BOLZANO, un programa d'ensenyament assistit per ordinador

Carles Batlle i Arnau
Dept. de Matemàtica Aplicada i Telemàtica
E.U.P. de Vilanova i la Geltrú
Universitat Politècnica de Catalunya

Abril 1990

Resum

Presentem una descripció d'un programa que permet comprovar de manera visual l'efectivitat dels diferents mètodes (bisecció, *regula falsi*, secant i Newton) d'obtenció de zeros de funcions reals de variable real.

1 Introducció

La difusió massiva dels ordinadors personals ha possibilitat la presentació a les escoles i facultats de mètodes numèrics que abans eren intractables per l'enorme quantitat de "càlcul" manual que representaven. Des d'un punt de vista pràctic, molts problemes que tenien una solució analítica complicada o inexistent poden ara tractar-se de manera aproximada sense gaires complicacions conceptuals, utilitzant la "força bruta" de l'ordinador. Ens interessa però una altra vessant de la situació: la possibilitat d'usar l'ordinador com a eina d'ensenyament, que reforçi o presenti d'una manera visual els mètodes i conceptes explicats a classe. En aquest sentit, hem creat un programa, anomenat *Bolzano* per raons òbvies, que vol posar de manifest els punts forts i les febleses dels diferents mètodes per cercar els zeros de funcions reals de variable real. Es tracta que l'alumne vegi això de manera directa, observant el progrés (o la degradació) de les iteracions en els diferents algorismes. No és, per tant, un programa creat per a trobar ràpidament els zeros, encara que pot utilitzar-se en aquest sentit si així es desitja. El programa incideix en la part gràfica, i això fa que la major part del temps estigui calculant les coordenades dels punts a representar. Un programa fet únicament per a calcular podria prescindir de tot això, encara que, a vegades, el suport visual ajuda a no caure en els desastres computacionals tan típics quan hom utilitza un mètode numèric sense saber ben bé què fa.

Aquest treball està organitzat de la següent manera. A la secció 2 presentem la teoria dels diferents algorismes i expliquem l'estratègia general, valorant la idoneitat de cada mètode en cada situació. A la secció 3 descrivim la utilització del programa tal com

es presenta a la versió 1.0. Finalment, a la secció 4 proposem una sessió pràctica, amb diferents exemples illustratius.

Bolzano ha estat escrit, compilat i *linkat* amb la versió integrada de Turbo C 2.0 de Borland. El codi font per la realització dels menús ha estat tret del llibre *Stretching Turbo C* de Kent Porter, editat per Brady, New York. *Bolzano* és un programa de lliure distribució.

2 Els diferents algorismes

El nostre objectiu és resoldre equacions reals d'una variable real, que sempre es poden escriure en la forma

$$f(x) = 0$$

. Per tant el problema es redueix a trobar els zeros d'una funció real de variable real $f(x)$.

L'eina fonamental serà el teorema de Bolzano:

Si $f(x)$ és contínua en l'interval tancat $[a,b]$, i es verifica $f(a)f(b) < 0$, llavors existeix $\xi \in (a,b)$ tal que $f(\xi) = 0$.

Els mètodes que utilitzarem poden classificar-se en dos grups:

1. mètodes que no utilitzen la derivada:
   (a) bisecció,
   (b) *regula falsi*,
   (c) secant.
2. mètodes que utilitzen la derivada. Estudiarem el mètode de Newton.

Val a dir que, a la pràctica, els algorismes més potents utilitzen una combinació dels mètodes anteriors, depenent de la disposició de $f(x)$ a collaborar. Existeixen també mètodes especials pel cas que $f(x)$ sigui un polinomi, ja que la possibilitat d'arrels múltiples, a més d'enganyar al teorema de Bolzano quan la multiplicitat és parella, presenta dificultats per a tots els mètodes esmentats, especialment per als mètodes de la secant i de Newton, degut a l'anullació de la derivada. Més informació pot trobar-se a Press, William H., et al., *Numerical Recipes in C*, Cambridge University Press, New York.

El primer pas per a qualsevol dels mètodes (no estrictament necessari per als mètodes de Newton i la secant però sí convenient), consisteix en trobar l'interval dins el qual estem segurs que hi ha almenys un zero. Per aplicar el teorema de Bolzano, és necessari que la funció sigui contínua en aquest interval tancat. Fet això hom pot passar a utilitzar els mètodes que tot seguit descriurem.

2.1 Mètode de la bisecció

El mètode de la bisecció és un que no pot fallar. La idea és senzilla. Sabem que en un interval $[a,b]$ hi ha un zero perquè la funció contínua $f(x)$ hi canvia de signe. Avaluem la funció en el punt $c = (a+b)/2$, el centre de l'interval. Utilitzem llavors c per substituir a

si $f(a)$ i $f(c)$ tenen el mateix signe i seguim llavors amb $[c, b]$, o per substituir b si $f(b)$ i $f(c)$ tenen el mateix signe i continuem en aquest cas amb $[a, c]$. Després d'una d'aquestes iteracions, l'interval que conté l'arrel ha disminuït la seva longitud per un factor 2. Si després d'n iteracions l'arrel està dins un interval de tamany ϵ_n, després de la següent iteració estarà dins un interval de longitud

$$\epsilon_{n+1} = \epsilon_n/2.$$

Per tant, si $\epsilon_0 = b - a$ és el tamany de l'interval inicial i volem arribar a conèixer l'arrel amb un error ϵ, haurà de ser $\epsilon = \epsilon_0/2^n$, és a dir, necessitarem efectuar

$$n = log_2(\epsilon_0/\epsilon)$$

biseccions. Per construcció, és evident que això no pot fallar. Si un interval conté una arrel, la bisecció la trobarà. Si en conté un nombre senar, el mètode en trobarà una d'elles.

Quan un mètode convergeix com un factor més petit que 1 multiplicat per l'error del pas anterior elevat a la primera potència, com en el cas de la bisecció, on el factor és 1/2, hom diu que la convergència és lineal. Mètodes que convergeixen amb una potència superior

$$\epsilon_{n+1} = constant \text{ x } (\epsilon_n)^m, \ m > 1$$

s'anomenen superlineals. Convergència lineal vol dir que xifres significatives successives es guanyen linealment amb l'esforç de càlcul.

Donat que els ordinadors utilitzen un nombre fixat , o potser variable per *soft*, però en qualsevol cas finit, de dígits binaris per a representar un real, pot donar-se el cas que, mentre que la nostra funció s'anulli per a un cert real, no ho faci per cap dels reals que la màquina pot representar. Per tant hom ha de decidir fins quina precisió vol arribar. Per exemple, un error de 10^{-6} és assolible si l'arrel està al voltant de $x = 1$, però no té sentit intentar aconseguir-lo si l'arrel és a la vora de $x = 10^{26}$. Un criteri pràctic pot ser demanar un error final de l'ordre de

$$\alpha(b + a)/2$$

on α és la precisió de l'ordinador (el menor nombre real positiu tal que $1 + \alpha$ és encara diferent de 1).

2.2 Mètodes de la *regula falsi* i de la secant

Per a funcions que són prou suaus vora l'arrel, els mètodes de la *regula falsi* (posició falsa) i de la secant convergeixen generalment més depressa que la bisecció, ja que utilitzen més informació sobre la funció. En els dos mètodes se suposa que la funció és aproximadament linial a la regió d'interès, i la següent millora de l'arrel s'agafa en el punt on la recta talla l'eix. Després de cada iteració un dels punts interiors es descarta en favor de la nova aproximació.

L'única diferència entre els dos mètodes està precissament en quin dels punts és abandonat. Mentre que la *regula falsi* es queda amb l'extrem de l'interval anterior on la

funció té signe canviat respecte a la nova aproximació, de tal manera que els dos punts continuen tancant una arrel, tal com passava amb la bisecció, el mètode de la secant es queda amb la més recent de les aproximacions anteriors (això requereix una decisió arbitrària en el primer pas), independentment de si el nou interval conté o no el zero que busquem. Quan les coses van bé, el mètode, si en un determinat moment ha escollit l'interval "dolent", torna a posar-se dins d'un de bo al cap d'una estona.

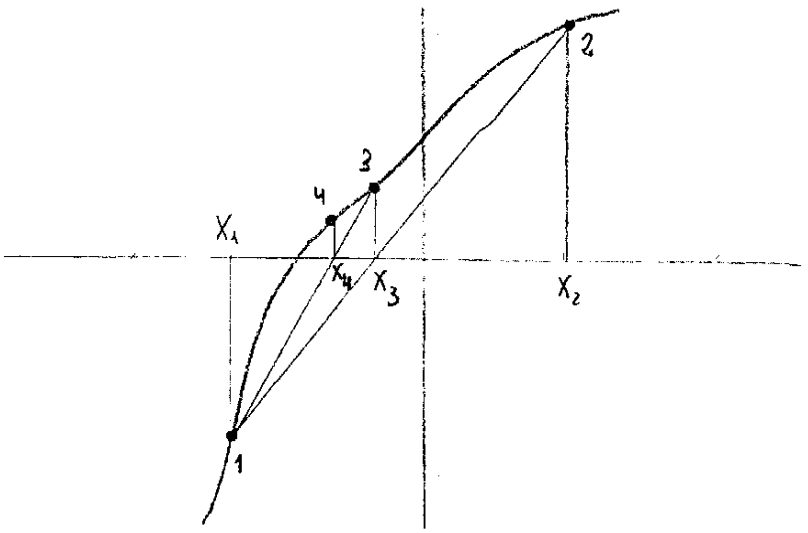
Matemàticament, el mètode de la secant convergeix més ràpidament a la vora d'una arrel d'una funció prou ben comportada. Hom pot demostrar que

$$\lim_{k\to\infty} \epsilon_{k+1} = const.\ \mathbf{x}\ (\epsilon_k)^{1.618...}$$

de manera que la convergència és superlineal. El mètode de la secant té, com hem dit, però, la desventatja de que els intervals poden deixar de contenir l'arrel. Per funcions que no co-operin, l'algorisme pot per tant no convergir.

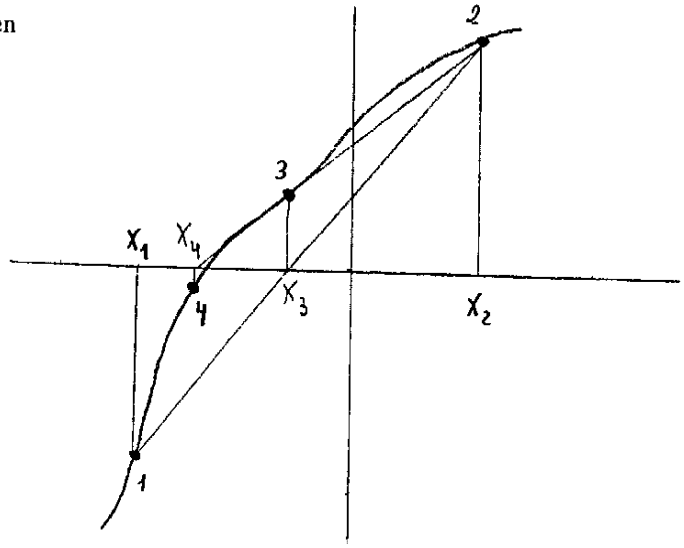
El mètode de la *regula falsi* , com que sovint mantè un valor antic, té un ordre de convergència menor. Com que el valor més recent és a vegades l'escollit, sovint el mètode és superlineal.

Gràficament tenim, per al mètode de la *regula falsi* ,



mentre que per al mètode de la secant, la mateixa funció i els mateixos punts inicials

donen



Analíticament, donats x_1 i x_2, hom té que l'equació de la recta que passa per $(x_1, f(x_1))$, $(x_2, f(x_2))$ és

$$y = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Si anomenem x_3 el punt on aquesta recta talla l'eix de les X, serà

$$0 = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x_3 - x_1)$$

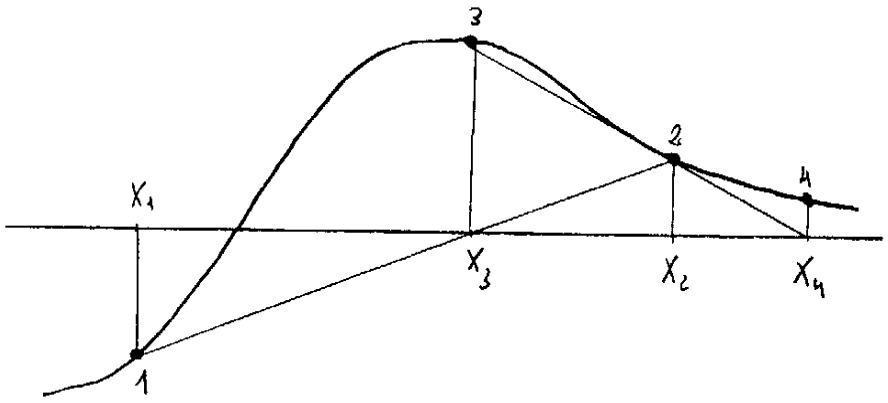
d'on

$$x_3 = x_1 - f(x_1)\frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

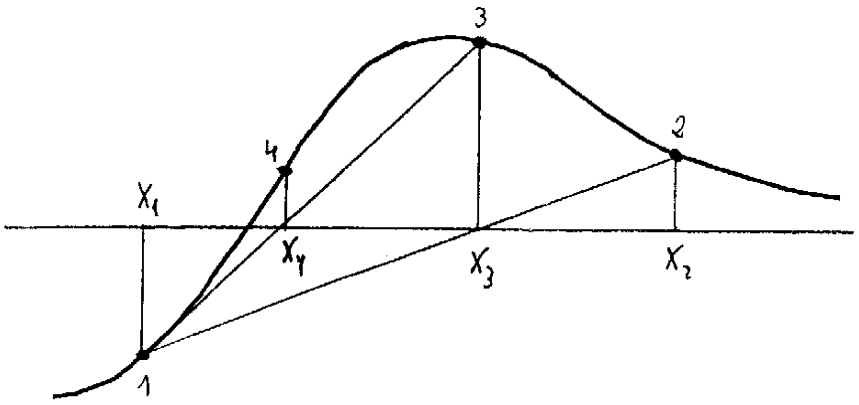
Amb el mètode de la secant, ens quedaríem amb x_2 i x_3, mentre que amb la *regula falsi* agafaríem x_1 i x_3 si $f(x_2)f(x_3) > 0$, i x_3 i x_2 si $f(x_1)f(x_2) > 0$.

Un exemple on el mètode de la secant fracassa, degut a una mala elecció dels punts

inicials, és

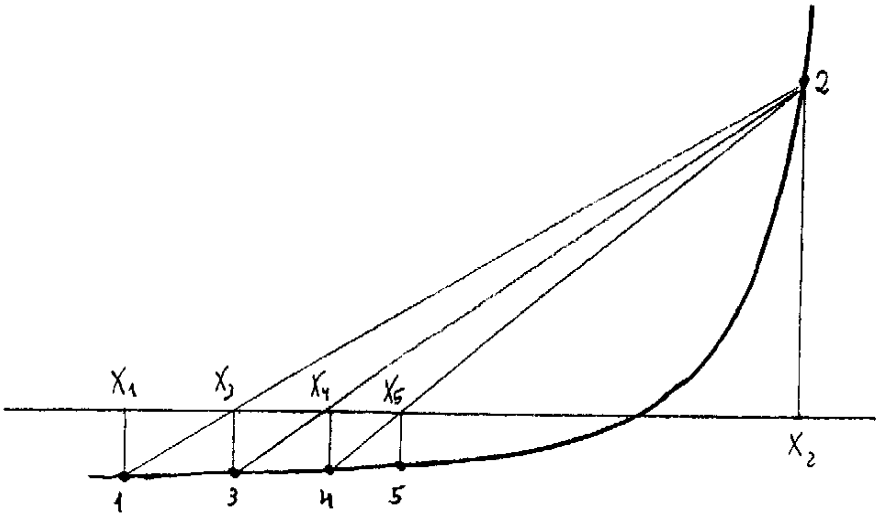


En canvi la *regula falsi* no hi tindria cap problema



Finalment, mostrarem una situació on la *regula falsi* convergeix (com sempre), però

ho fa molt lentament, mentre que la bisecció s'acostaria ràpidament a l'arrel



Una estratègia combinada prou efectiva en qualsevol cas seria utilitzar la *regula falsi* i la bisecció per acostar-se a l'arrel, intercalant passos amb la bisecció quan la *regula falsi* no s'acosti prou depressa, i quan fóssim prou a la vora de l'arrel, sense tenir extrems relatius pel mig, emprar el mètode de la secant per a millorar ràpidament el valor. Tot això, es clar, sempre que no vulguem o no puguem utilitzar la derivada.

2.3 Mètode de Newton

Aquest mètode es coneix també a vegades amb el nom de mètode de Newton-Raphson i, tal com hem dit, es distingeix dels anteriors pel fet que necessita el coneixement de la funció derivada $f'(x)$ a més de $f(x)$. L'algorisme consisteix en extrapolar la funció per la seva recta tangent en un punt inicial, i agafar com aproximació següent el punt on la recta tangent talla l'eix. A diferència també dels mètodes anteriors, sols ens cal un punt per començar l'algorisme.

Analíticament, el mètode es fonamenta en el desenvolupament de Taylor d'una funció al voltant d'un punt. Anant fins a primer ordre, tenim

$$f(x_2) = f(x_1) + f'(x_1)(x_2 - x_1) + \frac{f''(\xi)}{2}(x_2 - x_1)^2$$

amb ξ entre x_2 i x_1. Per a valors de $x_2 - x_1$ prou petits i per funcions prou suaus, el reste de Lagrange d'ordre 1

$$R_1(x_2) = \frac{f''(\xi)}{2}(x_2 - x_1)^2$$

pot ser menyspreat i per tant $f(x_2) = 0$ implica

$$0 = f(x_1) + f'(x_1)(x_2 - x_1),$$

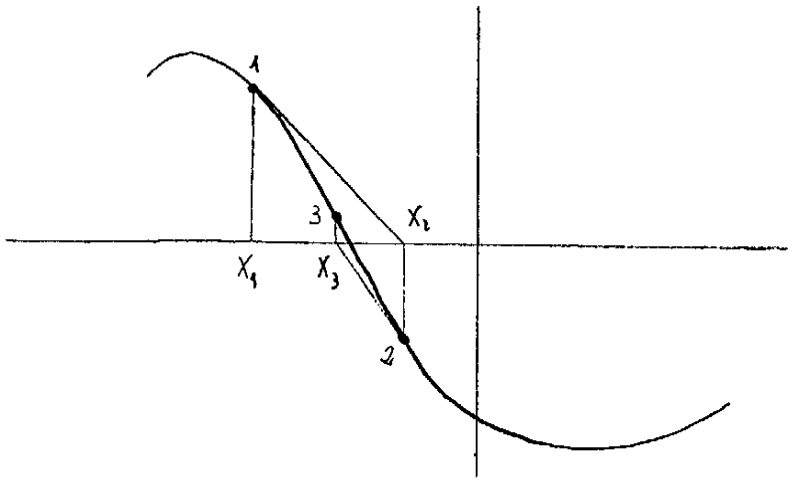
és a dir,

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

i, en general,

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

Gràficament tenim



Si la diferència entre x_i i l'arrel exacta β és ϵ_i, de manera que $\epsilon_i = x_i - \beta$, tindrem, restant β dels dos membres de l'anterior equació,

$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)}$$

Com que $f(\beta) = 0$, tenim

$$f(x_i) = f'(\beta)\epsilon_i + \frac{f''(\beta)}{2}{\epsilon_i}^2 + o(\epsilon_i)^3$$

on la notació $o(\epsilon_i)^3$ vol dir termes que van com a mínim com potències de ϵ_i al cub, i que seran per tant molt petits si ϵ_i és petit. De la mateixa manera tenim

$$f'(x_i) = f'(\beta) + f''(\beta)\epsilon_i + o(\epsilon_i)^2$$

Si posem tot això a l'expressió que ens dóna l'error de dues iteracions successives, ens queda

$$\epsilon_{i+1} = \epsilon_i - \frac{f'(\beta)\epsilon_i + 1/2f''(\beta){\epsilon_i}^2 + o(\epsilon_i)^3}{f'(\beta) + f''(\beta)\epsilon_i + o(\epsilon_i)^2}$$

Si a la fracció dividim per $f'(\beta)$ i escrivim $\kappa \equiv f''(\beta)/f'(\beta)$, ens quedarà

$$\epsilon_{i+1} = \epsilon_i - \frac{\epsilon_i + 1/2\kappa{\epsilon_i}^2 + o(\epsilon_i)^3}{1 + \kappa\epsilon_i + o(\epsilon_i)^2}$$

Ara expandim el denominador d'acord amb el desenvolupament

$$\frac{1}{1+z} = 1 - z + o(z)^2$$

Ens resulta així

$$\epsilon_{i+1} = \epsilon_i - \left(\epsilon_i + 1/2\kappa{\epsilon_i}^2 + o(\epsilon_i)^3\right)\left(1 - \kappa\epsilon_i + o(\epsilon_i)^2\right)$$

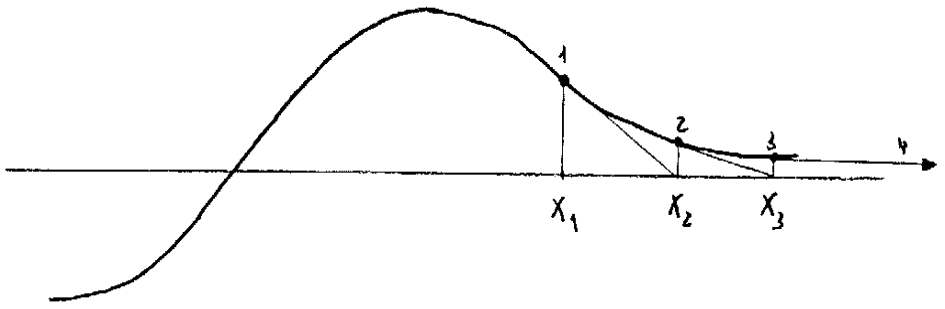
Multiplicant i agafant fins a termes quadràtics en ϵ ens resulta

$$\epsilon_{i+1} = \epsilon_i - (\epsilon_i - \kappa{\epsilon_i}^2 + 1/2\kappa{\epsilon_i}^2 + o(\epsilon_i)^3)$$

I ara és quan es manifesta la gràcia del mètode de Newton: el terme lineal en ϵ_i desapareix i resulta finalment

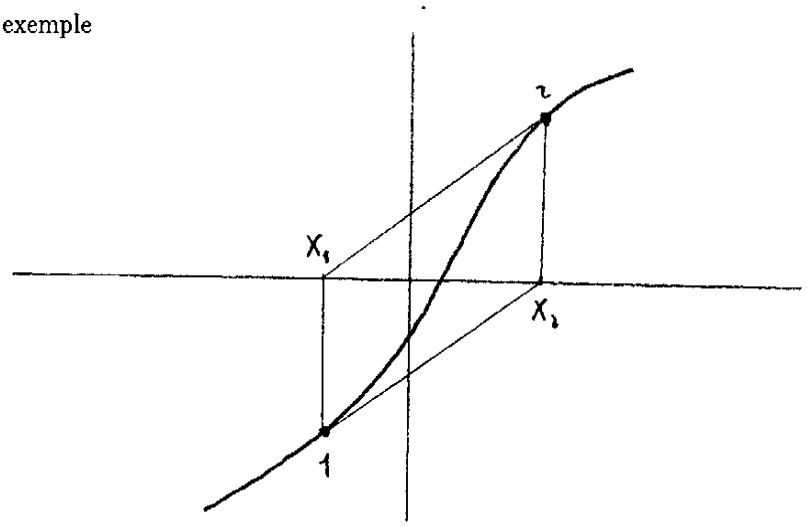
$$\epsilon_{i+1} = {\epsilon_i}^2 \frac{f''(\beta)}{2f'(\beta)} + o(\epsilon_i)^3$$

Per tant el mètode de Newton convergeix *quadràticament*, *i.e.*, si en un moment l'error és 10^{-3} serà 10^{-6} al pas següent. S'ha d'anar però amb compte. Tal com passava amb el mètode de la secant, res ens garanteix que ens mantinguem vora un zero. En particular, si ens posem en una zona de derivada quasi nulla o que té un pendent "equivocat", el mètode pot enviar la següent aproximació molt lluny del zero, amb molt poques possibilitats de recuperació. Per exemple



Naturalment, enxampar un extrem relatiu suposa la defunció immediata de l'algorisme. Una altra patologia que pot presentar-se és quan hom entra en un cicle no convergent.

Per exemple



Resumint, el mètode de Newton, igual que el de la secant, no és convenient en les primeres etapes de la recerca d'una arrel, però una vegada som a la vora, la seva convergència quadràtica fa que sigui el mètode ideal per enllestir la feina aviat.

3 Utilització de *Bolzano*

Bolzano requereix un ordinador compatible IBM, corrent sota MSDOS 3.0 o superior i dotat de gràfics Hercules, CGA, MCGA, EGA o VGA. Si hi ha instal.lat un coprocessador 8087, 80287 o 80387, el programa l'utilitza i això accelera considerablement el temps de resposta de la màquina. *Bolzano* es presenta en un sol fitxer executable, `bolzano.exe`, i es crida teclejant `bolzano` a la línia de comandaments. Apareix immediatament una pantalla de presentació que desapareix en pitjar qualsevol tecla. Tenim llavors a la part superior central de la pantalla un menú amb les opcions

Precisió
Regula falsi
Secant
Newton
Bisecció

amb la primera opció preseleccionada. A la part baixa de la pantalla hi ha una barra d'ajuda amb les indicacions

$F1 - Ajuda \quad Esc - Tornar\ al\ menú\ principal \quad F10 - Executar$

de les quals sols la primera està activada en aquest punt. Si premeu F1 apareix una finestra amb indicacions per a seleccionar les diferents opcions del menú. Podeu utilitzar les fletxes i apretar ENTER quan sigueu sobre l'opció dessitjada, o podeu directament

prémer la lletra majúscula que apareix a cada opció. Anem tot seguit a explicar en què consisteix cada una d'elles.

Precisió. Quan la diferència entre dues aproximacions successives d'una arrel és menor que la precisió que hi ha definida, l'algorisme finalitza i es dóna l'arrel per bona. Per defecte, la precisió és 10^{-4}. L'algorisme també acaba si el valor de la funció en el valor actual de l'arrel és menor en valor absolut que 10^{-10} (això no és modificable en aquesta versió). Per introduir la precisió cal acabar amb ENTER. Cal tenir en compte el següent:

1. els valors numèrics no es poden escriure en forma exponencial. Cal escriure totes les xifres.
2. després del punt decimal, . , ha d'aparèixer al menys un dígit.

Aquestes regles són vàlides tot al llarg del programa. Si el que s'ha introduït no és acceptable, surt un missatge i la precisió resta inalterada.

Regula falsi. Si escolliu aquesta opció apareixerà superposat, lleugerament a la dreta, un nou menú amb les opcions

Funció
extrem Inferior
extrem Superior

Us podeu moure entre aquestes opcions de la mateixa manera que en el menú principal, al qual podeu retornar amb ESC. Si pitgeu ara F1 apareixerà una finestra amb una breu descripció del mètode de la *regula falsi*. Per poder començar l'algorisme cal introduir la funció els zeros de la qual volem buscar i un interval dins el qual esperem trobar-ne algun. Si trieu l'opció **Funció** apareix una caixa que demana l'expressió de la funció en termes de la variable x. Cap altra variable és acceptable. Es finalitza la introducció amb ENTER i podeu corregir i borrar qualsevol cosa. En aquesta versió no hi ha possibilitat d'escriure sobre el que ja està posat, de manera que per corregir quelcom cal borrar el text previ. Si l'expressió funcional no és intel·ligible pel programa apareix un avís. En aquest sentit cal tenir en compte el següent:

1. el producte s'ha d'indicar explícitament amb *.
2. el signe decimal és . i després del mateix ha d'aparèixer sempre com a mínim un dígit.
3. els arguments de les funcions han d'estar entre parèntesis, ( ), i les funcions que aquesta versió reconeix són:
   (a) `abs(x)`, el valor absolut d'x.
   (b) `acos(x)`, inversa de cos(x).
   (c) `asin(x)`, inversa de sin(x).
   (d) `atan(x)`, inversa de tan(x).
   (e) `cos(x)`, el cosinus d'x.
   (f) `cosh(x)`, el cosinus hiperbòlic d'x.
   (g) `exp(x)`, l'exponencial d'x en base e.

(h) `int(x)`, la part entera d'x.
(i) `log(x)`, el logaritme neperià d'x.
(j) `log2(x)`, el logaritme d'x en base 2.
(k) `log10(x)`, el logaritme d'x en base 10.
(l) `pow2(x)`, l'exponencial d'x en base 2.
(m) `pow10(x)`, l'exponencial d'x en base 10.
(n) `sin(x)`, el sinus d'x.
(o) `sinh(x)`, el sinus hiperbòlic d'x.
(p) `sqr(x)`, el quadrat d'x.
(q) `sqrt(x)`, l'arrel quadrada positiva d'x.
(r) `tan(x)`, la tangent d'x.
(s) `tanh(x)`, la tangent hiperbòlica d'x.

Podeu composar funcions fins el nivell que vulgueu, però en aquesta versió l'expressió d'una funció no pot tenir més de 70 caràcters.

4. podeu expressar les potències mitjançant `^`. Per exemple, `x^3`.
5. una expressió de longitud nulla no és acceptable.

Si apareix l'avís de funció no acceptable, el programa retorna al menú i cal modificar la funció si voleu fer quelcom (no cal fer-ho, però, immmediatament). Tot seguit podeu introduir els extrems inferior i superior de l'interval. Si tot està correcte, prement ara F10 s'executa l'algorisme de la *regula falsi*. En aquest moment el programa canvia a mode gràfic i, de no produir-se cap desastre, poden pasar dues coses. Si la funció no canvia de signe en els extrems de l'interval, veureu la gràfica en l'interval considerat (més un 10% a cada banda per motius estètics), i un missatge apareixerà comunicant-vos que heu triat un interval inacceptable. Qualsevol tecla us tornarà en mode texte al menú principal. Podeu tornar a escollir `Regula falsi` i canviar els extrems (es manté la funció que heu introduït abans). Si torneu a provar i aquesta vegada l'interval és correcte us sortirà en pantalla el resultat del primer pas de *regula falsi*: una recta que uneix els extrems de la gràfica en l'interval considerat i que, al tallar l'eix d'abcisses, determina el nou zero. Aquest apareix escrit explícitament a la banda esquerra de la pantalla, així com el valor de la funció en aquest punt. També teniu un recordatori de quina funció esteu considerant, del mètode i de les iteracions que porteu. Si premeu qualsevol tecla, excepte ESC, s'executa una nova iteració, i així se segueix fins que s'assoleix la precisió desitjada o fins que el valor absolut de la funció és menor que 10^{-10}. Si en qualsevol moment us canseu podeu tornar al menú principal amb ESC.

Secant i bisecció. Aquestes opcions són del tot semblants a l'anterior. La funció i els extrems introduïts en qualsevol opció es propaguen a totes les altres, de manera que no cal tornar a a teclejar-los si canvieu de mètode. La funció es propaga també a l'opció `Newton`.

Newton. Cal aquí introduir, a més de la funció si no ho heu fet abans, l'expressió de la derivada. Aquesta és la vostra responsabilitat. Si no sabeu derivar *Bolzano* no és prou savi per suplir les vostres deficiències i seguirà endavant amb l'algorisme, produint resultats incoherents. Per començar l'algorisme cal introduir un valor inicial. El programa determina automàticament quin interval representarà en pantalla.

Si qualsevol dels algorismes porta a punts fora del domini de la funció, és la fi i torneu al menú principal, amb un avís de quin punt ha provocat el problema. Si és la zona auxiliar del 10% la que cau fora del domini, simplement no es representa la funció en aquesta zona i se segueix endavant amb l'algorisme.

El programa està raonablement protegit contra perversitats i/o accidents. De tota manera, la imaginació de la gent és illimitada quan es tracte de trobar l'estret camí vers el desastre. Si l'ordinador queda penjat, intenteu un *reset* en calent (Ctrl-Alt-Del) i, si fins i tot això falla, ja sabeu què heu de fer. L'autor agrairà que li sigui comunicat qualsevol problema amb una descripció detallada de les circumstàncies en què s'hagi produït.

4 Una proposta de pràctica

1. Començarem amb una funció força complicada i veurem com podem calcular tots els seus zeros amb els diferents mètodes. Podrem apreciar algunes de les complicacions més freqüents.
   (a) Introduïu la funció $f(x) = \sin x^3 + x^2 - 2$. Efectueu una *regula falsi* entre -4 i 4. La funció no canvia de signe en aquest interval, però això ens permet fer una idea del nombre de zeros. Observem que n'hi ha tres, un al voltant de -1.5 i tres més entre 1 i 2. Podem estar segurs que no hi ha més zeros?
   (b) Anem a calcular el zero negatiu. Apliqueu una *regula falsi* entre -3 i -1. Fixeu-vos en com són escollits els intervals. Apunteu els extrems de cada interval per cada iteració. Podeu preveure quin serà l'interval següent? Amb la precisió per defecte (10^{-4}), quantes iteracions són necessàries? Apunteu el zero que obteniu.
   (c) Feu el mateix per la secant. De nou, podeu preveure l'interval següent. Hi ha algun canvi respecte a la *regula falsi* ?
   (d) El mateix amb el mètode de la bisecció. Observeu que mentre que la *regula falsi* i la secant han tardat més o menys el mateix, la bisecció s'eternitza.
   (e) Anem ara a utilitzar el mètode de Newton. Cal introduir la derivada. No us equivoqueu. Comenceu amb el punt inicial $x = -1$. Què passa? Es produeix un desastre. Sortiu. Quin punt inicial hem d'escollir? Hem d'anar més a la dreta o més a l'esquerra de -1? Podeu respondre a la pregunta examinant la funció entre -2 i -1. La resposta és que hem d'anar més a l'esquerra. Amb $x = -1.2$ encara es produeix l'allunyament, però amb $x = -1.3$ ja es produeix la convergència.
   (f) Anem ara a buscar els altres tres zeros. En primer lloc, canvieu la precisió a 10^{-6} (recordeu, no pot utilitzar-se notació exponencial, de manera que heu de teclejar els zeros). Amb la *regula falsi* , examineu la funció entre .5 i 2. Un dels zeros està al voltant de 1, un altre vora 1.5 i l'últim vora 1.7.
   (g) Apliqueu *regula falsi* , secant i bisecció entre 1 i 1.2. Anoteu els resultats i el nombre d'iteracions. Apliqueu Newton començant amb $x = 1$. Observeu la ràpida convergència.

(h) Feu el mateix entre 1.3 i 1.7. Què passa amb la secant? Apliqueu Newton a partir de 1.3. Les coses no van gaire bé. Si proveu amb 1.32 i 1.35 no es produeix el desastre però us acosteu cap l'altre zero en lloc del que voleu. Amb $x = 1.37$ convergiu cap el zero que ens interessa.

(i) Passeu ara a capturar l'últim zero amb l'interval entre 1.65 i 1.9. La *regula falsi* , que fins ara havia estat capaç de mantenir el duel amb la secant, queda ara descol·locada quasi al nivell de la bisecció. Quin és el motiu?

2. En l'exemple anterior, el mètode de Newton, si començava molt a prop de l'arrel, sempre era el més ràpid. Anem ara a posar-lo en problemes.

(a) Considereu $f(x) = \sin x$. Començant amb $x = 1$, Newton convergeix ràpidament cap a $x = 0$. Proveu ara el mateix amb $f(x) = \sin x^3$. Sou capaços d'esbrinar perquè Newton convergeix tant lentament? Si no enteneu res, considereu $f(x) = x^3$ i apliqueu-hi Newton. També tenim el mateix problema. Demostreu analíticament que la convergència és *lineal* amb constant multiplicativa 2/3 i comproveu-ho experimentalment. Tornant a $f(x) = \sin x^3$, agafeu un interval no simètric al voltant del zero, i apliqueu *regula falsi* i la secant. El problema no és per tant exclusiu de Newton. En canvi, la modesta bisecció enllesteix aviat perquè, de manera natural, es construeix un interval simètric al voltant de $x = 0$ i l'encerta de ple a la següent.

(b) Considereu ara $f(x) = \arctan x$. Apliqueu Newton començant amb un valor més gran que 2. Divergeix. Aneu disminuint el punt inicial de 0.1 en 0.1 fins aconseguir entrar en un cicle convergent. Podeu calcular analíticament el valor màxim del punt inicial que fa que l'algorisme no se'n vagi a l'infinit? Qué passa si comenceu exactament en aquest valor?

3. Sigui ara $f(x) = x^4 - x - 3$ i apliqueu la bisecció, la *regula falsi* i la secant entre -2 i 0. Anoteu el nombre d'iteracions. La *regula falsi* queda a la cua. Apliqueu ara Newton començant per $x = 0$. Acabeu en vuit iteracions. Torneu a repetir la secant i Newton. On li treu Newton l'avantatge? Resposta: als llocs on la funció té més pendent. Aquí Newton posa la directa mentre que la secant podríem dir que s'en va per la tangent, però què més voldria ella.

4. Cerqueu tots els zeros (o els de valor absolut més petit si n'hi ha un nombre infinit) de les següents funcions utilitzant cada vegada l'algorisme o combinació d'algorismes que creieu més convenients. Per exemple, podeu fer unes quantes biseccions per acostar-vos al zero, i després utilitzar l'últim interval de la bisecció com interval d'entrada per un altre mètode, o un valor dins de l'interval per seguir amb Newton. En cada cas penseu si val la pena calcular o no la derivada.

(a) $f(x) = \mid x \mid^3 - \exp x$.

(b) $f(x) = x \sin x - \exp x$.

(c) $f(x) = \cos x - x \tan x$.

(d) $f(x) = \log x - 3 \cos x$.

(e) $f(x) = x^5 + 6x^4 - 2x^3 - x^2 + 1$.