IIP (E.T.S. d'Enginyeria Informàtica)
Curs 2019-2020

Pràctica 6. Iteració: realització d'una classe d'utilitats

Duració: dues sessions

Professors d'IIP
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Índex

1 Objectius i treball previ a la sessió de pràctiques

En algunes ocasions el processador amb el que es treballa no té predefinides funcions matemàtiques d'ús habitual. En aquest cas, no disposem d'operacions trigonomètriques o logarítmiques, que són moltes vegades necessàries per a càlculs tan habituals com els de posició o de desplaçament en petits elements robòtics, drons, etc. Això és molt freqüent quan els processadors dels que es disposa són de poca potència; com passa, per exemple, en molts microcontroladors (dispositius corrents en teclats, ratolins, mòbils de poca potència, etc.). Quan això passa, cal implementar les nostres pròpies funcions matemàtiques quan ens siguen necessàries.

Prenent com a exemple l'anterior, en aquesta pràctica resoldràs alguns problemes numèrics per als quals és necessari fer ús d'estructures iteratives. En concret, realitzaràs una classe d'utilitats que contindrà mètodes per al càlcul aproximat de dues funcions conegudes: l'arrel quadrada i el logaritme d'un valor.

Després d'haver-les implementat, hauràs de comprovar la seua correcció comparant-les amb les predefinides en el llenguatge Java. Finalment, hauràs de representar gràficament les dues funcions utilitzant la llibreria que et vam proporcionar a la pràctica 5.

Per realitzar adequadament la pràctica és convenient que hages estudiat prèviament l'exemple 8.6 i el problema 21 del capítol 8 del llibre "Empezar a programar usando Java" (3ª edició)[1]. Entendre la solució de tots dos et facilitarà la resolució de les activitats proposades.

Activitat inicial

Has de crear un nou paquet `pract6`, corresponent a aquesta pràctica, en el projecte `iip` que engloba les pràctiques de l'assignatura. A continuació, inclou en aquest les classes que se't donen (algunes parcialment resoltes) com material d'ajuda. Veuràs que hi ha una classe `IIPMath` el propòsit de la qual és que inclogues en ella alguns mètodes per calcular les funcions proposades a continuació.

2 Problema 1: Arrel quadrada

La següent recurrència permet calcular l'arrel quadrada, t, de cert valor no negatiu x:

$$t_1 = \frac{1+x}{2}, \qquad t_{i+1} = \frac{t_i + x/t_i}{2}$$

La recurrència consisteix en una aproximació successiva, $t_1 \ldots t_n$, al valor desitjat (arrel quadrada de x), on el darrer terme, t_n, és el més pròxim a l'arrel de x d'entre tots els termes, t_i, generats.

2.1 Precisió dels càlculs. Condició de terminació

És possible provar que la successió de termes $t_1 \ldots t_n \ldots$, s'aproxima estrictament a l'arrel del nombre x, de manera que l'error en el càlcul decreix estrictament amb cada nou terme calculat.

Donats dos termes consecutius t_{i-1} i t_i $(i > 1)$ es pot considerar la seua diferència com una mesura de l'error que es comet quan s'ha calculat el terme i-èsim. Per tant, si vols fer el càlcul de l'arrel quadrada amb un cert error màxim, tindràs prou en acabar el procés quan la diferència entre dos termes consecutius siga menor que l'esmentat error[2].

Seguint això, es pot establir un criteri de terminació de la iteració: quan la diferència entre dos termes consecutius siga menor que l'error desitjat.

Activitat #1

a) Tenint en compte la recurrència anterior, implementa en la classe `IIPMath` un mètode públic per calcular l'arrel quadrada de cert valor x, no negatiu, amb un error màxim ϵ, seguint per a això el perfil:

```
/** Torna l'arrel quadrada de x >= 0, amb error epsilon > 0. */
public static double sqrt(double x, double epsilon)
```

b) Implementa en la classe `IIPMath`, fent ús del mètode anterior, un altre mètode públic que sobrecarregue al primer, per a calcular l'arrel quadrada de cert valor x, no negatiu, amb un error màxim `1e-15`, seguint el perfil:

```
/** Torna l'arrel quadrada de x >= 0, amb error 1e-15. */
public static double sqrt(double x)
```

c) Documenta adequadament cadascun dels mètodes. Per això, a més del comentari en què es destaque el que realitza el mètode, hauràs d'escriure també la descripció dels paràmetres, així com del seu resultat (usant, respectivament, el *tag* `@param` per descriure els paràmetres i el `@return` per al resultat). Com a exemple, el mètode anterior, es podria documentar de la manera següent:

```
/** Torna l'arrel quadrada de x >= 0, amb error 1e-15.
 *  @param x. El valor, que ha de ser igual o major que zero.
 *  @return double. L'arrel de x amb error màxim 1e-15.
 */
public static double sqrt(double x)
```

[1] Exemple 9.6, problema 24 en la 2ª edició.

[2] Tingues en compte que l'error amb el qual pugues realitzar el càlcul estarà limitat per la precisió, o nombre de dígits amb què treballes. Un valor `double` en Java té una precisió d'uns 16 dígits.

d) Comprova que el codi realitzat és correcte, utilitzant l'avaluador d'expressions del *BlueJ* (*Code Pad*) per a comparar la funció arrel que has fet: `IIPMath.sqrt(double)`, amb la que proporciona el llenguatge Java: `Math.sqrt(double)`.

3 Problema 2: Logaritme d'un valor

Per calcular el logaritme natural d'un valor qualsevol $x \in R^+$, se sol utilitzar en primer lloc el següent desenvolupament en sèrie que permet calcular el logaritme de cert z, amb $1/2 \leq z < 1$. Siga:

$$y = \frac{1-z}{1+z} \tag{1}$$

llavors es coneix que:

$$log(z) = -2\sum_{i=1}^{\infty} \frac{y^{2i-1}}{2i-1} \tag{2}$$

Si es representa el terme i-èsim ($1 \leq i$) del desenvolupament de la suma anterior per u_i, llavors:

$$log(z) = -2(u_1 + u_2 + u_3 + \cdots + u_k + u_{k+1} + \cdots + u_n) - R_n$$

és a dir, tota la sèrie pot representar-se com suma de termes u_i, juntament amb un residu R_n, que representa la suma dels termes restants, posteriors al n-èsim. O el que és el mateix:

$$log(z) = -2\sum_{i=1}^{n}(u_i) - R_n$$

3.1 Càlcul dels termes

El mètode que construïsques ha de calcular cadascun dels termes u_i de la expressió anterior. Substituint en (2), es pot obtenir el primer terme del sumatori, u_1, que val y. També substituint en (2), s'obté per als dos termes consecutius qualssevols u_k i u_{k+1}, les següents expressions:

$$u_k = \frac{y^{2k-1}}{2k-1} \qquad u_{k+1} = \frac{y^{2k+1}}{2k+1}$$

A partir d'aquestes es pot observar que es compleix la següent relació entre dos termes consecutius qualssevols:

$$u_{k+1} = y^2\frac{2k-1}{2k+1}u_k \tag{3}$$

Com pots veure, és possible estalviar càlculs si cada nou terme s'obté a partir l'immediatament anterior, en lloc de calcular-ho de forma independent.

3.2 Precisió dels càlculs. Condició de terminació

Per característiques pròpies de la sèrie es pot demostrar que l'error total que es comet (R_n), és sempre menor que el valor de l'últim terme calculat (u_n).

Per això, si vols que l'error comès siga menor que cert ϵ, n'hi ha prou amb que calcules un terme rere l'altre (sumant-los) fins que arribes a un amb valor inferior a l'esmentat ϵ[3].

Activitat #2

Tenint en compte la recurrència en (3), implementa en la classe `IIPMath` un mètode públic per calcular el logaritme de cert valor z, $1/2 \leq z < 1$, amb un error màxim ϵ, seguint per a això el perfil:

```
/** Torna log(z), 1/2 <= z < 1, amb un error epsilon > 0. */
public static double logBase(double z, double epsilon)
```

[3] Un cop més, tingues en compte que l'error amb el que pugues realitzar el càlcul estarà limitat per la precisió, o nombre de dígits amb els que treballes.

3.3 Càlcul general del logaritme d'un valor

Conegut el càlcul anterior (que permet determinar el logaritme d'un valor en el interval $[1/2, 1[$), vegem com s'aplica per calcular el logaritme de qualsevol valor $x \in R^+$.

Donat un valor x, no negatiu qualsevol, és possible transformar-lo en un valor z en l'interval $[1/2, 1[$ bé dividint-lo tantes vegades com siga necessari per 2 (quan el valor original de x siga major o igual a 1), bé multiplicant-lo totes les vegades necessàries per 2 (quan el valor original de x siga menor que 1/2).

És a dir, es tindrà que x pot expressar-se com:

$$x = 2^m z \qquad (1/2 \leq z < 1) \tag{4}$$

essent m un valor enter positiu o negatiu. A més, després d'aplicar logaritmes a les dues parts de l'equació anterior, es té:

$$log(x) = mlog(2) + log(z) \tag{5}$$

Permetent realitzar el càlcul desitjat que en resum consistirà, per a cert x, en:

1. Si $x \in [1/2, 1[$ aplicar directament el mètode `logBase(double, double)`.
2. Si $x \notin [1/2, 1[$, llavors:
   (a) Si $x \geq 1$, atès que perque es compleixca l'equació (4) es dóna que $z = x/2^m$; calcular z i m dividint x per 2 el nombre de vegades necessari (m vegades) per a reduir-lo a un valor en $[1/2, 1[$ (aquest valor reduït serà el valor de z).
   (b) Si $x < 1/2$, pot calcular-se z i m mitjançant productes successius de x per 2 en lloc de fer servir divisions. També pot utilitzar-se que $-log(1/x) = log(x)$. Per exemple, si es desitja calcular $log(0.2)$ $(0.2 < 1/2)$ es té que $-log(5) = log(0.2)$. I, per a calcular $log(5)$ s'està en les condicions del cas anterior.
3. Aplicar l'equació (5) per, donat $log(z)$, calcular el logaritme desitjat. Cal tenir en compte que $log(2)$ és un valor conegut, l'aproximació figura en l'enunciat de l'activitat següent:

Activitat #3

a) Defineix el valor aproximat de $log(2)$ (0.6931471805599453) com una constant en el teu programa.

b) Tenint en compte les expressions en (4) i (5), implementa en la classe `IIPMath` un mètode públic per calcular el logaritme de cert valor x, positiu, amb un error màxim ϵ, segons:

```
/** Torna log(x), x > 0, amb error epsilon > 0. */
public static double log(double x, double epsilon)
```

c) Implementa a la classe `IIPMath`, fent ús del mètode anterior, un altre mètode públic que sobrecarregue el d'abans, per calcular el logaritme de cert x, positiu, amb un error màxim 1e-15, segons:

```
/** Torna log(x), x > 0, amb error 1e-15. */
public static double log(double x)
```

d) Igual que abans, documenta adequadament cadascun dels mètodes, destacant el que fan. Usa, els *tags* `@param` i `@return` per a descriure els seus paràmetres i resultat.

4 Prova de les teues funcions arrel i logaritme

Pots comparar les funcions que has realitzat amb les que proporciona el llenguatge Java en la seua classe estàndard `Math` per a, d'aquesta manera, comprovar la precisió dels teus resultats.

Per facilitar aquesta tasca, se't proporciona la classe `IIPMathTest` que has descarregat en el paquet `pract6`, i que et permet comparar per a un `x` donat, introduït per tu, el resultat de calcular la seua arrel i el seu logaritme utilitzant els mètodes que has fet (`IIPMath.sqrt(double)` i `IIPMath.log(double)`) amb els dos definits en Java (`Math.sqrt(double)` i `Math.log(double)`).

Activitat #4

Executa la classe `IIPMathTest` per comprovar que els resultats de les funcions que has realitzat són correctes (comparables als de la versió del Java estàndard) amb diferents valors, tant petits (per exemple, de l'ordre de $10^{-12}, 10^{-6}, 10^{-3}$), grans (de l'ordre de $10^3, 10^6, 10^{12}$), com valors en el rang de les unitats i de les desenes. Per descomptat, pots comprovar els resultats per qualsevol altre valor en el rang que consideres.

5 Representació gràfica de les funcions

Per poder mostrar gràficament funcions com les que has implementat, vas a usar la llibreria gràfica (classe `Graph2D` del paquet `graph2D`) que vas instal·lar a la pràctica 5. Recorda que al fitxer `Graph2D.html` pots consultar la documentació d'aquesta llibreria, mitjançant la qual es poden fer representacions gràfiques de punts i línies, entre altres elements, en un espai bidimensional.

En la figura 1, tens una mostra d'aquesta representació feta mitjançant el dibuix punt a punt de dues funcions: $sin(x)$ i $sin(x)/x$ en l'interval $[-1, 4\pi]$.

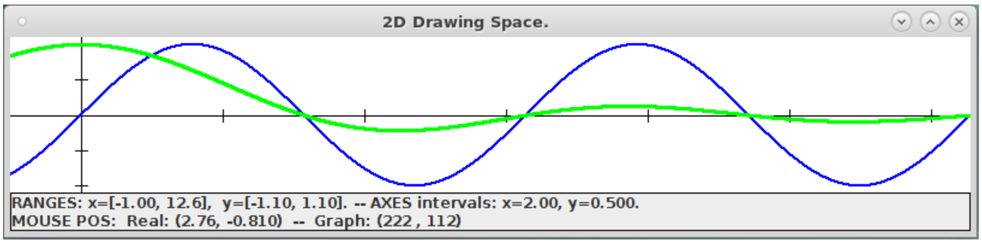

Figura 1: Representació punt a punt de $sin(x)$ i $sin(x)/x$ en $[-1, 4\pi]$.

Un dibuix punt a punt dels valors de certa funció f, s'obté calculant per a diferents valors possibles de x (en l'interval de representació que es desitge) els valors $f(x)$ corresponents i representant gràficament el parell $(x, f(x))$ mitjançant la primitiva de representació gràfica corresponent (mètode `drawPoint(double, double)` de la classe `Graph2D`).

Se't proporciona un exemple d'ús de `graph2D.Graph2D`, al fitxer `Graph2DTest.java` que hauràs inclòs en el teu paquet `pract6` i que en executar-se mostrarà una finestra similar a la de la figura 1. Pots basar-te en ell per resoldre les activitats que se't plantegen a continuació.

Addicionalment, en l'annex, al final d'aquest butlletí, es descriu com representar gràficament una de les dues funcions ($sin(x)/x$) de manera similar a com apareix en `Graph2DTest.java` i es representa a la figura 1.

Activitat #5

Representa gràficament en l'interval: $x \in [-1, 15]$ i $y \in [-3, 4]$, usant punts, els valors de les funcions que has desenvolupat (arrel i logaritme). A la figura 2 tens una mostra del resultat corresponent a aquesta activitat.

Activitat #6

Representa gràficament en l'interval: $x \in [-1, 15]$ i $y \in [-3, 4]$, mitjançant línies, és a dir, unint cada dos punts consecutius de la gràfica mitjançant una línia, els valors de les dues funcions que has desenvolupat.

NOTA: En ambdues activitats has de tenir en compte que l'arrel no està definida en $[-\infty, 0[$ i que el logaritme no està definit en $[-\infty, 0]$.

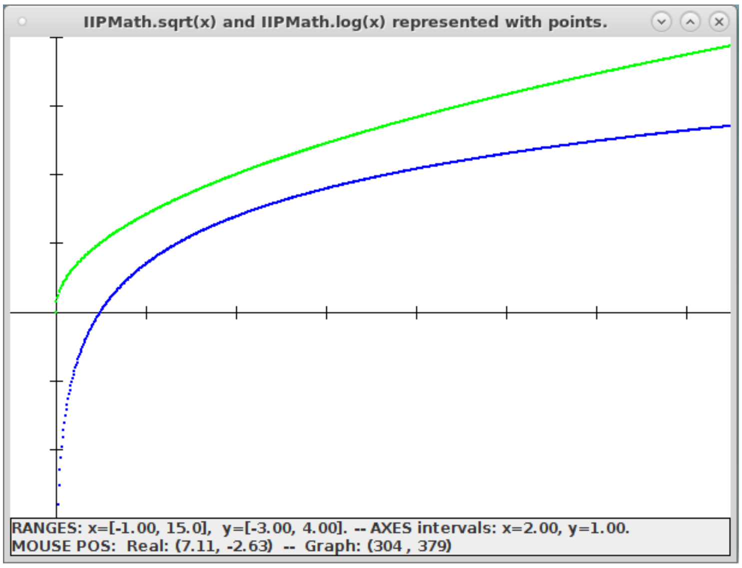

Figura 2: Representació de IIPMath.sqrt(x) i IIPMath.log(x) en $[-1, 15]$.

6 Annex: Exemple d'ús de la classe Graph2D

Com a exemple, es mostra l'ús de Graph2D per representar la funció $sin(x)/x$, com s'ha vist abans en la figura 1.

En primer lloc, és necessari incloure la directiva d'importació de la classe gràfica al començament del fitxer en el que s'escriga el programa:

```
// Importa la classe Graph2D (del paquet graph2D).
import graph2D.Graph2D;
```

Un cop fet això, ja és possible definir objectes d'aquesta classe i operar sobre ells en la classe que desenvolupem. Per simplicitat, fem servir el constructor amb menor nombre d'arguments: valors mínims i màxims possibles de x i de y. Per facilitar el seu ús posterior, definim prèviament variables amb aquests valors mínims i màxims, és a dir:

```
// Definir l'interval de valors per a x i per a y
double xMin = -1;
double xMax = Math.PI * 4;
double yMin = -1.1;
double yMax = +1.1;
// Crear l'espai de dibuix amb les dimensions desitjades
Graph2D gd1 = new Graph2D(xMin, xMax, yMin, yMax);
// Canviar la grossaria dels elements a 3 (per defecte es 1)
gd1.setThickness(3);
// Canviar el color a GREEN per dibuixar d'ara endavant
gd1.setForegroundColor(Color.GREEN);
```

A continuació, recórrer cada x possible (en el seu interval de definició) representant gràficament el punt (x, Math.sin(x) / x). Per a això, incrementar poc a poc cada valor de x des del seu valor inicial fins al final. Fem servir una variable (delta) per mantenir el valor d'aquest increment:

```
// Calcular l'increment en cada pas de x (delta)
double delta = (xMax - xMin) / Graph2D.INI_WIDTH;
```

on la constant Graph2D.INI_WIDTH representa el nombre de punts existents inicialment en l'eix d'abscisses a la finestra gràfica (800).

Finalment, recórrer cada `x` possible, calculant per a aquest valor la funció corresponent i representant gràficament aquest parell:

```
// Recorrer cada punt en x, calcular f(x) i dibuixar (x, f(x))
for (double x = xMin; x <= xMax; x = x + delta) {
    double y = Math.sin(x) / x;     // y es el valor de la funcio
    gd1.drawPoint(x, y);            // representar (x, y)
}
```

El resultat de l'execució del codi anterior es mostra, a continuació, en la figura 3.

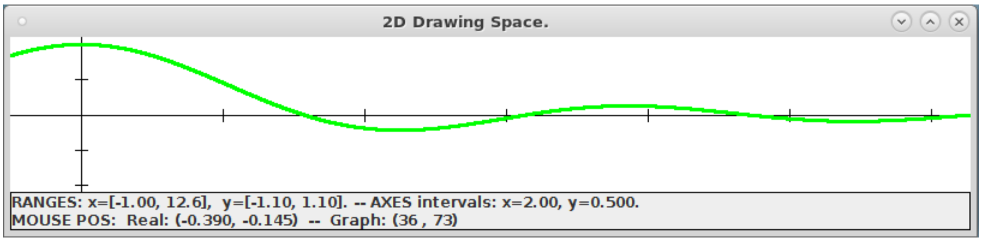

Figura 3: Representació de $sin(x)/x$ en $[-1, 4\pi]$.