1. Búsqueda de raíz

Cinética Química

En un sistema de reacción química, la concentración de un compuesto intermedio x se encuentra regulada por un mecanismo de retroalimentación que depende **logarítmicamente** de su propia concentración. Este tipo de comportamiento puede surgir, por ejemplo, en procesos enzimáticos o reacciones autocatalíticas con inhibición por producto.

$$x \approx \ln(x) \tag{1}$$

Sin embargo, este procedimiento depende de diversos parámetros relacionados a las propiedades del compuesto y el medio en el que se encuentra. Porr lo que considerando:

- x: como la concentración del compuesto intermedio.
- $k > 0$: una constante que representa la sensibilidad del sistema a la retroalimentación.
- $a > 0$, $b > 0$: parámetros que modelan la cinética y el punto de referencia de la retroalimentación.

Se puede llegar a una ecuación más general:

$$x \approx \frac{1}{k} \ln(ax + b) \tag{2}$$

En la que si se cumple la equivalencia significa que el sistema ha alcanzado el equilibrio. Sin embargo, esto no ocurre de forma inmediata por lo que una versión de la ecuación de mayor utilidad practica es aquella que modela el error o desfase del punto de equilibrio a partir de:

$$x \approx \frac{1}{k} \ln(ax + b) \tag{3}$$

$$x = \frac{1}{k} \ln(ax + b) + \Delta \tag{4}$$

$$\Delta = x - \frac{1}{k} \ln(ax + b) \tag{5}$$

A partir de esto, se modela el desequilibrio del sistema con la siguiente función de error:

$$f(x) = \left(x - \frac{1}{k} \ln(ax + b) \right)^2 \tag{6}$$

Este tipo de ecuación no puede resolverse de forma analítica en general, por lo que una alternativa es utilizar métodos numéricos para su resolución. Por esto, el departamento de química le solicita a usted un software que calcule de forma rápida, segura y lo más exácta posible el punto de equilíbrio de un sistema con valores de k, a y b dados.

Preguntas

Puede interesarle una explicación más puntual del contexto, la cual se incluye a continuación: el objetivo es encontrar el punto de equilibrio del sistema quimico, el cual se establece cuando:

$$x = \frac{1}{k} ln(ax + b)$$

lo que se puede entender también, como que la diferencia entre ambos lados de la ecuación se hace 0:

$$0 = x - \frac{1}{k} ln(ax + b)$$

Sin embargo, antes de este punto la ecuación se encuentra desequilibrada, por lo que nombramos como Δ a esa diferencia que aún no es 0:

$$\Delta = x - \frac{1}{k} ln(ax + b)$$

Finalmente, se llega a una versión más útil para este parametro de desequilibrio a través de un análogo de la función de error cuadrático, la cual se define como:

$$f(x) = \left(x - \frac{1}{k} ln(ax + b)\right)^2$$

Con esto claro podemos responder las preguntas.

(a) Describa y programe un algorítmo basado en el método de Newton que reciba los valores de x_0, k, a y b y encuentre el punto de equilibrio del sistema químico.

Notar que solo se nos pide el algoritmo mas no resolverlo, por lo que solo debemos establecer los parametros y un algorítmo que nos de las garantias esperadas:

Se debe utilizar el método de Newton, el cual requiere de la derivada de la función a iterar:

$$f'(x) = 2 \cdot \left(x - \frac{1}{k} ln(ax + b)\right) \cdot \left(1 - \frac{a}{k(ax + b)}\right)$$

Con esto listo podemos establecer una cantidad de iteraciones arbitraria ya que el contexto no nos da un parametro para ello (el ayudante les pide disculpas por esta falta).

Finalmente solo nos falta explicar nuestro algoritmo:

- Se utilizará el método de Newton para encontrar alguna raiz de $f(x)$ cercana al initial guess x_0 con parametros a, b y k dados, es decir, f es unidimensional. $f'(x)$ se encuentra descrita anteriormente y es un parámetro del algoritmo y la cantidad de iteraciones para aproximar esta raiz debe ser dada pero se fijara en 10 por defecto. Con esto se obtendrá una aproximación del valor de $\hat{x}$ tal que $f(\hat{x}) = 0$, es decir, la concenctración de sustancia necesaria para que el sistema llegue al equilibrio químico.

Se le recomienda describir el codigo para reforzar los contenidos asociados.

(b) Usted termina su algoritmo y decide probarlo utilizando $x_0 = 2.5$, $k = 1$, $a = 1$ y $b = 2$. Con lo que obtiene los resultados de la Tabla 1. Determine si su algorítmo es la mejor alternativa entre la que ha estudiado en Computación Científica y de no ser así, desarrolle una solución que garantice que es la mejor.

Iteración	x_i	$x_i - x_{i-1}$	$f(x_i)$
1	2.13090	–	5.08e-01
2	1.66093	-0.46997	1.32e-01
3	1.41107	-0.24985	3.39e-02
4	1.28088	-0.13019	8.61e-03
5	1.21416	-0.06672	2.17e-03
6	1.18034	-0.03382	5.45e-04
7	1.16331	-0.01703	1.37e-04
8	1.15476	-0.00855	3.42e-05
9	1.15048	-0.00428	8.56e-06
10	1.14834	-0.00214	2.14e-06

Tabla 1: Resultados del sistema para $x_0 = 2.5$, $k = 1$, $a = 1$ y $b = 2$ utilizando el método de Newton-Raphson.

Como estamos utilizando el método de Newton, lo primero es estudiar la convergencia cuadrática. Para ello podemos utilizar **las últimas iteraciones**, las cuales corresponden a nuestras **mejores aproximaciones de la raíz real** por lo que nos darán las **mejores aproximaciones de las tasas de convergencia**:

$$M \approx M_i = \frac{x_i - x_{i-1}}{(x_{i-1} - x_{i-2})^2}$$

$$M_{10} = \frac{x_{10} - x_9}{(x_9 - x_8)^2}$$
$$M_{10} = \frac{-0.00214}{(-0.00428)^2}$$
$$M_{10} = -116.82...$$

Usted podría considerar que como $-116.82 < \infty$ la tasa de convergencia cuadrática está garantizada, sin embargo, recordar que esto era a través de un límite al infinito, por lo que no es una equivalencia, sino una tendencia, en otras palabras, no se

espera que el calculo de M de ∞, se espera que a través de las interaciones, este valor vaya creciendo indefinidamente. Con esto claro, sabemos que es necesario revisar otra iteración:

$$\begin{aligned} M_9 &= \frac{x_9 - x_8}{(x_8 - x_7)^2} \\ M_9 &= \frac{-0.00428}{(-0.00855)^2} \\ M_9 &= -58.54... \end{aligned}$$

Notar que de la iteración 9 a la 10 el modulo de M creció significativamente, es decir, existe una tendencia a aumentar con el paso de las iteraciones, en terminos matemáticos, podemos decir que a medida que $i \to \infty$ tambíen $M \to \infty$.

Como la tasa cuadrática no dió un valor fijo, podemos analizar la tasa de convergencia lineal:

$$S \approx S_i = \frac{|x_i - x_{i-1}|}{|x_{i-1} - x_{i-2}|}$$

$$\begin{aligned} S_{10} &= \frac{|x_{10} - x_9|}{|x_9 - x_8|} \\ S_{10} &= \frac{0.00214}{0.00428} \\ S_{10} &= 0.5 \end{aligned}$$

$$\begin{aligned} S_9 &= \frac{|x_9 - x_8|}{|x_8 - x_7|} \\ S_9 &= \frac{0.00428}{0.00855} \\ S_9 &= 0.50058... \end{aligned}$$

Notar que también es necesario calcular más de una iteración para revisar si efectivamente la tasa lineal se estabiliza con el paso de las iteraciones.

La tasa de convergencia lineal se centra en 0.5 por lo que podemos decir que el método de Newton está convergiendo linealmente. A partir de esto pueden surgir varias soluciones, pero la más importante de revisar es si es posible utilizar el método de Newton modificado:

$$\frac{m-1}{m} = S$$

Si $S = 0.5$:

$$\begin{aligned} \frac{m-1}{m} &= 0.5 \\ m - 1 &= \frac{m}{2} \\ 2m - 2 &= m \\ 2m - m &= 2 \\ m &= 2 \end{aligned}$$

Recordad que m corresponde a la mutiplicidad de la raíz, este valor es un parámetro del método de Newton modificado con el que se puede garantizar que se recuperará la convergencia cuadrática: Existen otras formas de calcular este m como podría ser derivar la función nuevamente y revisar en que momento la raíz evaluada deja de ser 0 o incluso una forma más rápida aunque compleja de ver, es analizar visualmente la función $f(x)$, que si es una función de segundo orden (y por tanto dentro del exponente no pueden estar contenidas funciones con más exponentes), entonces tras derivar 2 veces obtendremos una constante, con lo que sabremos que las raices tienen multiplicidad 2. Tener mucho cuidado con este análisis pues todo depende de la función $g(x)$ en la composición $f(x) = (g(x))^2$.

Con esto claro sabemos con total seguridad de que es posible recuperar la convergencia cuadrática, con esto claro, no es necesario estudiar la posibilidad de una IPF o la bisección, ya que sabemos que su convergencia será peor que cuadrática.

2. Pérdida de importancia

Se propone una variante del formato de double precision del estándar de punto flotante de la IEEE754, en la cual, si bien, aún se consideran 64 bits en total, se representa a traves del formato FPS(m), donde m denota la cantidad de bits utilizados para el exponente. En específico , FPS(m) se describe de la siguiente forma:

- Número de bits utilizados para el signo: 1
- Número de bits utilizados para el exponente: m
- Número de bits utilizados para la mantisa: $63 - m$

En particular se considera que $m \in \{2, ..., 62\}$, así se asegura que se pueden considerar los mismos casos especiales que en la versión tradicional del formato double precision, que en esta representación corresponde a FPS(11). En esta variante, el bias se obtiene como $2^{m-1} - 1$.

(a) Determine el Machine Epsilon en el formato FPS(m) para todo m. Justifique su resultado.

Recordar que el Machine Epsilon es el intervalo entre el 1 y el siguiente número representable, para lo cual se aumenta el bit de más a la derecha de la mantisa, el cual en el caso de double precission correspondía al 52avo bit:

$$+1.000000...000000 \cdot 2^0 = 1$$

$$+1.000000...000001 \cdot 2^0 = 1 + 2^{-52}$$

$$\epsilon_{mach} = 2^{-52}$$

Sin embargo, en este caso en vez de 52 bits de mantisa se tienen $63 - m$ por lo que el Machine Epsilon sería:

$$+1.000000...000000 \cdot 2^0 = 1$$

$$+1.000000...000001 \cdot 2^0 = 1 + 2^{-63+m}$$

$$\epsilon_{mach} = 2^{-63+m}$$

(b) Determine el menor número positivo representable en el formato FPS(m) para todo m. Justifique su resultado.

El ejecicio nos sugiere que existe un shift del exponente, para poder representar exponentes negativos. De todos modos, por completitiud se recuerda la demostración del rango de exponentes validos para double precission:

- el rango de exponentes va desde un exponente con 11 bits en 0 hasta uno con 11 bits en 1, lo cual en decimal representa el intervalo $[0, 2047]$
- pero a este intervalo se le quitan los extremos para los casos especiales que comentaremos en un momento, por lo que el intervalo valido es $[1, 2046]$
- este rango se intenta centrar en 0 buscando tener la misma cantidad de exponentes positivos, que de negativos, para lo cual se introduce un shift, que corresponde a $2^{11-1} - 1 = 1023$.
- Así nos resulta el conocido intervalo: $[1 - 1023, 2046 - 1023] = [-1022, 1023]$

Entendida la motivación del shift, revisamos los casos particulares: partiremos por el que corresponde a un exponente de solo 1's (111...111):

- si el bit de signo es 0 y la mantisa son solo 0's el número representa $+\infty$
- si el bit de signo es 1 y la mantisa son solo 0's el número representa $-\infty$
- si algún bit de la mantisa es distinto de 0 entonces es un NaN.

El otro caso especial y el cual nos importa para responder esta pregunta, es cuando el exponente son solo 0's (000...000): Si recordamos, el formato normalizado, que es el que se usa la mayoría del tiempo asume que el número siempre es un $1.bbb...$ como se señala en (10). Sin embargo, cuando ocurre el caso especial de que el exponente sean solo 0's se pasa al formato no-normalizado, en el cual el número es $0.bbb...$ (11):

- Formato normalizado:

$$\pm 1.bbb... \cdot 2^p \tag{7}$$

- Formato no-normalizado (caso especial en el que el exponente es 000...000):

$$\pm 0.bbb... \cdot 2^p \tag{8}$$

En el caso de tener solo 0's se considera que el exponente es el mínimo posible por lo que sería el intervalo desplazado en el shift que nos entregan:

- exponente: $1 - (2^{m-1} - 1) = 2 - 2^{m-1}$

notar que estamos desplazando el lado izquierdo del intervalo (el cual tambien parte inicialmente en 1).

Así en el caso no normalizado el cual nos quita el 1 reglamentario que se adhiere a la izquierda de la mantisa, y ya con el menor exponente posible, solo queda representar el número más pequeño posible con esta mantisa no-normalizada. En el caso de double precission son 52 bits de mantisa por lo que queda:

- mantisa: $0.000...001 = 2^{-52}$

Sin embargo, ahora tenemos 63-m bits, por lo que en nuestro caso queda:

- mantisa: $0.000...001 = 2^{-63+m}$

Así juntando mantisa y exponente nos queda el siguiente número:

- $0.000...001 \cdot 2^{2-2^{m-1}} = 2^{-63+m} \cdot 2^{2-2^{m-1}} = 2^{-61+m-2^{m-1}}$

Para revisar podemos utilizar el caso FPS(11) que corresponde al double precission estandar:

- $2^{-63+11} \cdot 2^{2-2^{11-1}} = ^{-52} \cdot 2^{2-2^{10}} = 2^{-52} \cdot 2^{2-1024} = 2^{-52} \cdot 2^{-1022}$

Por lo tanto el número más pequeño representable en el formato FPS(m) sería:

$$2^{-61+m-2^{m-1}}$$

(c) Determine el primer entero no representable para cualquier m.

Para entender este ejercicio revisaremos primero un caso pequeño, en el que considerarémos 3 bits de mantisa y 3 de exponente:

mantisa	exponente	binario	decimal
1.000	000	1.000	1
1.001	000	1.001	1.125
1.010	000	1.010	1.250
1.011	000	1.011	1.375
1.100	000	1.100	1.500
1.101	000	1.101	1.625
1.110	000	1.110	1.750
1.111	000	1.111	1.875
1.000	001	10.000	2.000
1.001	001	10.010	2.250
1.010	001	10.100	2.500
1.011	001	10.110	2.750
1.100	001	11.000	3.000
...	...	...	...
1.111	011	1111.000	15.000
1.000	100	10000.000	16.000
1.001	100	10010.000	18.000

Respecto a la tabla anterior se sacan las siguientes conclusiones:

- notar que al desplazarse entre el 1 y el 2, el aumentar un bit implica aumentar el número decimal en 0.125, sin embargo entre 2 y 4, el aumentar un bit implica aumentar el número decimal en 0.25. Lo anterior ocurre debido a que al pasar al 2, necesitamos hacer crecer el exponente, por lo que perdemos el bit menos significativo a costa de ganar un bit más significativo, es decir, estamos shifteando la mantisa.
- Si la situación evidenciada se extrapola, se puede intuir que en algún punto el número empezara a crecer en intervalos lo suficientemente grandes como para saltar unidades, pero ¿Cuando?.
- Se empiezan a saltar unidades cuando luego de shiftear toda la mantisa, se hace un shift más, lo que implica que al dar el siguiente paso (aumentar el bit menos significativo), en vez de aumentar una unidad, se aumentarán dos. Es decir, cuando el exponente sea igual a la mantisa más uno, y se intente dar el siguiente paso, se saltará el **primer entero no representable**.
- Lo anterior pasado a números, considerando una mantisa n, queda:

$$2^{\text{mantisa} + 1} + \text{un paso (2 unidades)} - 1 = 2^{n+1} + 1$$

Como en nuestro caso la mantisa es $63 - m$, el primer entero no representable queda expersado por la ecuación:

$$2^{63-m} + 1$$

(d) Determine el m más pequeño para el cual existe al menos un entero no representable dentro del intervalo que es capaz de representar.

Lo primero es entender la pregunta: nos piden encontrar el m, osea el exponente, más pequeño, para el que aparece un entero no representable **dentro** de su intervalo. Esto puede parecer raro, pero si volvemos a la tabla de la pregunta anterior: si el exponente fuera 2, en vez de 3, no habría sido posible representar el 18, por lo que en ningún momento habriamos tenido enteros no representables, dentro del intervalo que abarca nuestra representación con muy pocos bits. Por otro lado, desde que se tengan 3 bits para el exponente, en adelante, el problema siempre existirá, debido a que el intervalo ya abarca un entero no representable fijado por la precisión de la mantisa. Así podemos encontrar un valor para el exponente, para el cual empiezan a aparecer números no representables, pues como solo se tienen 64 bits para la representación: a costa de extender el rango de números abarcables (más bits para el exponente), se pierde precisión (menos bits para mantisa). Pero ¿como podemos saber cuando aparece este número? bueno, es fácil notar que lo único necesario es que el intervalo permitido por el exponente abarque dicho numero, así:

$$2^{64-m} + 1 \leq 2^{\text{exponente más grande posible}}$$

Si revisamos la pregunta b) notarémos que el exponente más grande posible corresponde al shift. Por lo que:

$$2^{64-m} + 1 \leq 2^{2^{m-1}-1}$$

$$log_2(2^{64-m} + 1) \leq log_2(2^{2^{m-1}-1})$$

$$log_2(2^{64-m}) < log_2(2^{64-m} + 1) \leq log_2(2^{2^{m-1}-1})$$

$$64 - m < 2^{m-1} - 1$$

$$65 - m < 2^{m-1}$$

$$65 - m < 2^{m-1}$$

Considerando que m solo toma valores enteros y sabiendo que $2^5 = 32$ y $2^6 = 64$ podemos revisar ambos casos:

- $m = 6$

$$65 - 6 < 2^{6-1}$$

$$59 < 2^5 = 32$$

- $m = 7$

$$65 - 7 < 2^{7-1}$$

$$58 < 2^6 = 64$$

Podemos ver que el valor más pequeño que puede tomar m es 7. Notar que se hizo una simplificación en la desigualdad quitando el caso de la igualdad, esto se revisa a continuación:

- $m = 6$

$$2^{64-6} + 1 \leq 2^{2^{6-1}-1}$$

$$2^{58} + 1 \leq 2^{2^5-1}$$

$$2^{58} + 1 \leq 2^{31}$$

- $m = 7$

$$2^{64-7} + 1 \leq 2^{2^{7-1}-1}$$

$$2^{57} + 1 \leq 2^{2^6-1}$$

$$2^{57} + 1 \leq 2^{63}$$

Así queda demostrado que el m más pequeño para el cual aparecen enteros no representables es 7.