Universitat Autònoma de Barcelona, Departament de Física

Mètodes numèrics

Alfredo Hernández Cavieres

2014-2015

Índex

1 Resolució d'equacions no lineals

L'objectiu és trobar $\{x \in \mathbb{R} \mid f(x) = 0\}$, on $f : \mathbb{R} \to \mathbb{R}$. Cal tenir en compte que potser no hi ha solució o bé pot ser que sí que n'hi hagi però el «mètode» seguit no sigui vàlid per torbar-la.

1.1 Mètode zero

Teorema 1.1. *Sigui* $g(x) \in \mathbb{R}$ *contínua en* $[a, b]$. *Si* $\{\exists L < 1 \mid |g(x) - g(y)| \leq L\,|x - y|\}, \forall x, y \in [a, b]$, *llavors:*

i) $\exists!$ *solució a l'equació* $x = g(x)$ *en* $[a, b]$.

ii) La solució és $x = \lim_{i\to x}\{x_i\}$, *amb*

$$\boxed{x_{i+1} = g(x_i)} \tag{1.1}$$

on x_0 *és qualsevol* $x \in [a, b]$.

□

Exemple 1.1. Volem resoldre $\ln x = 1$. Analíticament sabem que la solució és $x = e$ ($\approx 2.718\,28$).

i) Estudiem l'equació $g(x) = 1 - x - \ln x = x$.

ii) Intuïm que la solució és $0 < x < 10$.

iii) $g(x) - g(y) \cdots = \left|1 + \frac{\ln(y/x)}{x-y}\right| < 1 \Rightarrow \exists!$ solució $\in (0, 10)$.

Fent moltes iteracions arribem a $x_{15} = 2.717\,346\ldots$, és a dir hem obtingut 3 xifres significatives. Com es pot veure, aquest mètode té un error que decau molt lentament. ▲

1.2 Mètode de Newton–Raphson

Sigui $f(x)$ una funció de la qual volem trobar els punts on $f(x) = 0$. El mètode consisteix en agafar un punt inicial x_0 i trobar la recta tangent al punt: $y = f'(x_0)(x - x_0) + f(x_0)$ i igualant-la a zero. És fàcil veure que la següent recurrència ens porta a la solució $x = \lim_{i\to x}\{x_i\}$:

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \tag{1.2}$$

Ara bé, no sempre és possible trobar una solució $\forall x_0 \in \mathbb{R}$.

Teorema 1.2. *Es pot demostrar que* $\exists!$ *arrel de* $f(x) = 0$ *en* $[a, b]$ *si es compleixen les següents condicions:*

1. $f(x) \in C^1_{[a,b]}$.

2. $f'(x) \neq 0, \quad \forall x \in [a,b]$.
3. $f''(x) \neq 0, \quad \forall x \in [a,b]$.
4. $f(a)f(b) < 0$.
5. $\max\left\{\left|\frac{f(a)}{f'(a)}\right|, \left|\frac{f(b)}{f'(b)}\right|\right\} < b-a$.

□

Exemple 1.2. Volem trobar $\sqrt{3}$ ($\approx 1.732\,050\,807\,5$). Llavors estudiarem l'equació $f(x) = x^2 - 3 = 0$. La seva derivada és $f'(x) = 2x$.

i) $f(x) \in C^\infty_{\mathbb{R}}$, en particular agafem l'interval $[-1, 4]$.

ii) $f''(x)$ no canvia de signe (és sempre positiva).

iii) $f(-1)f(4) = -26 < 0$.

iv) $\max\left\{\left|\frac{-3}{-2}\right|, \left|\frac{16}{8}\right|\right\} < 5$.

Agafant $x_0 = 2$ arribem a $x_3 = 1.732\,050\,8\ldots$, és a dir, amb tres iteracions arribem a un resultat exacte en 8 xifres decimals.

```
for(i = 0; i < N; ++i) {
  x[i+1] = x[i] - (x[i]**2 - 3)/(2*x[i]));
  if (fabsl(x[i+1]-x[i]) < pow(10, -tolerance)) {
    N = i+1; // Es redefineix per loops posteriors
    break;
  }
}
```

▲

1.3 Mètode de regula falsi

El mètode consisteix en fer la següent iteració, on a_i i b_i són variables.

$$c_i = \frac{f(b_i)a_i - f(a_i)b_i}{f(b_i) - f(a_i)} \tag{1.3}$$

En particular, els intervals $[a_i, b_i]$ venen definits de la següent manera:

- Si $f(c_i)f(a_i) < 0 \Rightarrow b_{i+1} = c_i$ i $a_{i+1} = a_i$.
- Si $f(c_i)f(b_i) < 0 \Rightarrow a_{i+1} = c_i$ i $b_{i+1} = b_i$.

És a dir, volem intervals on sempre es pugui aplicar el teorema de Bolzano.

Exemple 1.3. Volem trobar $\sqrt{3}$ ($\approx 1.732\,050\,807\,5$). Llavors estudiarem l'equació $f(x) = x^2 - 3 = 0$ a l'interval $[1, 2]$. Agafant aquest interval, arribem a $c_5 = 3691/2131 = 1.732\,050\,680$, és a dir, amb 6 iteracions (la primera és la iteració que genera c_0) arribrem a un resultat exacte en 6 xifres decimals.

```
for (i = 0; i < N; ++i) {
  if (!((c[i]<a[i]) || (c[i]>b[i])) && \
      (((a[i]**2 - 3)*(c[i]**2 - 3))) < 0 )) {
    b[i] = c[i];
    a[i+1] = a[i];
    c[i+1] = ((b[i]**2 - 3)*a[i] - (a[i]**2 - 3)*b[i]) \
             /(b[i]**2 - a[i]**2 - 4);
    if (fabsl(c[i+1]-c[i]) < pow(10, -tolerance))
      {
        N = i+1; // Es redefineix per loops posteriors
        break;
      }
  }
  else if (!((c[i]<a[i]) || (c[i]>b[i])) && \
           !(((a[i]**2 - 3)*(c[i]**2 - 3))) < 0 )) {
    a[i] = c[i];
    b[i+1] = b[i];
    c[i+1] = ((b[i]**2 - 3)*a[i] - (a[i]**2 - 3)*b[i]) \
             /(b[i]**2 - a[i]**2 - 4);
    if (fabsl(c[i+1]-c[i]) < pow(10, -tolerance))
    {
      N = i+1; // Es redefineix per loops posteriors
      break;
    }
  }
}
```

▲

Si bé el mètode de regula falsi és relativament més complicat de programar que el de Newton–Raphson, la seva interpretació geomètrica és molt senzilla i no requereix tantes condicions de validesa, ja que només és necessari un interval $[a, b]$ on $f(x)$ sigui contínua i que es pugui aplicar el teorema de Bolzano.

No només això, sinó que, com hem pogut veure als exemples 1.2 i 1.3, el mètode de regula falsi és considerablement més lent que el de Newton–Raphson.

1.4 Resolució de sistemes d'equacions no lineals

Sigui $\vec{f}(\vec{x}) = \vec{0}$ un sistema d'equacions. El mètode emprat per trobar solucions és una generalització del mètode de Newton–Raphson:

$$\boxed{\vec{x}_{i+1} = \vec{x}_i - \left(\vec{\mathcal{J}}^{\vec{f}}_{\vec{x}_i}\right)^{-1} \vec{f}(\vec{x}_i)} \tag{1.4}$$

on $\vec{\mathcal{J}}^{\vec{f}}_{\vec{x}_i}$ és la matriu jacobiana de $\vec{f}(\vec{x})$ avaluada a $\vec{x}_i$. Anàlogament, s'ha de complir la següent condició per l'existència de la solució:

i) $\det \vec{\mathcal{J}}^{\vec{f}}_{\vec{x}_i} \neq 0 \Leftrightarrow \exists \left(\vec{\mathcal{J}}^{\vec{f}}_{\vec{x}_i}\right)^{-1}$.

Exemple 1.4. Sigui $\begin{cases} x^2+y^2-1=0 \\ x^2-y^2+1/2=0 \end{cases}$. La construcció de la recurrència seria la següent:

$$\begin{aligned}\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} &= \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} 2x_i & 2y_i \\ 2x_i & -2y_i \end{pmatrix}^{-1} \begin{pmatrix} x^2+y^2-1 \\ x^2-y^2+1/2 \end{pmatrix} \\ &= \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} \dfrac{2x_i^2 - y_i^2 + 1/y_i + 1/2}{4x_i} \\ \dfrac{2x_i^2 - y_i^2 + 1/y_i + 1/2}{4y_i} \end{pmatrix} = \frac{1}{8} \begin{pmatrix} \dfrac{4x_i^2+1}{x_i} \\ \dfrac{4y_i^2+3}{y_i} \end{pmatrix}\end{aligned}$$

Amb $x_0 = 1$, $y_0 = 1$, arribem a $x_3 = \dfrac{3281}{6560} \approx 0.500\,152\,439$, $y_3 = \dfrac{18817}{21728} \approx 0.866\,025\,405$, que és una de les quatre solucions amb 3 xifres exactes a x i 8 xifres significatives a y.

```
1 for(i = 0; i < N; ++i) {
2   x[i+1] = (4 * x[i]**2 + 1)/(8*x[i]);
3   y[i+1] = (4 * y[i]**2 + 3)/(8*y[i]);
4   if ((fabsl(x[i+1]-x[i]) < pow(10, -tolerance)) && \
5       (fabsl(y[i+1]-y[i]) < pow(10, -tolerance))) {
6     N = i+1; // Es redefineix per loops posteriors
7     break;
8   }
9 }
```

▲

1.5 Estimació dels errors

Per a mètodes iteratius es pot fitar l'error e comès entre cada iteració. Sigui $\bar{x} \equiv$ la solució exacta de $f(x) = 0$ i $x_n \equiv$ valor aproximat de la $n-$èsima iteració. Llavors tenim

$$\begin{aligned} e_{n+1} = \bar{x} - x_{n+1} &= \bar{x} - x_n + \frac{f(x_n)}{f'(x_n)} \quad \text{(Newton–Raphson)} \\ f(\bar{x}) = f(x_n + e_n) &\approx f(x_n) + f'(x_n)e_n + f''(\zeta)e_n^2 = 0 \quad \text{(Taylor)} \\ \Rightarrow \boxed{e_{n+1} = \frac{|f''(\zeta)|}{|2f'(x_n)|} e_n^2 \leq k e_n^2}, &\quad k \in \mathbb{R} \end{aligned}$$

on $\zeta \in (\bar{x}, x_n)$ o bé $\zeta \in (x_n, \bar{x})$ segons la situació, i per tant és un valor fitat.

Estimar l'error en un programa

Una de les coses que volem controlar en un programa és el temps de càlcul. Com ens podem assegurar que passat x temps el nostre programa ha trobat la solució amb l'exactitud que volem? Per controlar el càlcul podem emprar les següents estratègies:

i) Comparar x_i amb valors anteriors: l'objectiu és parar de calcular quan el resultat ha arribat a una tolerància (i.e., xifres exactes) predeterminada, és a dir, que pari si $|x_i - x_{i-1}| < \varepsilon$.

ii) Predeterminar un nombre màxim d'iteracions: l'objectiu és evitar que el programa es quedi calculant infinitament. El nombre màxim d'iteracions no sempre és trivial d'establir de manera òptima, però mentre major sigui, més exacte hauria de ser el resultat.

Exemple 1.5. A la línia 1 establim el nombre màxim d'iteracions `N`. A la línia 4 establim $\varepsilon = 10^{-\texttt{tolerance}}$.

```
1 for(i = 0; i < N; ++i) {
2     [...]
3     if (fabsl(x[i+1]-x[i]) < pow(10, -tolerance)) {
4         printf("S'ha arribat a la tolerància desitjada.\n");
5         break;
6     }
7     [...]
8 }
```

▲

2 Derivació i integració numèrica

2.1 Derivació numèrica

Si bé la derivació numèrica és senzilla, s'hauria d'evitar sempre que sigui possible, ja que introdueix molt error numèric.

Primera derivada

$$\boxed{f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h} = \frac{\Delta f}{\Delta x}} \tag{2.1}$$

on h és la llargada de cada pas de la discretització del domini de la funció $f(x)$.

Observem que si s'avalua la funció als extrems del domini la fórmula anterior falla. No és difícil veure que les fórmules equivalents són les següents:

$$\text{Extrem esquerre: } f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$$
$$\text{Extrem dret: } f'(x_0) \approx \frac{f(x_0)-f(x_0-h)}{h}$$

Segona derivada

A través de l'expansió per Taylor podem arribar fàcilment a una expressió per a la segona derivada:

$$f(x_0+h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + O(h^3)$$
$$f(x_0-h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + O(h^3)$$

$$\Rightarrow \boxed{f''(x_0) = \frac{f(x_0+h)+f(x_0-h)-2f(x_0)}{h^2}} \tag{2.2}$$

2.2 Integració pel mètode del rectangle

Sigui $f(x)$ una funció de la qual volem saber la seva integral en $x \in [a,b]$. El mètode consisteix bàsicament en fer una suma de Riemann, on la seva discretització consisteix en partir el domini en n subintervals de mida $h = \dfrac{b-a}{n}$.

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} f\left(a + h\left(i+\frac{1}{2}\right)\right)h} \tag{2.3}$$

L'error associat al mètode és de $O(h)$, és a dir, lineal.

2.3 Integració pel mètode del trapezi

Sigui $f(x)$ una funció de la qual volem saber la seva integral en $x \in [a, b]$. El mètode consisteix bàsicament en fer una suma de Riemann, on la seva discretització consisteix en partir el domini en n subintervals de mida $h = \dfrac{b-a}{n}$. La diferència amb el mètode del rectangle és que en comptes de calcular àrees de rectangles es fa de trapezis.

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f(a+hi)\right)\frac{h}{2}} \tag{2.4}$$

L'error associat al mètode és de $O(h^2)$, és a dir, quadràtic.

2.4 Integració pel mètode de Simpson

Sense aprofundir gaire, el mètode consisteix en agafar el punt mitjà de cada subinterval. D'aquesta manera a cada subinterval hi ha definits tres punts, i es pot descriure unívocament una paràbola que passi pels tres.

La primera aproximació per la suma de Simpson (per a $n = 1$) és

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{b-a}{2}\right) + f(b)\right)} \tag{2.5}$$

L'error associat al mètode és de $O(h^3)$.

2.5 Integració pel mètode de Romberg

El mètode es deriva del mètode del trapezi. Sigui $T(h)$ l'aproximació de la integral de $f(x)$ a $[a, b]$ pel mètode del trapezi associada al pas h. Aquesta suma la podem expressar (mitjançant l'expansió de MacLaurin) com:

$$\begin{aligned} T(h) &\equiv \int_a^b f(x)\,\mathrm{d}x + \alpha_1 h^2 + \alpha_2 h^4 + \dots \\ T\left(h/2\right) &\equiv \int_a^b f(x)\,\mathrm{d}x + \alpha_1 \left(h/2\right)^2 + \alpha_2 \left(h/2\right)^4 + \dots \end{aligned}$$

Manipulant aquests dos valors arribem a una expressió exacta del valor de la integral:

$$\boxed{\int_a^b f(x)\,\mathrm{d}x \equiv \frac{4T(h/2) - T(h)}{3} + O(h^4)} \tag{2.6}$$

Observem que el que hem pogut aconseguir és eliminar l'error d'ordre h^2, de manera que aquest mètode és molt més precís que el del trapezi.

Generalització de Romberg

Si en comptes de treballar amb dos valors de l'aproximació pel mètode del trapezi treballem amb m valors, l'expressió de Romberg esdevé

$$\boxed{R_{m,j} = \frac{4^{j-1}R_{m,j-1} - R_{m-1,j-1}}{4^{j-1} - 1}} \tag{2.7}$$

on $R_{m,1}$ són les aproximacions de la integral pel mètode del trapezi, que segueixen la recurrència següent:

$$\boxed{R_{m,1} = T\left(\frac{h}{2^{m-1}}\right)} \tag{2.8}$$

Una manera senzilla d'entendre l'algoritme és pensar en una matriu de resultats. Per exemple, a partir de quatre aproximacions inicials ($m_{\max} = 4$), tenim:

$$\begin{pmatrix} R_{1,1} & 0 & 0 & 0 \\ R_{2,1} & R_{2,2} & 0 & 0 \\ R_{3,1} & R_{3,2} & R_{3,3} & 0 \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} \end{pmatrix}$$

Notem que la precisió del resultat R augmenta amb el creixement de m i de j, de manera que el cal esperar que $R_{m,m}$ sigui el que s'aproximi amb més exactitud al resultat veritable.

Observem que l'error associat al mètode de Romberg depèn de l'ordre m, en particular és de $O(h^{2m})$.

Exemple 2.1. Volem saber $\int_0^1 e^{x^2}\,\mathrm{d}x$. Les tres primeres iteracions pel mètode del trapezi ens donen els següents resultats:

i) $T(1) \approx 1.859\,140\,914$.

ii) $T(1/2) \approx 1.571\,583\,165$.

iii) $T(1/4) \approx 1.490\,678\,862$.

Llavors, mitjançant el mètode de Romberg calculem els següents resultats:

iv) $R_{2,2} \approx 1.475\,730\,582$.

v) $R_{3,2} \approx 1.463\,710\,761$.

vi) $R_{3,3} \approx 1.462\,909\,439$.

Comparant amb el valor real de la integral, $\int_0^1 e^{x^2}\,\mathrm{d}x \approx 1.462\,651\,745$, podem veure que per cada iteració, el valor calculat convergeix a la solució de la integral.

```
1 # TODO: implement the code
```

▲