. That's my personal reason.*

  I did not get the impression that the Matlab language was nearly so beautiful, well designed, expressive, etc., as Python, especially **when one is interested in writing code to do a huge range of things – networking, web servers, GUI's, symbolic algebra, etc.**, *none of which is numerical computation.*

Ejecución en local

- Interfaz gráfica de Octave:
  - Ventana de comandos (ejecución uno a uno).
  - Editor integrado para .m

  ✓Orden de ejecución clara.

- Interfaz gráfica de SageMath:
  - Navegador
  - Fichero Jupyter .ipynb (celdas).
  - Editor de texto para .sage

  ✓Mini-script en cada celda.

Ejecución en la nube

- COCALC
- Para Octave:
  1. Abrir una terminal (.term) y editar ficheros (.m)
  2. Abrir una interfaz gráfica (.x11)
- Para Octave:
  - Abrir un fichero Jupyter (.ipynb) y editar ficheros (.sage)

Comparativas

- Python (numpy) vs Matlab vs Octave.
- Python vs lenguajes compilados: x70.
- SageMath usa Cython.
  - Rendimiento de Cython (cdef).
- Ojo con sage.rings.
  - RDF usa numpy.
  - QQ $\rightarrow$ x4 RR $\rightarrow$ x1000 RDF

Arrays

```
u = 1:10
v = u.^2
norm(v)
f = @(x) sin(x.^2)
f(v)
```

```
u = [1,2..10.] #u = range(1,11)
v = [u[i]**2 for i in range(len(u))]
norm(v) #error!
norm(vector(v))
f(x) = sin(x^2)
vector(v).apply_map(f) # vector(map(f, v))
```

Arrays II

```
norm(u(2:end-1)./v(2:end-1))
```

```
norm(vector(RDF,[u[i]/v[i] for i in range(1,len(v)-1)]))
```

Gráficas, simbólico

```
load pkg symbolic
syms x
f = cos(x)
ezplot(f, [1,2])
```

```
f(x) = cos(x)
plot(f, 1,2)
```

Gráficas, numérico

```
t = 1:0.05:3
plot(t, cos(t.^2))
```

```
t = [1,1.05..3]
line(zip(t, map(cos(x^2),t)))
```

Gráficas 3D

```
[R,T] = meshgrid(0:0.1:1.5, 0:pi/10:2*pi)
X = R.*cos(T)
Y = R.*sin(T)
surf(X,Y,sin(X.^2+Y.^2))
```

```
var('r t')
T = Cylindrical('height', ['radius', 'azimuth'])
plot3d(sin(r^2), (r, 0, 1.5), (t, 0, 2*pi), transformation=T)
```

E.D.O. en Octave

- `lsode`.
- Basado en ODEPACK.
- Matlab compatible solvers.
  - `ode23`, third-order Bogacki-Shampine method, adapts the local step size.
  - `ode45`, high-order, variable-step Dormand-Prince method.
  - `ode15s`, variable order method based on Backward Difference Formulas (BDF).
  - `ode15i`, ídem `ode15s` for fully-implicit ODEs.
- `dsolve`, resolvedor simbólico, antes con GiNaC, ahora SymPy.

WARNING: *As of SymPy 0.7.6 (May 2015), there are many problems with systems, even very simple ones. Use these at your own risk, or even better: help us fix SymPy.*

E.D.O. en Octave II

```
function xdot = f (x, t)
  xdot = zeros (3,1);
   xdot(1) = 77.27*(x(2) - x(1)*x(2) + x(1) - 8.375e-06*x(1)^2);
   xdot(2) = (x(3) - x(1)*x(2) - x(2)) / 77.27;
   xdot(3) = 0.161*(x(1) - x(3));
end
t = linspace (0, 500, 1000);
x0 = [ 4; 1.1; 4 ]
y = lsode ("f", x0, t);
plot(t,y)
```

E.D.O. en SageMath

- Resolvedores simbólicos y numéricos
- `desolve`, vía Maxima.
- `desolve_odeint`, vía scipy.integrate (Python).

Bibliografía

- Sage Reference v8.7 [en línea]. *Sage Documentation* [fecha de consulta: 23 abril 2019]. Disponible en: https://doc.sagemath.org/html/en/reference/index.html
- Cálculo numérico con Sage [en línea]. *O blog da Materia "Fundamentos de Matemáticas"* [fecha de consulta: 23 abril 2019]. Disponible en: https://fundmat.wordpress.com/temas-de-fundamentos-de-matematicas/calculo-numerico-con-sage
- *Octave Forge* [en línea] [fecha de consulta: 23 abril 2019]. Disponible en: https://octave.sourceforge.io
- Introducción al Octave [en línea]. *SlideShare*, 2010 [fecha de consulta: 23 abril 2019]. Disponible en: https://es.slideshare.net/franpenabra/introccpresentacion-octave