Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Métodos Numéricos

Trabajo Práctico Nº1

"Perdidos" en el Pacífico

Nombre	LU	Mail
Pablo Herrero	332/07	pablodherrero@gmail.com
Thomas Fischer	489/08	tfischer@dc.uba.ar
Kevin Allekotte	490/08	kevinalle@gmail.com

Abstract

Este trabajo pretende hacer una comparación sobre el desempeño y la precisión de la aritmética flotante de diversos entornos y mostrar en los resultados lo que pueden producir pequeños errores de cálculo. Para ello trabajamos sobre la *Recurrencia de Müller* detallada en el Apéndice A, sobre la cuál sabemos que converge y conocemos una ecuación cerrada para sus términos, lo que nos permite calcular el límite exacto analíticamente. Las pruebas consisten en comparar el límite exacto de la sucesión con el límite obtenido por la sucesión de recurrencia programada en distintos entornos y lenguajes.

Aritmética Finita **Recurrencia de Müller**

Índice

Introducción teorica

Cuando trabajamos con números reales en una computadora, nos encontramos con algunos factores limitantes para nuestros cálculos como lo son la aritmética finita y también el tipo de representación que se nos provee para trabajar.

Esto surge como una consecuencia de la necesidad de una memoria física finita de la computadora en contraste con la precisión infinita que requieren la mayoría de los números reales.

Luego, al intentar representar números en una computadora, cuya precisión va mas allá de la otorgada por la misma, surgen en la representación del número pequeños errores, los cuáles, como muestra este trabajo, no son despreciables al realizarse ciertos tipos de cálculos.

Cada uno de estos sistemas puede tener una precisión muy distinta, lo que produce una diferencia en los errores cometidos al representar números en cada uno.

Además existen distintos tipos de precisión a analizar, como por ejemplo la resolución o granularidad, o los limites superiores e inferiores, tanto de la parte entera como de la parte decimal.

La representación que elijamos depende totalmente de los tipos de cálculo que vallamos a realizar y el tipo de precisión que necesitemos para obtener un resultado razonable.

Existen múltiples estándares de representación para los números reales, de los cuáles tal vez el más difundido sea la norma `IEEE-754`.

Desarrollo

Análisis Teórico de la sucesión

En primer lugar, es necesario un análisis de la fórmula cerrada de la *Recurrencia de Müller*, para encontrar el límite al cual converge.

La formula cerrada para los términos de la *Recurrencia de Müller* esta dada por

$$x_n \;=\; \frac{\alpha 3^{n+1} + \beta 5^{n+1} + \gamma 100^{n+1}}{\alpha 3^n + \beta 5^n + \gamma 100^n}$$

por lo cuál el límite esta dado por $\lim_{n\to\infty} x_n$. Separando en casos, calculamos los posibles límites de la siguiente manera.

- para $\gamma \neq 0$

$$\begin{aligned}
\lim_{n\to\infty} \frac{\gamma 100^{n+1}}{\alpha 3^n + \beta 5^n + \gamma 100^n} &\leq \lim_{n\to\infty} x_n \leq \lim_{n\to\infty} \frac{\alpha 3^{n+1} + \beta 5^{n+1} + \gamma 100^{n+1}}{\gamma 100^n} \\
\lim_{n\to\infty} \frac{\gamma 100}{\alpha(\frac{3}{100})^n + \beta(\frac{5}{100})^n + \gamma} &\leq \lim_{n\to\infty} x_n \leq \lim_{n\to\infty} \frac{\alpha 3^{n+1}}{\gamma 100^n} + \frac{\beta 5^{n+1}}{\gamma 100^n} + \frac{\gamma 100^{n+1}}{\gamma 100^n} \\
\lim_{n\to\infty} 100 &\leq \lim_{n\to\infty} x_n \leq \lim_{n\to\infty} \frac{3\alpha}{\gamma}(\frac{3}{100})^n + \lim_{n\to\infty} \frac{5\beta}{\gamma}(\frac{5}{100})^n + \lim_{n\to\infty} 100 \\
100 &\leq \lim_{n\to\infty} x_n \leq 100 \\
\Rightarrow \quad \lim_{n\to\infty} x_n &= 100
\end{aligned}$$

- para $\gamma = 0$ y $\beta \neq 0$

$$\begin{aligned}
\lim_{n\to\infty} \frac{\beta 5^{n+1}}{\alpha 3^n + \beta 5^n} &\leq \lim_{n\to\infty} \tfrac{\alpha 3^{n+1} + \beta 5^{n+1}}{\alpha 3^n + \beta 5^n} \leq \lim_{n\to\infty} \frac{\alpha 3^{n+1} + \beta 5^{n+1}}{\beta 5^n} \\
\lim_{n\to\infty} \frac{\beta 5}{\alpha(\frac{3}{5})^n + \beta} &\leq \lim_{n\to\infty} x_n \leq \lim_{n\to\infty} \frac{\alpha 3^{n+1}}{\beta 5^n} + \frac{\beta 5^{n+1}}{\beta 5^n} \\
\lim_{n\to\infty} 5 &\leq \lim_{n\to\infty} x_n \leq \lim_{n\to\infty} \frac{3\alpha}{\beta}(\frac{3}{5})^n + \lim_{n\to\infty} 5 \\
5 &\leq \lim_{n\to\infty} x_n \leq 5 \\
&\Rightarrow \lim_{n\to\infty} x_n = 5
\end{aligned}$$

- para $\gamma = 0$, $\beta = 0$ y $\alpha \neq 0$

$$\lim_{n\to\infty} \frac{\alpha 3^{n+1}}{\alpha 3^n} = \lim_{n\to\infty} 3 = 3$$

- para $\gamma = \beta = \alpha = 0$ la función no esta definida, y por lo tanto no nos interesa, porque suponemos que para todo $x0$, $x1$ existen α, β y γ que cumplen con la fórmula cerrada de tal manera que alguno de ellos es distinto de 0.

Luego queremos ver para que casos de entrada, o valores de $x0$ y $x1$, se dan las distintas opciones para α, β y γ.

Consideramos que si γ puede ser 0 entonces lo es, y solamente en caso contrario $\gamma! = 0$.

- para $\gamma = 0$

  Despejamos la fórmula cerrada para $x0$ y $x1$ y restando las ecuaciones obtenemos la igualdad

$$b(\frac{5 - x_0}{x_0 - 3} - \frac{5}{3}\frac{5 - x_1}{x_1 - 3}) = 0$$

  de la cuál podemos deducir que $x0(x1 - 8) = -15$.

  Dentro de este caso existe otro caso especial en el cual $\beta = 0$ y $\alpha \neq 0$. Podemos ver que este caso solo aparece cuando $x0 = x1 = 3$.

- El resto de las entradas cae en el caso $\gamma \neq 0$

Luego quedan resueltos los límitesde la *Recurrencia de Müller* en función de las distintas entradas de la siguiente manera:

$$\lim_{n\to\infty} x_n = \begin{cases} 3 & x0 = 3 \wedge x1 = 3 \\ 5 & x0(x1 - 8) = -15 \\ 100 & \text{sino} \end{cases}$$

Implementación

Luego el trabajo consistió en implementar la *Recurrencia de Müller* en distintos lenguajes y plataformas, y calcular hasta un n-ésimo término la sucesión de tal manera que $|x_n - x_{n-1}| < \epsilon$ donde ϵ es una diferencia entre términos sucesivos a partir de la cuál consideramos que la recurrencia ya ha alcanzado el límite.

Los resultados de ejecutar estos programas son comparados con los resultados teóricos de límite obtenidos, explicados en la seccion precedente. Así podemos obtener un valor del error cometido por el algoritmo por consecuencia de las limitaciones de representación aritmética de las computadoras.

Para Implementar la recurrencia elegimos los lenguajes:

- `C++` con precisión simple (`float`)
- `C++` con precisión doble (`double`)
- `Python`
- `Python` con precisión arbitraria
- `OpenOffice Spreadsheet`

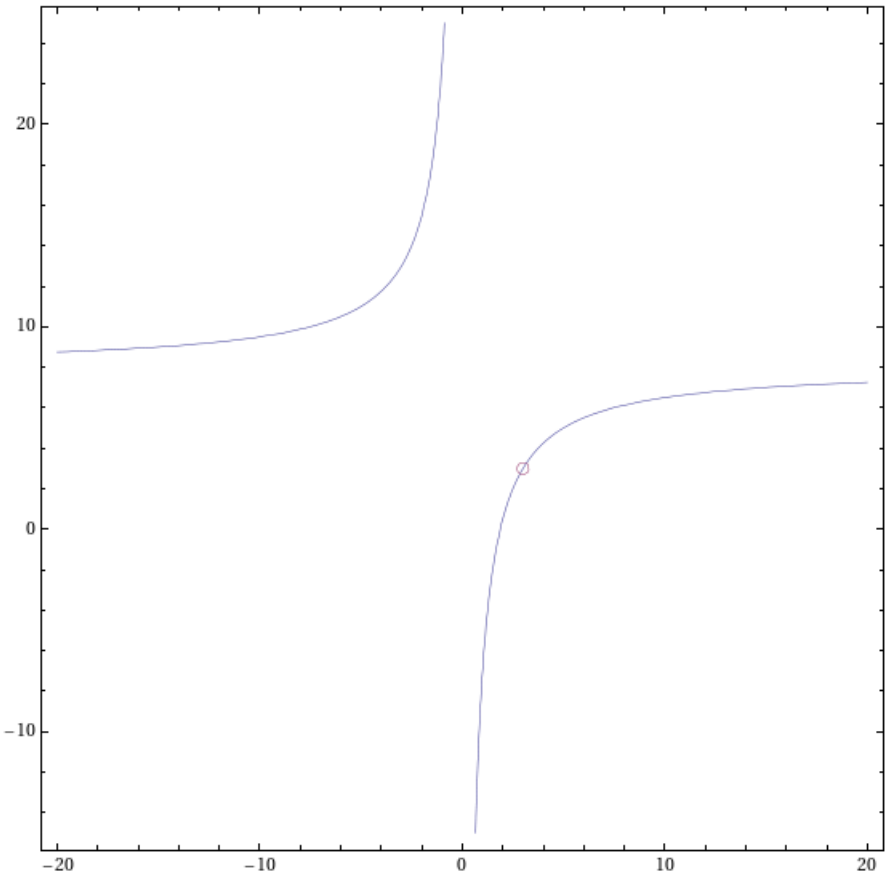
Creemos que este subconjunto de lenguajes ilustrará los resultados para entornos distintos para poder comparar.

Los resultados obtenidos son analizados en la sección siguiente.

Resultados

En este primer gráfico representamos los pares (x_0, x_1) para los cuales el limite teórico es 5 (línea azul), y el punto $(3, 3)$ que es el único que tiene límite teórico 3. El resto del plano son los (x_0, x_1) para los cuales el límite es 100.

Figura 1: Dominio del límite de la sucesión



A continuación graficamos los sucesivos valores de x_n para $1 \leq n \leq 20$ con $x_0 = 2$ y $x_1 = 8$, comparando el resultado exacto con el resultado de generar la sucesion con la fórmula recurrente usando C++ (con double de presicion).

Se observa que a pesar de unos errores importantes el límite es el mismo.

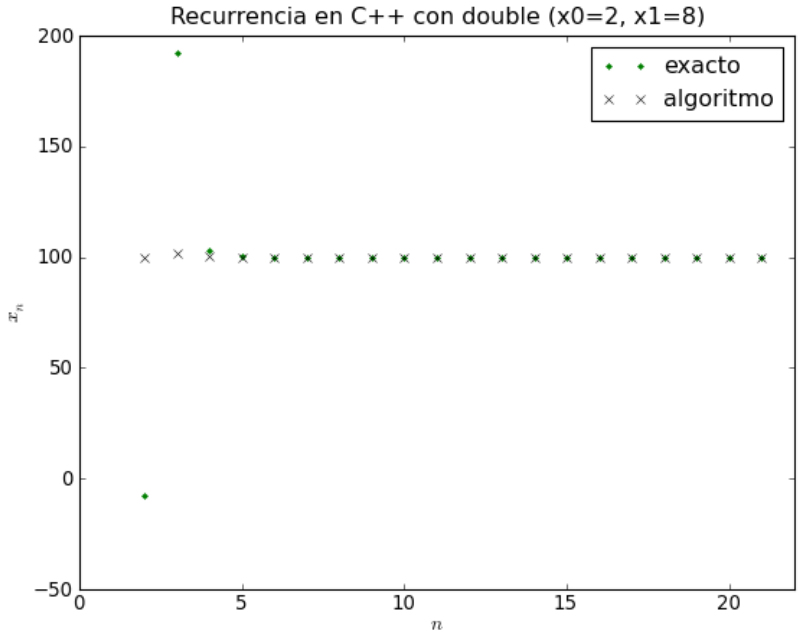
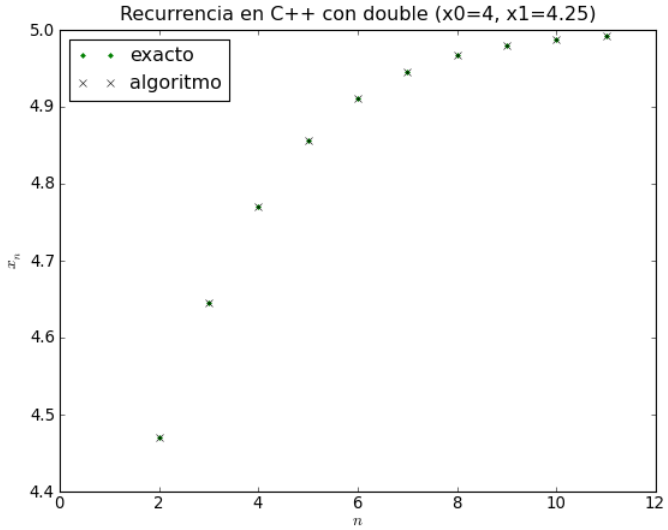


Figura 2: Acá graficamos los valores de los sucesivos x_n de la recurrencia vs. el valor exacto. Vemos que hasta la iteración 10 estos valores coinciden, pero luego el resultado de la recurrencia se dispara a 100.



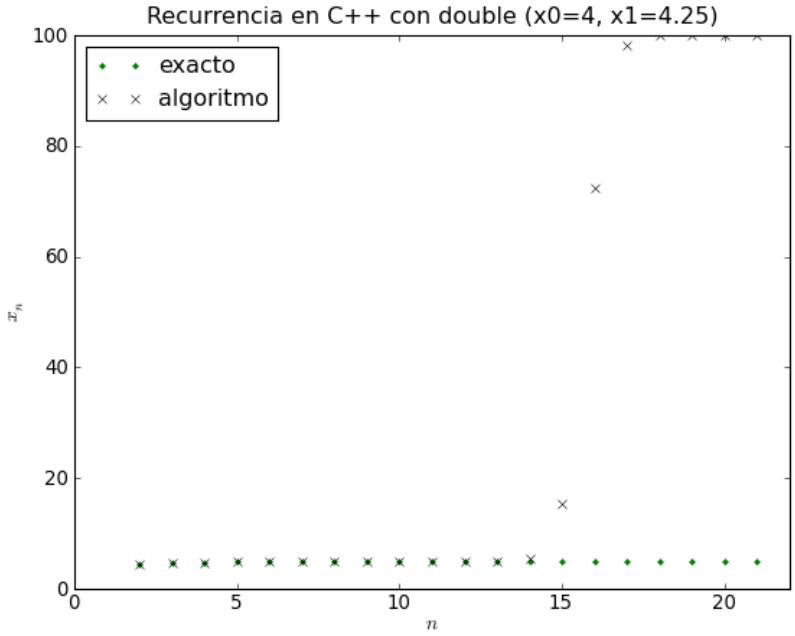
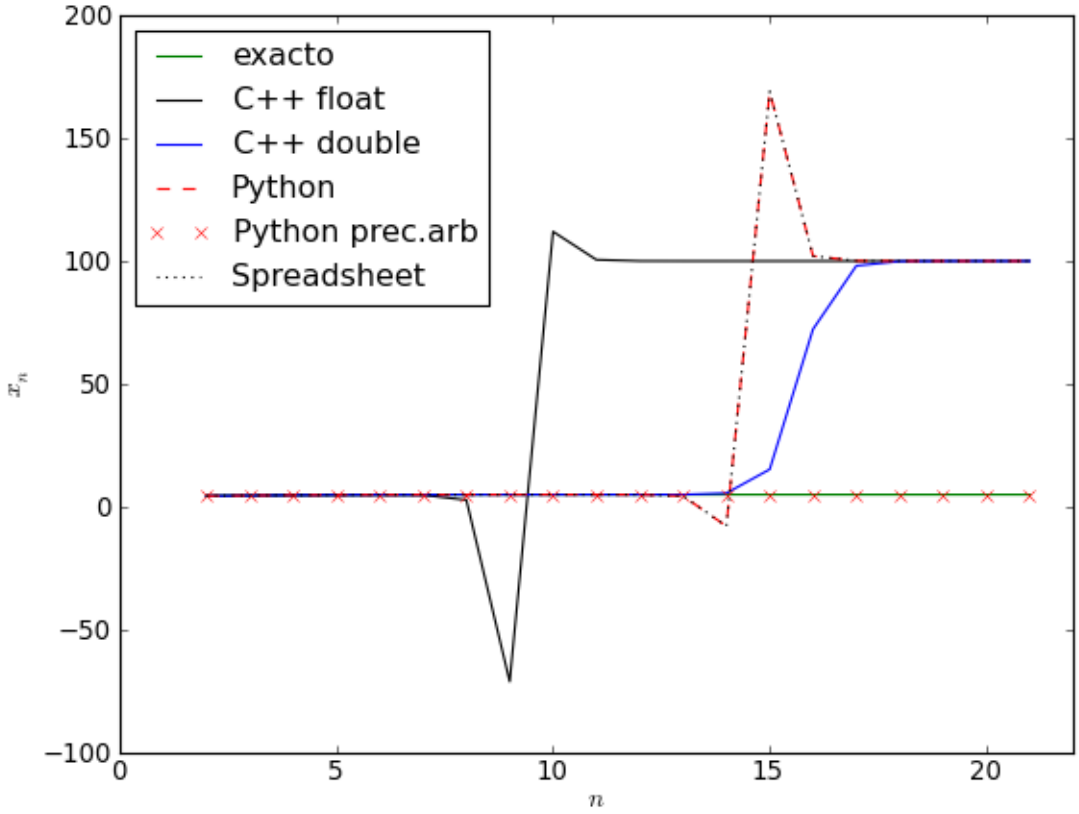
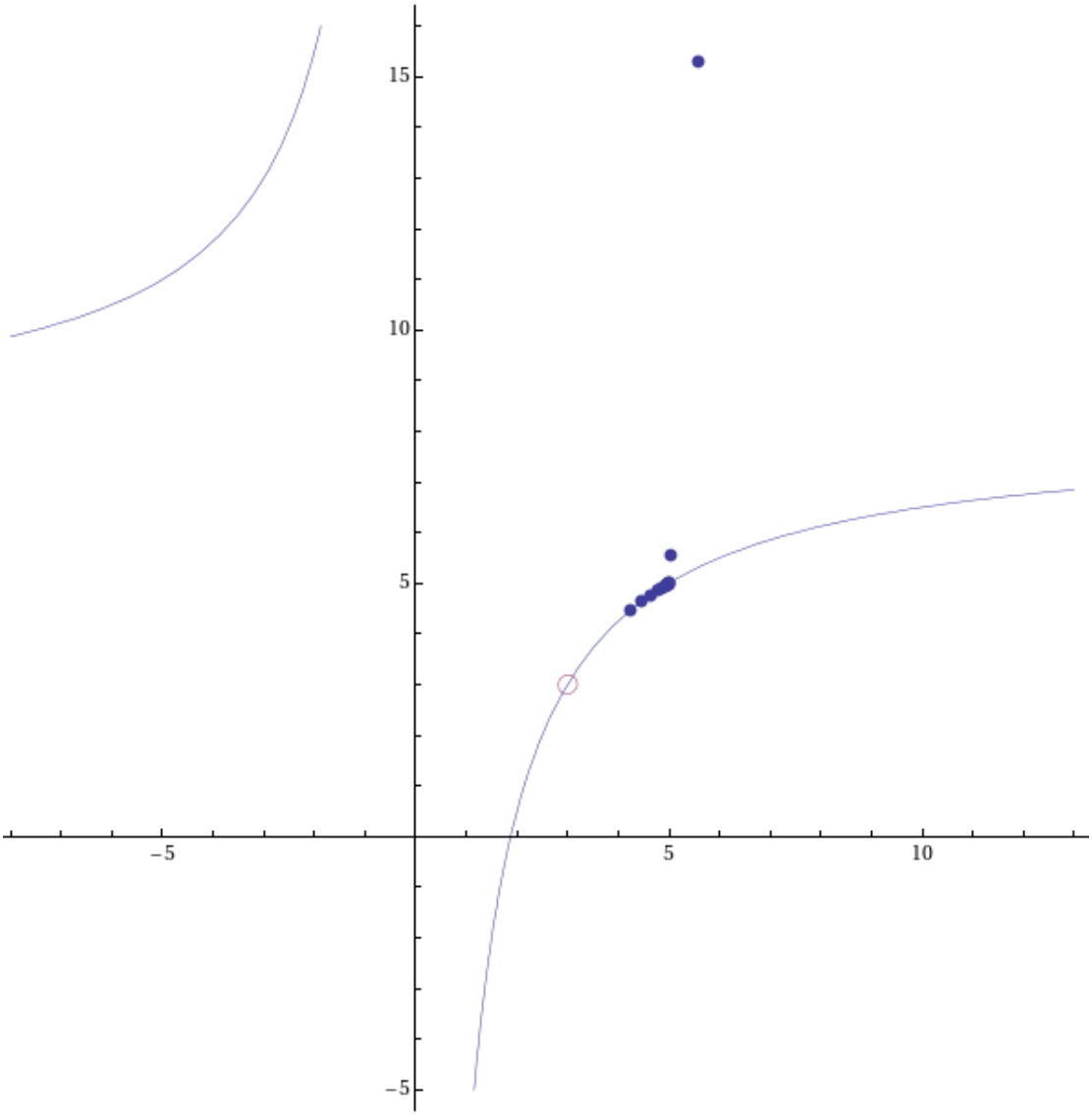


Figura 3: Comparación de lenguajes



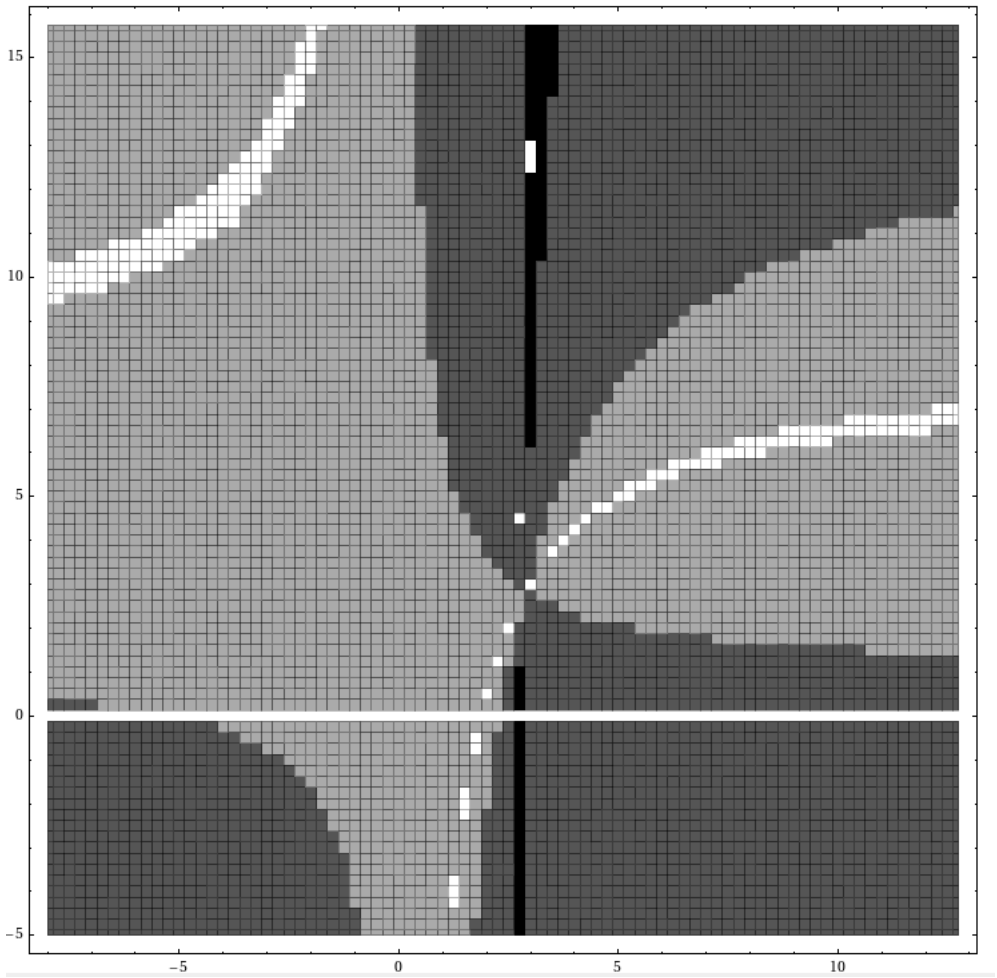
En estre gráfico se observan las sucesiones de los x_n de distintos lenguajes de programación/precisiones hasta la iteración 20.
Vemos, por ejemplo, que el Python con precision arbitraria coincide con el x_n exacto, que el float (precisión simple en C++) diverge mucho antes que todos los demas, y que el Python y el Spreadsheet se comportan exactamente igual.
(Obs: todos los valores son discretos. algunas secuencias las graficamos como líneas para que se aprecie mejor, no para interpolar los resultados)

Figura 4: Divergencia de la recurrencia para una instancia



Este gráfico ilustra los distintos puntos (x_k, x_{k+1}) partiendo de (x_0, x_1) =(4,4.25)
Se observa que tiende lentamente al punto $(5, 5)$, como es esperado, pero luego salta a (5,5.5) y luego a (5.5,15)
De ahi en más se va acercando al punto $(100, 100)$, que es el límite para todos los puntos fuera de la curva azul.

Figura 5: Iteraciones necesarias para llegar a límite 100



Acá pretendimos visualizar la cantidad de iteraciones que hay que hacer en `C++` (con precision double) para llegar cercano al límite 100, partiendo del punto $(x, y) = (x_0, y_0)$
Las zonas más oscuras corresponden a una menor cantidad de iteraciones, y las zonas más claras significan que son necesarias más iteraciones para llegar desde ese punto al límite 100. (Blanco significa 20 o más iteraciones)

Discusión

Estudiando un poco los datos obtenidos y conociendo los problemas de representación de números reales en los lenguajes utilizados, podemos dar algunas posibles explicaciones para los errores que vemos en los resultados.

En el caso de que los valores de entrada cumplan la condición $C(x0, x1)$ necesaria para converger a un número, podemos deducir que todos los términos de la recurrencia para esta entrada deben cumplir también con la condición $C(x_n, x_{n+1})$ ya que si tomamos $x_0 = x_n$ y $x_1 = x_{n+1}$ la recurrencia para estos valores de entrada también debe converger al mismo número.

Merece atención entonces notar en la figura 1, que casi todo el dominio corresponde a instancias tales que $\lim_{n\to\infty} x_n = 100$.

Por lo tanto, si x_0 y x_1 cumplen con la condición $x0(x1 - 8) = -15$, todos los pares de términos sucesivos de la recurrencia deben también caer en los puntos representados por la línea azul en el gráfico. Aqui vemos que con un pequeño error, un término puede fácilmente caer fuera de esta línea en el espacio que corresponde al límite 100, para luego continuar convergiendo a 100 erróneamente por la propiedad antes mencionada.

En cambio, la probabilidad de que un término de una instancia de la sucesión que deba converger a 100, caiga exactamente sobre la línea azul por un pequeño error , parece casi imposible, por lo que un error en el cálculo del límite para estas instancias no es de esperar.

En el caso de que los valores de entrada sean $x0 = x1 = 3$, la recurrencia llega a su límite de inmediato sin necesidad de calcular mas de un término, y dado que la probabilidad de que se cometa un error mayor que el criterio de parada en una sola cuenta es casi nulo (para un buen criterio de parada), la probabilidad de error para este caso de entrada también es muy baja.

Por esto es que notamos varios errores en el cálculo de límite para recurrencias que deben converger a 5, y que terminan convergiendo a 100, pero no en aquellas que convergen a 3 o 100.

Es importante también aclarar, que aunque pareciera que en el caso promedio casi no encontramos errores, ya que para casi toda la entrada el limite es 100, esto no necesariamente es cierto ya que no sabemos nada sobre la distribución de los datos de entrada. Estos podrían caer inicialmente en su mayoría en el caso en que $\lim_{n\to\infty} x_n = 5$ y podríamos estar cometiendo un error para la mayoría de los casos de entrada.

Además no sabemos el significado de riesgo que puede llegar a tener un error en estos casos, por lo que tampoco podemos dejar que el programa responda erróneamente a los mismos ya que por alguna razon el factor de riesgo podría ser muy grande para esta entrada, mas alla de que la probabilidad de que toque uno de estos casos sea muy baja.

Por lo tanto el cálculo del límite mediante la recurrencia, con este algoritmo y estas limitaciones en la aritmética puede llegar a ser incluso inútil.

En la figura 3 podemos observar como se comportan los distintos lenguajes para algunos casos de entrada particulares arbitrarios tal que $\lim_{n\to\infty} x_n = 5$, ya que no tiene mucho sentido analizar las recurrencias para $\lim_{n\to\infty} x_n = 3$ o 100, ya que vimos que en estos casos es muy fácil para el algoritmo converger correctamente.

Como podemos apreciar, el formato float de C hace que el programa "explote" mucho antes que el implementado con el tipo double, ya que este último tiene el doble de precision por lo cuál los errores que se generan son mucho mas chicos, y el error que se arrastra también.

Como podemos leer en la referencia de python, los numeros de punto flotante en este lenguaje son implementados con el tipo double en C, y como vemos en el gráfico, la similitud de las precisiones hace que ambos algoritmos "exploten" aproximadamente en el mismo lugar. Difieren un poco en comportamiento ya que el python puede compilar un algoritmo ligeramente distinto que el g++.

Sobre el *OpenOffice spreadsheet* no tenemos referencias sobre la precisión usada, pero vemos claramente que el comportamiento es exactamente el mismo que el de python, por lo que creemos que la aritmética es resuelta en python, ya que el programa además acepta macros en este lenguaje. Esto no necesariamente se traduce a todas las instancias del problema, pero podemos intuir al menos que la precisión que usa es la misma, o sea la *double precision IEEE-754*[2].

Por último tenemos la implementación del algoritmo en python pero usando una librería numerica llamada *decimal*[3], que nos permite elegir una precisión arbitraria para la representación en punto flotante. Vemos que con una precisión de 2000 dígitos significativos decimales, el X_{20} nos sigue dando sobre el límite exacto. Y calculamos que para esta entrada y esta precisión el algoritmo "explota" recien cerca del término X_{1400}. Así, podemos acotar el criterio de parada de la recurrencia con un epsilon arbitrariamente chico, eligiendo una precisión suficientemente grande. Esto sin embargo nun-

ca garantiza que el algoritmo a partir de un término "explote".

Otro aspecto importante a discutir es la elección del criterio de parada para los algoritmos que calculan los términos de la sucesión por recurrencia. si tomamos $|x_{n+1} - x_n| < \epsilon$ como criterio de parada, el valor de ϵ depende tanto de la estabilidad del algoritmo como del tipo de precisión aritmética usada.

Si el valor de ϵ es muy grande, probablemente x_n no este tan cerca del límite como quisieramos para tomarlo como tal. si al contrario lo hacemos muy chico, el cálculo de la recurrencia será mas prolongado y probablemente el error que lleva acumulado haga que la recurrencia salte a otro espacio de solución antes de llegar al límite correcto.

Un criterio de parada más adecuado sería tal vez cortar la recurrencia cuando suceda que $|x_{n+2} - x_{n+1}| > |x_{n+1} - x_n|$, o sea, cuando la sucesión comienza a alejarse del lugar adonde estaba convergiendo.

También se puede mejorar un poco este criterio aplicando una cota mínima para decidir si la recurrencia comienza a diverger, de manera de cuidarse de los casos en que la sucesion parezca diverger simplemente por un error aritmético.

Conclusiones

Analizamos el comportamiento de la recurrencia en distintos entornos, con distintas precisiones y con distintos parámetros. La principal conclusión que sacamos es que sin importar las condiciones bajo las cuales se realiza el experimento, la solución nunca es exacta. Esto se debe a que la representación de los números reales en la computadora es inexacta, porque el almacenamiento es finito.

Como es notado en las secciones anteriores, la recurrencia tiende a un límite distinto a 100 sólo en los puntos de la forma $(t, -15/t+8)$. Si calculamos uno a uno los términos de la recurrencia para estos puntos, observamos que todos los pares (x_k, x_{k+1}) tienen que ser también de la forma $(t, -15/t+8)$, (y son cada vez más cercanos al punto $(5, 5)$). Dada esta condición, vemos que el más mínimo error en alguno de los cálculos, y el siguiente par de valores cae afuera de la curva. Como la recurrencia involucra restas y divisiones, un error de representacion y/o de cálculo es esperable en prácticamente todos los casos.[1]

Incluso con precisión arbitraria de 2000 digitos decimales la sucesión tiene un error importantísimo para los x_k con $k \geq 1400$.

[1] Valores particulares como por ejemplo $(5, 5)$ no tienen error porque las operaciones de la recurrencia siempre dan enteros

Apéndices

Apéndice A: Enunciado

Laboratorio de Métodos Numéricos - Primer cuatrimestre 2010
Trabajo Práctico Número 1: "Perdidos" en el Pacífico

Dados dos números $x_0, x_1 \in \mathbb{R}$, la *recurrencia de Müller* es la sucesión $\{x_n\}_{n=0}^{\infty}$ definida por

$$x_{n+1} = 108 - \frac{815 - 1500/x_{n-1}}{x_n}, \qquad n \geq 1. \tag{1}$$

Esta sucesión tiende a un límite L, es decir, $\lim_{n\to\infty} x_n = L$ y se puede diseñar un algoritmo sencillo para determinar empíricamente este valor de L en función de los valores iniciales x_0 y x_1: computar x_n para $n \geq 1$ y detener el cómputo cuando $|x_n - x_{n-1}| < \epsilon$, donde $\epsilon \in \mathbb{R}_+$ es una tolerancia especificada de antemano. El objetivo del trabajo práctico es analizar la efectividad de este procedimiento.

Para realizar este análisis, es interesante observar que se puede encontrar una fórmula cerrada para x_n. Para esto, definimos $x_n = y_{n+1}/y_n$ y hacemos este cambio de variables en (1), obteniendo

$$y_{n+2} \;=\; 108y_{n+1} - 815y_n + 1500y_{n-1}$$

para $n \geq 1$. Esta recurrencia lineal se puede resolver por medio de su polinomio característico $p(z) = z^3 - 108z^2 + 815z - 1500 = (z-3)(z-5)(z-100)$, generando la fórmula cerrada

$$x_n \;=\; \frac{\alpha 3^{n+1} + \beta 5^{n+1} + \gamma 100^{n+1}}{\alpha 3^n + \beta 5^n + \gamma 100^n}$$

para $n \geq 0$, donde α, β y γ son constantes que se deben ajustar de acuerdo con los valores iniciales x_0 y x_1. Por ejemplo, $\alpha = \beta = 1$ y $\gamma = 0$ generan los valores iniciales $x_0 = 4$ y $x_1 = 4{,}25$. De esta forma se puede obtener analíticamente el límite exacto de la sucesión, para compararlo con el límite obtenido empíricamente.

El trabajo práctico consiste en implementar la recurrencia de Müller con aritmética de punto flotante sobre distintos entornos (por ejemplo, en distintos compiladores de C usando `float` y `double`, en planillas de cálculo, software matemático, etc.) para analizar el comportamiento del método empírico

en cada uno de estos entornos. Sobre la base de las implementaciones realizadas, se piden los siguientes experimentos obligatorios:

1. Graficar el límite exacto en función de x_0 y x_1. ¿Cómo se comporta este límite en función de los valores iniciales?

2. Comparar la aproximación del límite del método empírico contra el valor analítico obtenido en el experimento anterior. ¿Cómo se comporta este valor hallado experimentalmente? ¿Existen combinaciones de valores iniciales para los cuales el método empírico tenga problemas? En caso afirmativo, ¿se pueden caracterizar estos puntos y explicar por qué se producen estos efectos?

Los invitamos a que realicen todos los experimentos adicionales que sean necesarios para analizar y explicar convenientemente los efectos observados. El informe debe contener los resultados de todos los experimentos realizados, en un formato adecuado para su visualización y análisis.

Fecha de entrega: 16 de abril de 2010

Apéndice B: Codigos Fuente

Algoritmo para calcular x_n de forma exacta, en Python

```
def xn_exacto(x0,x1,n):
   # si gamma puede ser 0
   if abs((x1-8.)*x0 + 15.) < e:
      # elijo un b
      b = 1.
      # calculo a
      a = b*(5.-x0)/(x0-3.)
      return ( a*3.**(n+1) + b*5.**(n+1) ) / ( a*3.**n + b*5.**n )
   # si gamma != 0
   else:
      # calculo b/g
      bg = 50.*(600.-297.*x1+x0*(91.+2.*x1)) / (15.+x0*(x1-8.))
      #calculo a/g
      ag = (10000.-100.*x0-bg*(x0-5.)) / (x0-3.)
      return ( ag*3.**(n+1) + bg*5.**(n+1) + 100.**(n+1) ) /
             ( ag*3.**n + bg*5.**n + 100.**n )
```

Algoritmo que calcula la recurrencia, en Python con presicion arbitraria

```
from decimal import *
getcontext().prec=2000    #presicion: 2000 digitos
x0,x1=tuple(map(Decimal,sys.argv[1:3]))
n=int(sys.argv[3])
for i in range(n):
   #hacer todas las cuentas en presicion arbitraria
   temp = Decimal(108)-(Decimal(815)-(Decimal(1500)/x0))/x1
   x0 = x1
   x1 = temp
   print x1
```

Algoritmo que calcula la recurrencia, en C++

```
double x0, x1, temp, e;
e = 1e-3;
cin >> x0 >> x1;
do{
   temp = 108.-(815.-(1500./x0))/x1;
   x0 = x1;
   x1 = temp;
}while( abs(x1-x0) >= e );
```

Función usada en la Hoja de cálculo para calcular los términos de la recurrencia

```
=IF(ABS(B2-C2)<$B$1,C2,108-(815-(1500/B2))/C2)
```

(`B1` es la celda donde esta almacenado ϵ, `B2` y `C2` son x_{n-2} y x_{n-1} respectivamente)

Referencias

1. `http://docs.python.org/tutorial/floatingpoint.html`
   Explicación de la aritmética de punto flotante en Python

2. `http://steve.hollasch.net/cgindex/coding/ieeefloat.html`
   Estándard de representación de punto flotante más utilizado

3. `http://docs.python.org/library/decimal.html`
   Documentación de la libreria `decimal` en Python, para presicion arbitraria