UNIVERSIDAD TECNOLÓGICA NACIONAL

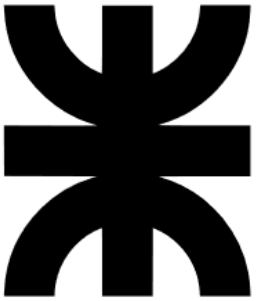
Facultad Regional Santa Fe

Matemática Superior

Ingeniería en Sistemas de Información

Trabajo Práctico 2

Sistemas de ecuaciones no lineales



Alumno	Correo electrónico
Izaguirre, Ezequiel	ezequielmizaguirre@gmail.com
Pacheco Pilan, Federico Ignacio	fedepacheco2112@gmail.com
Rodríguez, Alejando	rodriguezalejandro_@hotmail.com

ISI A • 2021

1. Herramientas utilizadas

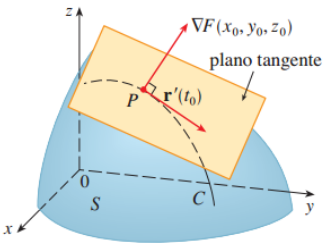
Para la realización del presente trabajo práctico se recurrió al lenguaje de programación *Python* junto a las librerías *sympy, numpy* y *scipy*. El código se adjunta al presente informe.

2. Generalidades del funcionamiento de *Rebote MS*

La implementación en *Python* del método iterativo se realizó con las ideas del enunciado además de lo que se detalla a continuación:

2.1. Recta "paralela" entre dos superficies:

Se aplica el principio de que el vector gradiente de cada función (obtenido usando derivadas simétricas calculadas con *scipy.derivative*), evaluado en un punto, es perpendicular a la curva de nivel que pasa por el mismo punto.



Luego, para obtener un vector perpendicular a ambos vectores gradientes de las superficies se efectúa el producto cruz (usando *numpy.cross*), obteniendo el vector director de la recta que es "paralelo" a ambas superficies de nivel. Como se conoce un punto por donde pasa la recta, la ecuación de la recta queda definida como la suma de dicho punto de paso y el parámetro por el vector director.

$$r(t) = \begin{cases} x = x_0 + v_1 \,.\, t \\ y = y_0 + v_2 \,.\, t \\ z = z_0 + v_3 \,.\, t \end{cases}$$

2.2. Intersección entre una recta y una superficie:

La intersección viene dada por reemplazar las componentes *x*, *y* y *z* de la superficie por las componentes paramétricas de la recta. Así, se obtiene un función de una sola variable *t*, a la cual se pueden aplicar distintos algoritmos de búsqueda de raíces. En la implementación se recurrió a Newton-Raphson escalar provisto por *scipy.optimize.newton*. Como valor inicial se propone $t = 0$. Además de ser una enfoque simple, esto puede ser razonable porque si el

método está convergiendo los "rebotes" son próximos entre sí, con lo que el punto de intersección está cerca del punto de paso de la recta.

2.3. Criterios de corte del método iterativo:

Se consideran dos: cantidad máxima de iteraciones a realizar (no requiere demasiada explicación) y tolerancia. En el caso del segundo criterio, el método finaliza si dicho valor numérico es más grande que la norma infinito del vector cuyas componentes consisten de las tres funciones del sistema no lineal evaluadas en la aproximación a la solución que se tenga en el momento. Se eligió la norma infinito por ser intuitiva: si el valor más lejano al 0 es menor que la tolerancia, las otras dos ecuaciones están aún más próximas a ser 0. Por otra parte es simple de evaluar. Se recurrió a *scipy.linalg.norm*.

3. Análisis de *Rebote MS*

3.1. Comparación con Newton-Raphson para sistemas mediante ejemplos:

Para llevar a cabo la comparación de ambos métodos, se utilizará la implementación de Newton-Raphson (modificada para nuestros fines) abordada en la clase de teoría, ubicada en la sección de *Código*s del campus de la materia (*clase_16Newton.py*). Se usará una tolerancia de 1×10^{-5}. Con anterioridad se consideró además el tiempo de procesamiento (obtención de la solución), pero se llegó a la conclusión de que mantener las mismas condiciones en la computadora (inclusive durante un período corto de tiempo), con el objetivo de obtener los resultados de la forma más consistente posible, era difícil de conseguir. Por otro lado, realizar una comparación relativa (tiempo de un método dividido el otro) tampoco daba resultados constantes.

Sistema	(x^0, y^0, z^0)	***Rebote MS***		***Newton-Raphson***	
		Iteraciones	(x^n, y^n, z^n) (*)	Iteraciones	(x^n, y^n, z^n) (*)
$z - 2 = 0$ $x + y + z + 1 = 0$ $x - 2 = 0$	$(0, 0, 0)$	1	$(-2, -1, 2)$	1	$(-2, -1, 2)$
$z = 0$ $x^2 + y^2 + z = 0$ $x = 0$	$(-40, 39, 22)$	1	$(0, 0, 0)$	1258	$(0, 0, 0)$
$cos(x) - z = 0$ $x^2 + y^2 + 1 - z = 0$ $x + y + z - 1 = 0$	$(1, 0, 0)$ $(1, 1, 1)$	4 "∞"	$(0.0014, -0.0014, 1)$ -	469 Matriz singular	(0.0024, -0.002, 1) -
$z - e^{xy} = 0$ $y - x - z - 4 = 0$ $(y - 3)^2 + x - z = 0$	$(-3, 4, 5)$ $(0, 0, 0)$	4 3	$(-0.417, 3.79, 0.206)$ $(-7.038, -0.15, 2.888)$	14 21	$(-0.417, 3.79, 0.206)$ $(-2.668, 1.35, 0.026)$

$xcos(y) - z = 0$ $x^2 + y^2 - 2 - z = 0$ $(1/x) + y^2 - 2 - z = 0$	$(1, 0, 0)$	2	$(1, -1.177, 0.384)$	Matriz singular	-
$(cos(x)/x) + y - z = 0$ $z^2 + x - 2 - z = 0$ $(1/y) + 2x - z = 0$	$(1, 1, 0)$	4	$(-0.568, 0.306, -1.179)$	Matriz singular	-
$cos(y) * x + sen(x) - z = 0$ $(1/sen(x)) + 4 - z = 0$ $y + x - z = 0$	$(2, 3, 4)$	3	$(-9.243, 7.706, -1.536)$	107	$(-4.24, 9.363, 5.122)$
$x^2 + y - 1 = 0$ $y = 0$ $ln(x) - z = 0$	$(1, 4, 1)$	3	(1,0,0)	1	(1,0,0)

(*) Redondeado a 4 cifras decimales.

3.2. Casos favorables para la aplicación de Rebote MS:

De manera intuitiva, el método debe comportarse favorablemente si la raíz a buscar queda "encerrada" entre las superficies, de manera de facilitar los rebotes, que progresivamente se harían más cortos hasta "caer" a la raíz. También se pensó que el punto inicial elegido debe estar relativamente "cerca" de la raíz (análogo a Newton-Raphson para sistemas), sin embargo por las pruebas que se han hecho esta no parece ser una condición tan necesaria.

3.3. Casos perjudiciales para la aplicación de Rebote MS:

Bajo la presente implementación del método, este tendrá inconvenientes siempre que se obtenga un vector director nulo para alguna de las rectas empleadas en cada ciclo, o directamente no pueda calcularse. El primer caso hace que, al intersecar la "recta" (en realidad un punto) con la superficie, el método de Newton escalar falle por intentar obtener las raíces de una función constante. Esta situación -vector nulo o incalculable- se puede producir cuando:

1. Los vectores normales a las superficies de nivel en un punto son paralelos o antiparalelos.
2. Alguno o ambos vectores normales son nulos.

3. Existe al menos un componente de la función gradiente que presenta una discontinuidad, o no está definido, en el punto de análisis.

Otro caso perjudicial se dará cuando, para una iteración dada, el punto de intersección de una recta con la superficie quede “demasiado lejos” del punto de paso empleado. En este caso la búsqueda de raíz podría llevar un elevado número de cálculos. Adicionalmente, dependiendo de la naturaleza de esta evaluación, comenzar desde $t = 0$ podría llevar a un caso donde Newton diverge o se queda en un ciclo, por lo que ésta no sería una buena propuesta de valor inicial.

Por último, *Rebote MS* por como fue concebido sólo es apto para sistemas con tres ecuaciones y funciones de tres variables. Esto supone un inconveniente para casos más complejos.

4. Conclusión

Se puede concluir que *Rebote MS* es comparable a Newton-Raphson para sistemas, e incluso lo supera en muchos de los casos analizados (ver sección 3.1.).

En lo que respecta a posibles mejoras al método, estas van de la mano con lo comentado en las secciones 3.2 y 3.3. Para la cuestión del vector director de la recta nulo una opción es proponer, aleatoriamente o con algún criterio, un vector perteneciente al plano tangente en un punto a alguna de las superficies. Otra opción es directamente "pasar al siguiente paso" dentro de una iteración dada. Para el caso de gradiente no definido en un punto, también puede emplearse la estrategia recién descripta.

Para la cuestión de la divergencia del Newton-Raphson escalar, se requeriría hacer un estudio más profundo (continuidad, monotonía, cambio en el signo, si existen puntos de inflexión) de la función a encontrarle las raíces para, por un lado, encontrar una raíz, y por otro, encontrar la raíz correcta. Al momento de redacción, no se tiene muy en claro cómo se puede hacer de manera genérica, empleando sólo herramientas de cálculo numérico y factible en términos de rendimiento.

5. Webgrafía

- https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.norm.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.misc.derivative.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html
- https://en.wikipedia.org/wiki/Symmetric_derivative
- https://numpy.org/doc/stable/reference/generated/numpy.cross.html
- https://numpy.org/doc/stable/reference/generated/numpy.isnan.html
- https://stackoverflow.com/questions/10710328/comparing-numpy-arrays-containing-nan
- https://stackoverflow.com/questions/20708038/scipy-misc-derivative-for-multiple-argument-function
- https://stackoverflow.com/questions/50081980/finding-local-maxima-and-minima-of-user-defined-functions
- https://www.geeksforgeeks.org/how-to-create-a-vector-in-python-using-numpy/