Búsqueda de raíces y resolución de sistemas lineales

Padula, Nicolás, De Filippis, Darío

{dariodefilippis, nicolasvpadula}@gmail.com

UTN-FRSF - Matemática Superior - Trabajo Práctico N°3

6 de Noviembre de 2016

Índice

Introducción

Los métodos numéricos facilitan la resolución de diversos problemas en el área de las ciencias e ingeniería. Particularmente, respecto al problema de búsqueda de raíces de ecuaciones no lineales, los métodos de Bisección y Newton-Raphson se consolidan como dos de las alternativas más populares, por simplicidad y rendimiento respectivamente. El trabajo contrasta el rendimiento de los ya mencionados métodos, junto con enfoques híbridos, para la función:

$$f = 10\pi \frac{(t-\pi)}{((t-\pi)^2+4)((t-\pi)^2+1)} - \frac{\pi}{10}\arctan(-t)$$

Por otro lado, en la segunda sección, se compara el rendimiento y precisión de métodos directos para la resolución de sistemas de ecuaciones lineales (Reducción Gaussiana con simple y doble precisión) y dos implementaciones distintas de métodos iterativos (Gauss-Seidel) para la resolución del sistema lineal:

$$\begin{bmatrix} 1 & 3/2^2 & 4/3^2 & 5/4^2 & .... \\ 3/2^2 & 1 & 3/2^2 & 4/3^2 & .... \\ 4/3^2 & 3/2^2 & 1 & 3/2^2 & .... \\ 5/4^2 & 4/3^2 & 3/2^2 & 1 & .... \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \end{bmatrix}$$

Con 1<= n <= 500.

Metodología

1. Búsqueda de raíces (ej1.py)

Inicialmente, mediante el uso de Wolfram-Alpha, se aisló la raíz de manera analítica, obteniendo la derivada primera y segunda de la función y los puntos críticos. Evaluando los signos de la función, se determinó que la raíz pertenece al intervalo [2.366839542192523363221 ; 3.912639867208552436782492507708752127339].

Una vez aislada la raíz se procedió a la implementación de los métodos requeridos.

a) Se implementó el método de Newton-Raphson, eligiendo como aproximación inicial x0 = 3.1. Este valor se obtuvo de la disminución manual del intervalo previamente definido. De manera que alguno de los dos extremos satisfaga la condición de convergencia del método (x0 / f(x0)f''(x0) > 0). Una vez garantizada la convergencia del método se corrieron pruebas. La razón por la que el único punto donde se puede garantizar la convergencia del método es tan cercano a la raíz es el comportamiento de f(t) y f''(t) en el entorno de la raíz, esto se puede visualizar en la Figura 1. Se hicieron pruebas con otras aproximaciones iniciales (x0 = 2.4 y x0=2.7).

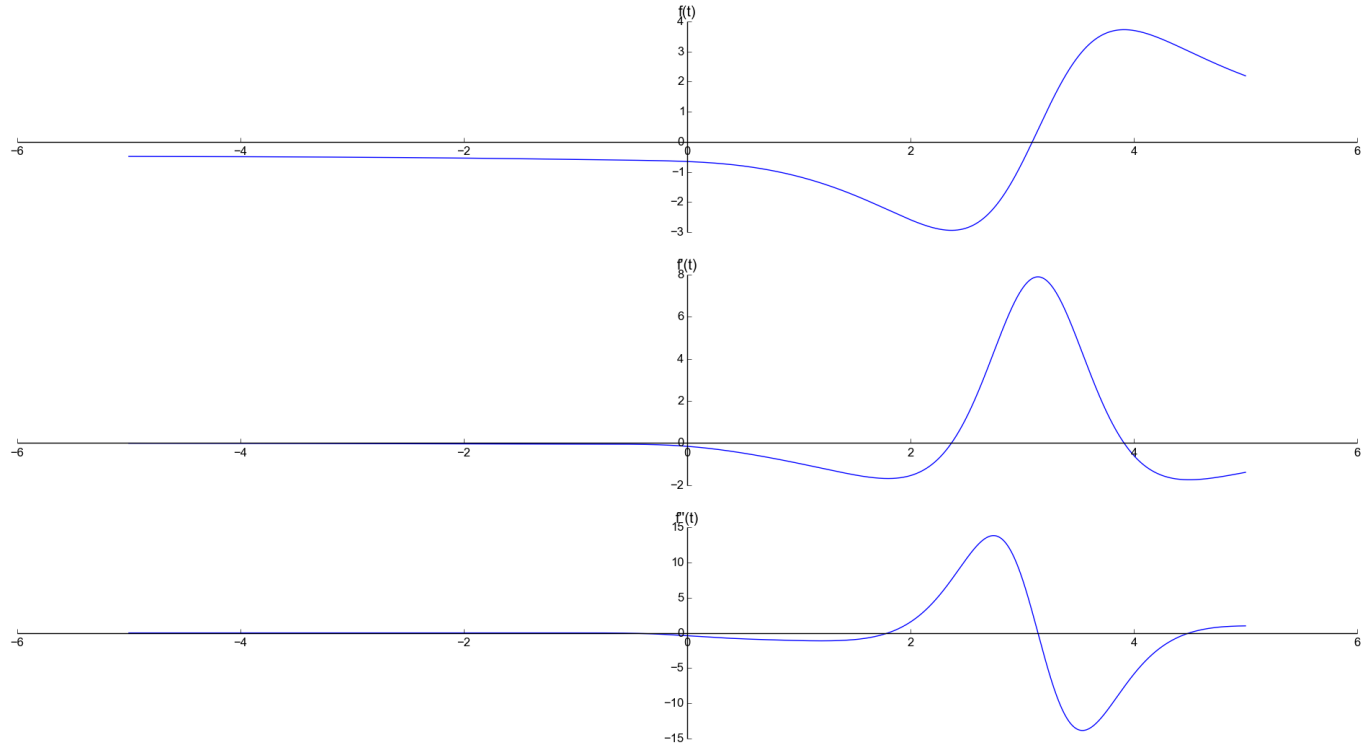

Figura 1. f(t), f'(t), y f''(t)

b) Se implementó el método de Bisección, y se realizaron pruebas aplicándolo en el intervalo previamente definido.

c) Se implementó el método *hibrido(a,b,f,f')*, el mismo realiza 2 iteraciones de bisección entre a y b para obtener una aproximación inicial a la raíz, luego usa esta aproximación como punto de partida para N-R.

2. Resolución de sistemas lineales (ej2.py)

a) Se implementaron las funciones *make_sys(n)* y *make_sys_simple(n)* para la generación del sistema lineal indicado en precisión doble y simple respectivamente.

b) Se implementó *solve_gauss(A,b)* (Eliminación Gaussiana con sustitución hacia atrás).

c) Se realizaron dos implementaciones de Gauss-Seidel, una descomponiendo la matriz en matrices triangulares superior e inferior [1], y otra "despejando" las incógnitas en cada fila.

En la primera implementación, la expresión de iteración del método resulta:

$$\mathbf{x}^{(k+1)} = L_*^{-1}(\mathbf{b} - U\mathbf{x}^{(k)})$$

d) Se comparó la norma del residuo y el tiempo para sistemas de dimensión n (2<=n<=500).

Resultados

1. Búsqueda de raíces

Los resultados obtenidos se presentan en la Tabla I.

Tabla I. Síntesis de los resultados obtenidos.

Método	Punto de partida	Iteraciones	Raíz aproximada	Tolerancia definida
N-R	**3.1**	**2**	**3.09111573155**	**1.48e-15**
	2.4	**999**	**Diverge**	**1.48e-15**
	2.7	**5**	**3.09111573155**	**1.48e-15**
Biseccion	**a=2.366839542192523363221** **b=3.912639867208552436782492507708752127339**	**54**	**3.09111573155**	**1.48e-15**
Híbrido	**a=2.366839542192523363221** **b=3.912639867208552436782492507708752127339**	**6**	**3.09111573155**	**1.48e-15**

2. Resolución de sistemas lineales

Los resultados obtenidos en esta sección se pueden ver en la Figura 2:

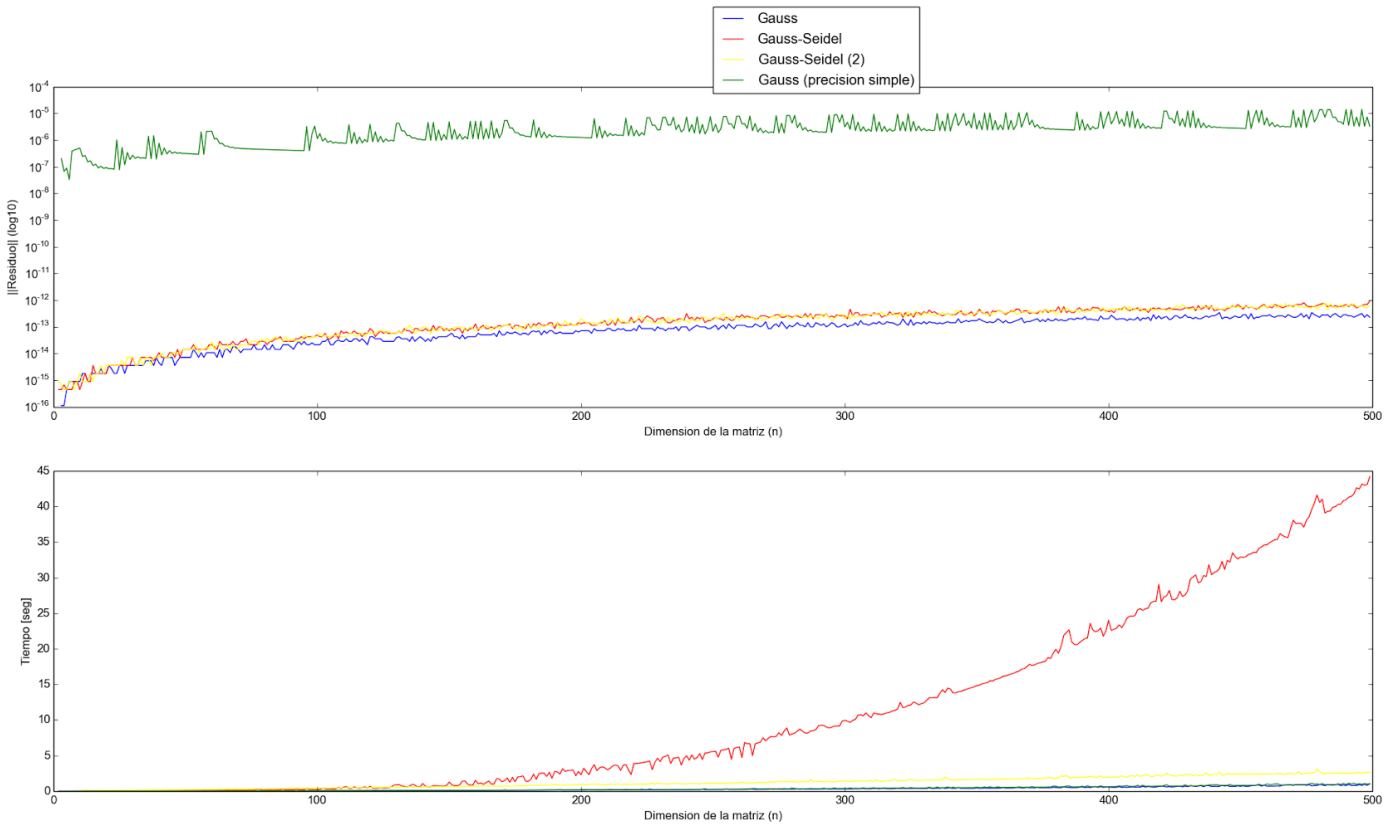

Figura 2. Rendimiento de las distintas implementaciones para n <= 500

Discusión y conclusiones

1. Búsqueda de raíces:

De la tabla presentada puede concluirse fácilmente que N-R tiene una performance mejor que bisección (como puede suponerse con anterioridad). En particular, para x0=3.1, al comenzar prácticamente sobre la raíz, solo se requieren dos iteraciones para lograr la tolerancia esperada. Mientras que al elegir un punto de partida sin garantizar convergencia, uno no puede aseverar el comportamiento del método con anterioridad.

Finalmente, el método híbrido representa una mejora significativa respecto de bisección, con un bajo costo de evaluación y sólo 6 iteraciones para aproximar la raíz. Además, el uso de bisección se muestra como una alternativa razonable para minimizar el espacio de búsqueda rápidamente y obtener una buena aproximación inicial para N-R sin la necesidad del trabajo analítico de garantizar convergencia.

2. Resolución de sistemas lineales:

En el cálculo del residuo se observa la diferencia de precisión de los distintos métodos. En particular, podemos observar que el método de precisión simple tiene una diferencia de aproximadamente 7 órdenes de magnitud con respecto a los otros métodos.

Respecto a los métodos iterativos, Si se analiza con más detalle el rendimiento de las distintas implementaciones de Gauss-Seidel para n < 200 (ver Figura 3) se puede notar que la implementación que hace uso de factorización LU presenta un rendimiento mejor que usando el "despeje" de las incógnitas hasta alrededor de n = 100, mientras que a partir de n=150 el tiempo de resolución de cada sistema por la implementación 1 de Gauss-Seidel empieza a crecer

significativamente más rápido que la implementación dos. Conjeturamos que el producto entre matrices y la rutina *solve_triangular* de Numpy están lo suficientemente optimizados como para que el rendimiento sea mayor en sistemas de dimensiones pequeñas, mientras que a partir de un cierto n, el costo de la inversión de matrices y producto matricial se convierte en el cuello de botella de esta implementación.

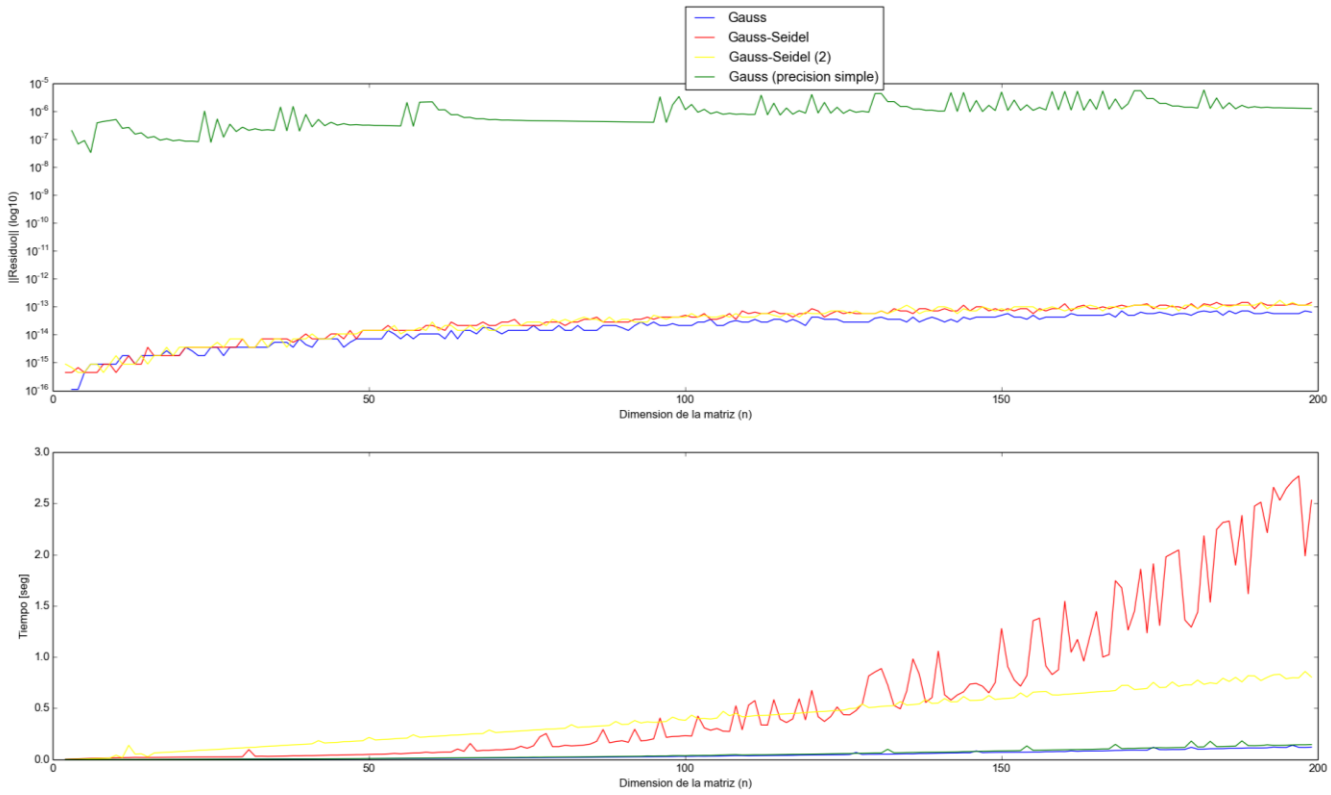

Figura 3. Rendimiento para n <= 200.

Por último, llama la atención el mayor rendimiento de los métodos directos contra las distintas implementaciones de G-S. Se requiere de mayor investigación para dilucidar si esto se debe a la naturaleza del sistema planteado en el enunciado, o si esta tendencia se revierte para un *n* mayor a los analizados.

Referencias

1. Golub, Gene H.; Van Loan, Charles F. (1996), Matrix Computations (3rd ed.), Baltimore: Johns Hopkins, ISBN 978-0-8018-5414-9.