1 Introduction

J'expose, dans ce chapitre, quelques méthodes de recherche des racines d'une équation non-linéaire. Cette équation sera toujours écrite sous la forme conventionnelle

$$f(x) = 0.$$

Résoudre l'équation $f = 0$, chercher les zéros de f ou les pôles de $1/f$ sont des expressions équivalentes. Ce genre de problème se rencontre souvent, qu'il s'agisse de déterminer le point de fonctionnement d'une diode d'après sa caractéristique, la concentration d'une espèce chimique dans un mélange réactionnel ou la fréquence de coupure d'un filtre électrique. Je n'envisagerai ici que des fonctions réelles. On peut être amené à chercher toutes les solutions de $f = 0$, ou seulement quelques unes ou une seule, la plus petite par exemple. Un cas particulier important est celui où f est un polynôme ; on sait alors que les racines sont réelles ou deux à deux complexes conjuguées, en nombre égal au degré du polynôme.

Il est rare que je puisse écrire une solution analytique de l'équation $f = 0$; en fait, cela ne se produira que pour les polynômes de degré inférieur ou égal à 4 ou pour quelques fonctions simples. En conséquence, toutes les méthodes générales de recherche de racine sont des méthodes itératives ; partant d'une solution approchée, j'obtiendrai une suite d'approximations de plus en plus précises. Je devrai donc me préoccuper de la convergence de la méthode et de la vitesse de convergence, je devrai définir un critère d'arrêt des itérations et prévoir le rôle des erreurs d'arrondi inévitables dans tout calcul numérique. J'insiste sur le fait qu'aucune méthode connue ne fonctionne «en aveugle» : on doit toujours avoir une connaissance au moins approximative de l'emplacement de la racine. Je vous recommande vivement de tracer le graphe de la fonction pour avoir une idée du nombre et de la position des zéros.

2 Méthode de bissection ou de dichotomie

Je m'intéresse à une fonction $f(x)$ continue sur l'intervalle $I = [a, b]$ et telle que $f(a)f(b) < 0$. Il découle de ces hypothèses que f s'annule au moins une fois dans I et c'est ce zéro que je souhaite localiser précisément. Le pseudo-code correspondant à l'algorithme de bissection s'écrit

poser $x_g := a$, $x_d := b$, $nit := 0$;
tant que $|x_g - x_d| > TOL$ et que $nit < NITMAX$
faire
    poser $x_m := (x_g + x_d)/2$
    si $f(x_g)f(x_m) < 0$ alors $x_d := x_m$
    sinon $x_g := x_m$
    incrémenter nit
afficher nit, $x^* = (x_g + x_m)/2$, $f(x*)$

Le principe de la méthode est facile à comprendre. Je calcule la valeur de la fonction au milieu de l'intervalle $[x_g, x_d]$ et j'observe le changement de signe : s'il se produit entre x_g et x_m, c'est que la racine se trouve dans cet intervalle ; dans ce cas, je redéfinis la borne droite de l'intervalle et je recommence ; le raisonnement est semblable si le changement de signe se produit entre x_m et x_d. A chaque itération, l'intervalle qui contient la racine voit sa longueur divisée par 2.

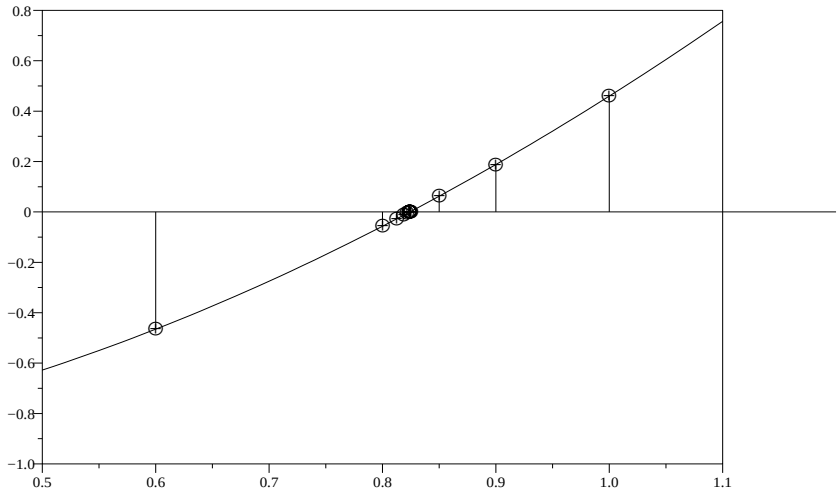
Comme dans toute méthode itérative, j'ai du définir un critère d'arrêt, ici le fait que la longueur du segment $[x_g, x_d]$ devienne inférieur à la constante TOL. Il se pourrait que cette condition ne

soit jamais remplie : c'est pourquoi je surveille aussi le nombre d'itérations et j'arrête le calcul si nit dépasse $NITMAX$. J'aurais pu aussi bien choisir la condition $f(x_m) < SEUIL$ comme critère d'arrêt.

Cette ébauche est incomplète : outre le fait que les constantes TOL et $NITMAX$ ne sont pas définies, je n'ai rien prévu pour le cas où $f(x_m) = 0$. De plus, il est nécessaire que $a < b$ et que l'intervalle I ne contienne qu'une racine.

Cependant, une fois amélioré, l'algorithme de dichotomie présente de nombreux avantages. La convergence est certaine dès lors que les conditions précédentes sont remplies ; la programmation est très simple ; le calcul de f n'a pas besoin d'être très précis, parce que j'utilise en fait le signe de la fonction et non sa valeur. Le désavantage est que la convergence est assez lente ; comme l'intervalle où peut se trouver la solution est divisé par deux à chaque itération, je dis que la convergence est approximativement linéaire.

La figure montre un exemple de recherche de racine de l'équation $x^2 = \cos x$ réalisé sous Scilab.

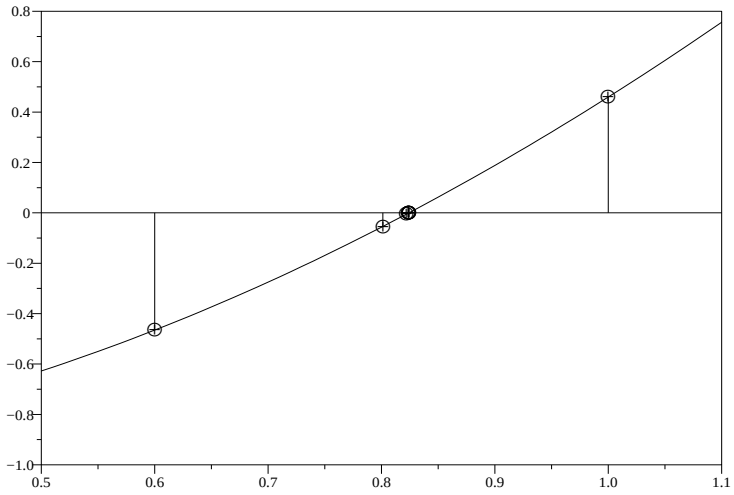


3 Méthode «Regula falsi»

Ces mots latins (règle des fausses positions, une expression qui date du 17ième siècle) désignent une variante de la méthode de dichotomie qui utilise une meilleure estimation de la nouvelle abscisse (x_m pour la bissection).

À chaque itération de l'algorithme précédent, je connais deux abscisses, x_g et x_d, qui encadrent la racine inconnue x^*. Les nombres $y_g = f(x_g)$ et $y_d = f(x_d)$ sont donc de signes contraires si bien que les deux points $\mathrm{G}(x_g, y_g)$ et $\mathrm{D}(x_d, y_d)$sont situés de part et d'autre de l'axe des x. La corde GD coupe donc cet axe en un point M, lequel sera «assez proche» de x^* ; je choisirai x_m comme nouvelle borne de l'intervalle contenant la racine, borne gauche si le changement de signe de f se produit entre x_m et x_d, borne droite dans le cas contraire. Il me reste à déterminer l'abscisse x_m :

$$x_m = x_g - y_g \frac{x_g - x_d}{y_g - y_d} = x_d - y_d \frac{x_g - x_d}{y_g - y_d}.$$



La convergence est souvent plus rapide qu'avec la méthode de bissection. J'ai choisi comme exemple la même équation qu'au paragraphe précédent, qui fait intervenir une fonction à variation régulière pour laquelle la convergence est très rapide. J'en ai profité pour modifier le critère d'arrêt : le calcul s'arrête dès que deux estimations successives de la racine sont «assez voisines» : $|x^{(n)} - x^{(n-1)}| < TOL$.

4 Méthode du point fixe ou d'itération

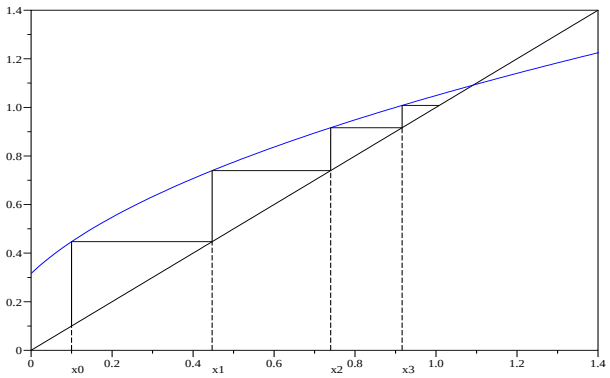
Je me propose de résoudre l'équation

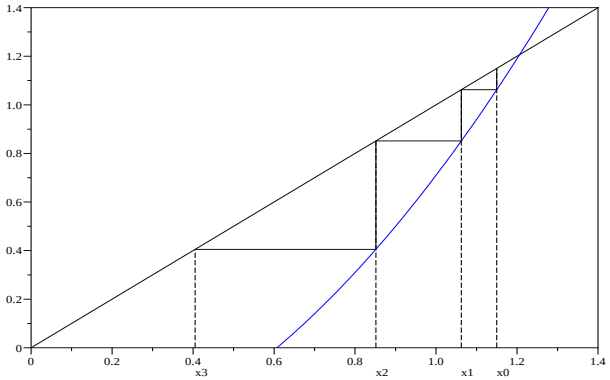
$$x = f(x). \tag{1}$$

Toute solution de cette équation s'appelle un point fixe de l'application f (l'image de x est x lui-même). Je vais procéder par itérations successives à partir d'une estimation $x^{(0)}$ de la solution :

$$x^{(n+1)} = f(x^{(n)}). \tag{2}$$

L'équation (2) admet une interprétation géométrique simple. La solution x^* de (1) est le point d'intersection de la première bissectrice (droite d'équation $y = x$) et de la courbe d'équation $y = f(x)$. L'ordonnée $f(x^{(0)})$ est la nouvelle abscisse $x^{(1)}$; le point $(0,x^{(1)})$ se déduit du point $(x^{(0)}, f(x^{(0)}))$ par une symétrie orthogonale par rapport à la droite $y = x$. Cette construction se répète pour $x^{(2)}, x^{(3)}, \cdots$ En essayant de résoudre (1) pour diverses fonctions f, vous verrez que l'on peut rencontrer deux dispositions de la suite $x^{(0)}x^{(1)}, \cdots$, correspondant soit à un point fixe attractif, soit à un point fixe répulsif, comme le montre la figure.





D'où la question : à quelle(s) condition(s) la suite des $x^{(n)}$ converge-t-elle, et vers quelle valeur ? Le raisonnement est simple et il est représentatif de nombreuses études de convergence. Je suppose que la fonction f et sa dérivée f' sont continues dans un intervalle $I = [a, b]$ entourant la racine x^*. De plus, f est telle que $a \leq f(x) \leq b$ si $x \in I$. Cette condition implique que tous les $x^{(k)}$ appartiennent à I si l'approximation initiale $x^{(0)}$ appartient à cet intervalle.

La racine vérifie (1) :

$$x^* = f(x^*).$$

Je défini l'erreur à l'itération k

$$e_k \equiv x^{(k)} - x^*.$$

Je commence par exprimer l'erreur à l'étape $k+1$ en fonction de e_k

$$e_{k+1} = x^{(k+1)} - x^* = f(x^{(k)}) - f(x^*).$$

D'après le théorème des accroissements finis, le second membre s'écrit

$$f(x^{(k)}) - f(x^*) = (x^{(k)} - x^*) f'(\xi_k)$$

où ξ_k est compris entre x^* et $x^{(k)}$. La relation cherchée s'écrit

$$e_{k+1} = f'(\xi_k) e_k. \tag{3}$$

Pour que l'itération converge, il faut que l'erreur diminue en valeur absolue à chaque tour, ce qui sera assuré si $|f'(\xi_k)| < 1$ quelque soit k. D'où le théorème

Théorème. Soit une fonction f continuement dérivable sur $[a,b]$ et telle que $f(x) \in [a,b]$ si $x \in [a,b]$. Soit encore M telle que

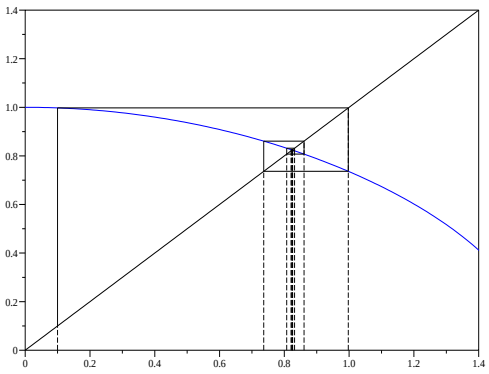
$$M = \sup_{a \le x \le b} |f'(x)| < 1.$$

Pour tout $x^{(0)} \in [a,b]$, la suite des itérés $x^{(0)}, x^{(1)}, \ldots, x^{(n)}, \ldots$ converge vers la solution de l'équation $x = f(x)$. On a de plus

$$\lim_{n \to \infty} \frac{x^* - x^{(n+1)}}{x^* - x^{(n)}} = f'(x^*).$$

Vous voyez que l'erreur est multipliée par un facteur inférieur à un à chaque itération : on dit que la convergence est linéaire.

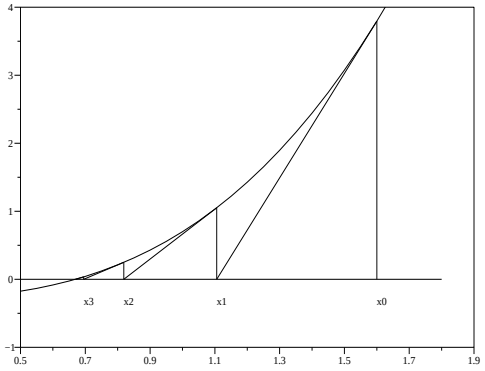
La méthode du point fixe est d'application assez générale car beaucoup d'équations peuvent se mettre sous la forme $x = f(x)$. L'exemple suivant remonte à l'Antiquité. L'équation $x^2 = a$ peut se mettre sous l'une des formes $x = a/x$ ou $x = \frac{1}{2}(x + a/x)$ qui ont été utilisées pour approcher $\sqrt{a}$.



La figure montre l'application de la méthode du point fixe à l'équation $x = \sqrt{\cos x}$; les conditions de convergence sont remplies dans [0,1] et, effectivement, la méthode converge vers $x^* = 0,824132$.

5 Méthode de Newton

La méthode de Newton s'applique à la résolution d'une équation de la forme $f(x) = 0$. Son interprétation géométrique est donnée sur la figure suivante.



Je dispose d'une approximation $x^{(0)}$ de la racine ; je construit la tangente à la courbe d'équation $y = f(x)$ au point d'abscisse $x^{(0)}$; cette droite coupe l'axe horizontal en $x^{(1)}$; je construit une nouvelle tangente en cette abscisse, dont l'intersection avec l'axe des x me donne $x^{(2)}$. Ce procédé est itéré jusqu"à convergence.

L'équation de la droite de pente $f'(x^{(k)})$ passant par le point $x^{(k)}, f(x^{(k)})$ s'écrit $y - f(x^{(k)}) = f'(x^{(k)})(x - x^{(k)})$. Cette droite coupe l'axe des x au point d'abscisse

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{4}$$

Cette équation définit l'algorithme de Newton pour la résolution des équations non-linéaires. Vous constatez immédiatement que la méthode va diverger lorsque que f' sera nulle ou même petite. En effet, si la pente de la courbe est faible, le point d'intersection $x^{(k+1)}$ de la tangente avec l'axe sera très éloigné du point d'abscisse $x^{(k)}$ et l'algorithme a toutes chances de se perdre.

Je vais maintenant établir la condition de convergence de façon précise ; la théorie devient très simple si l'on remarque qu'en posant

$$\phi(x) \equiv x - \frac{f(x)}{f'(x)},$$

je mets l'équation (4) sous la forme d'une itération vers un point fixe

$$x^{(k+1)} = \phi(x^{(k)}).$$

Cette fonction s'appelle parfois la fonction d'itération de la méthode de Newton. D'après le paragraphe précédent, je sais que la convergence dépend de $\phi'(x)$; or

$$\phi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}.$$

J'appelle encore x^* la racine que je cherche ; elle vérifie $f(x^*) = 0$ et par conséquent

$$\phi'(x*) = 0.$$

Comme je suppose que f, f', f'' sont définies et continues dans la région qui m'intéresse, il existe un intervalle $I = [x^* - \delta, x^* + \delta]$ tel que

$$|\phi'(x)| \leq C < 1.$$

Il me suffit donc de choisir $x^{(0)} \in I$ pour assurer la convergence de l'algorithme de Newton.

Pour estimer la vitesse de convergence, je fait un développement de Taylor de ϕ autour de x^* :

$$\phi(x^{(k)}) - \phi(x^*) = \frac{1}{2}(x^{(k)} - x^*)^2 \phi''(\eta_k)$$

où η_k est compris entre x^* et $x^{(k)}$. J'en déduis

$$e^{(k+1)} = \frac{1}{2}[e^{(k)}]^2 \phi''(\eta_k).$$

Lorsque k croit, $x^{(k)}$ et η_k se rapprochent de x^*, ce que je traduis par l'expression

$$\lim_{k\to\infty} \frac{e^{(k+1)}}{[e^{(k)}]^2} = \frac{1}{2}\phi''(x^*).$$

Je calcule $\phi''(x^*) = f''(x^*)/f'(x^*)$ d'où le résultat

$$\lim_{k\to\infty} \frac{e^{(k+1)}}{[e^{(k)}]^2} = \frac{f''(x^*)}{2f'(x^*)}. \tag{5}$$

Je peux donc énoncer le théorème suivant.

> **Théorème.** Si f, f' et f'' sont continues dans un voisinage de x^* et si $f(x^*) = 0$ et $f'(x^*) \neq 0$, si $x^{(0)}$ est choisi assez près de x^*, alors la suite définie par (4) converge vers x^*, avec une vitesse de convergence donnée par (5) ; la convergence est dite quadratique.

6 Méthode de la sécante

La méthode de la sécante est une méthode «à deux points» : connaissant les approximations $x^{(k-1)}$ et $x^{(k)}$ de la racine, je vais calculer une meilleure approximation $x^{(k+1)}$. Le principe en est simple. La corde qui joint les points de coordonnées $[x^{(k-1)}, f(x^{(k-1)})]$ et $[x^{(k)}, f(x^{(k)})]$ coupe l'axe des x en un point d'abscisse $x^{(k+1)}$. Je trouve facilement l'expression de cette abscisse

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}.$$

Ce procédé n'est pas très différent de la méthode de Newton. En effet, la fraction au second membre peut être considérée comme une approximation de $1/f'(x^{(k)})$. L'analyse de la convergence est cependant un peu plus compliquée et pourra être abordée en exercice en exercice.

7 Résolution de systèmes d'équations

Je ne ferais qu'effleurer le sujet vaste et compliqué des systèmes d'équations non linéaires à plusieurs variables. Dans le cas de deux dimensions, je cherche les solutions du système :

$$\begin{cases} f(x,y) = 0, \\ g(x,y) = 0. \end{cases} \tag{6}$$

Chaque équation définit une courbe du plan (x,y) ; résoudre le système équivaut donc à chercher les points d'intersection de ces deux courbes. Je recommande vivement une étude graphique sommaire de ces deux équations avant toute recherche de racine par le calcul ; en effet, quelque soit l'algorithme utilisé, la convergence dépendra fortement de la qualité de l'approximation initiale.

Je vais présenter une généralisation de la méthode de Newton. Je suppose connue une approximation de la solution, soit (x_0, y_0). Je peux toujours poser :

$$x = x_0 + u \quad ; \quad y = y_0 + v.$$

Je substitue dans (6) et j'effectue un développement de Taylor du premier ordre à deux variables :

$$f(x_0 + u, y_0 + v) = f(x_0, y_0) + uf_x + vf_y + \cdots = 0,$$
$$g(x_0 + u, y_0 + v) = g(x_0, y_0) + ug_x + vg_y + \cdots = 0,$$

où les dérivées partielles f_x, f_y, g_x et g_y sont calculées au point (x_0, y_0). Je suppose que les variations de f et g soient assez lentes pour que je puisse négliger les termes d'ordre supérieur. En isolant les termes en u et v, je trouve

$$\begin{cases} uf_x + vf_y &= -f(x_0, y_0), \\ ug_x + vg_y &= -g(x_0, y_0). \end{cases} \tag{7}$$

Les équations (7) constituent un système de deux équations linéaires à deux inconnues, u et v, dont la solution est immédiate. Il est maintenant facile d'imaginer l'algorithme de Newton à deux dimensions. J'obtiendrais une approximation encore meilleure de la solution en itérant ce procédé

$$x_1 = x_0 + u \quad ; \quad y_1 = y_0 + v.$$

À l'étape suivante de l'itération, je remplace (x_0, y_0) par (x_1,y_1) et ainsi de suite, jusqu'à satisfaire à un critère de convergence.

Ce formalisme se généralise facilement à un nombre quelconque d'équations ; il est alors commode d'utiliser une notation matricielle. Soit $\mathbf{f} \in \Re^n$ un vecteur de coordonnées $f_i(\mathbf{r})$; je suppose aussi que $\mathbf{r} \in \Re^n$. Le système d'équations à résoudre s'écrit :

$$\mathbf{f} = 0.$$

Si $\mathbf{r}_0$ est une approximation de la solution, je pose $\mathbf{r} = \mathbf{r}_0 + \mathbf{u}$. La correction $\mathbf{u}$ est alors définie par l'équation

$$\mathbf{f}(\mathbf{r}_0) + \sum_1^n u_i \frac{\partial\, \mathbf{f}}{\partial\, u_i} = 0$$

ou encore, en définissant la "matrice jacobienne" $\mathbf{J}$, d'éléments $J_{ij} = \partial f_i / \partial u_j$:

$$\mathbf{u} = -\mathbf{J}^{-1}\mathbf{f}(\mathbf{r}_0)$$
$$\mathbf{r}_1 = \mathbf{r}_0 + \mathbf{u}$$
$$\mathbf{r}_{n+1} = \mathbf{r}_n - \mathbf{J}^{-1}\mathbf{f}(\mathbf{r}_n)$$

La convergence est rapide si le point de départ est proche de la solution ; dans le cas contraire, nous ne sommes pas assurés de voir l'algorithme converger.

Exemple. Soit le système de deux équations :

$$f(x,y) = x^2 + y^2 - 4 = 0,$$
$$g(x,y) = e^x + y - 1 = 0.$$

Il est très simple de construire les deux courbes représentant les équations $f = 0$ et $g = 0$; elles se coupent en deux points voisins de (1,-2) et (-2,1) ; il n'y a pas d'autre solution. J'ai écrit un programme de résolution par la méthode de Newton à deux dimensions, que vous trouverez ci-dessous.

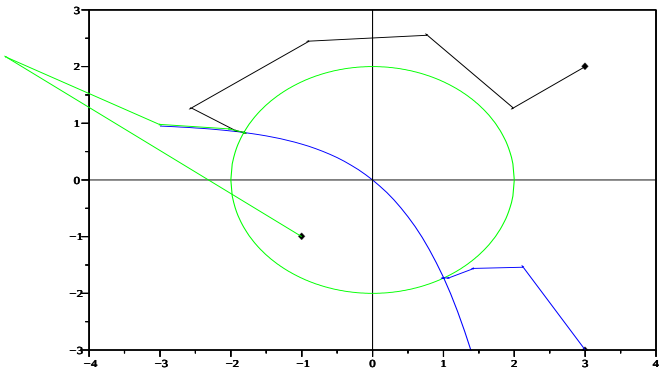
```
//algorithme de newton a deux variables
function u = f(x,y)
u = x*x + y*y - 4
endfunction
function u = dfdx(x,y)
u = 2*x;
endfunction
function u = dfdy(x,y)
u = 2*y
endfunction
function u = g(x,y)
u = exp(x) + y - 1;
endfunction
function u =  dgdx(x,y)
u = exp(x)
endfunction
function u = dgdy(x,y)
u = 1;
endfunction

tol = 1e-6; nitmax = 20;
xmin = -4; xmax = 4; ymin = -3; ymax = 3;
xset("window",0),xbasc(0)
xt = -3:0.1:3;
yt1 = 1 - exp(xt);
plot2d(xt,yt1,rect=[xmin,ymin,xmax,ymax],style=2)
xpoly([xmin,xmax],[0,0]), xpoly([0,0],[ymin,ymax])
xt = -2:0.02:2;
yt2 = sqrt(4 - xt.^2);
plot2d(xt,yt2,3,"000");
yt2 = -sqrt(4 - xt.^2);
plot2d(xt,yt2,3,"000");

for j = 1:3
a = input("abscisse initiale: "); b = input("ordonnee initiale: ");
nit = 1; xx(1) = a; x = a; yy(1) = b; y = b;
xset("mark size", 2), plot2d(a,b, -4)
u = 1; v = 1;
```

```
while ( ((abs(u) > tol) | (abs(v) > tol)) & nit < nitmax )
    delta = dfdx(x,y)*dgdy(x,y)-dfdy(x,y)*dgdx(x,y);
    u = (g(x,y)*dfdy(x,y)-f(x,y)*dgdy(x,y))/delta;
    v = (f(x,y)*dgdx(x,y)-g(x,y)*dfdx(x,y))/delta;
    nit = nit + 1
    x = x + u, y = y + v,
    xx(nit) = x; yy(nit) = y;
    xarrows([xx(nit-1),xx(nit)],[yy(nit-1),yy(nit)],0.05,j)
end
end
```

La figure montre les itérations successives. La convergence est rapide, bien que le point de coordonnées(x_i, y_i) ait tendance à «explorer» une assez grande région du plan. Comme toujours avec la méthode de Newton, le comportement dépend très fortement du point de départ.



8 Racines des polynômes

Soit $p(x)$ un polynôme de degré n à coefficients réels. On désigne parfois une équation comme $p(x) = 0$ sous le nom d'équation algébrique. Le théorème fondamental de l'algèbre énonce que $p(x)$ a n racines, réelles ou complexes conjuguées deux à deux. Pour de nombreuses applications, il est nécessaire de déterminer toutes les racines de $p(x)$.

8.1 Séparation des racines

Avant de chercher les valeurs numériques des zéros d'un polynôme, il est commode d'avoir une idée de leur localisation. Si $[a_0, a_1, \ldots, a_n]$ est la suite des coefficients de $p(x)$ supposés réels (certains pouvant être nuls), j'appelle ν le nombre de changements de signe dans la suite des $\{a_i\}$ (en ignorant les coefficients nuls). Je désigne par k le nombre de zéros réels positifs de $p(x)$, comptés selon leur multiplicité (une racine double compte pour 2, etc.). Je peux alors énoncer la règle de Descartes :

$k \leq \nu$ et de plus $\nu - k$ doit être pair.

Exemple. Les coefficients de $x^6 - x - 1$ présentent un changement de signe ($\nu = 1$) ; k vaut donc 0 ou 1, mais la valeur 0 est à rejeter puisqu'alors $\nu - k = 1$. Ce polynôme a donc une racine réelle positive.

Je peux aussi trouver le nombre de racines réelles négatives : il suffit d'appliquer la règle de Descartes au polynôme $q(x) = p(-x)$. Le polynôme de l'exemple précédent admet un zéro réel négatif.

8.2 Suites de Sturm

Une localisation plus précise des racines d'un polynôme $p(x)$ de degré n peut être obtenue en construisant une «suite de Sturm» basée sur p. Je commence donc par définir ce qu'est une suite de Sturm.

Soit $p_0(x), p_1(x), \ldots, p_m(x)$ une séquence de polynômes ; elle constitue une suite de Sturm si

- Les zéros réels de $p_0(x)$ sont simples.
- $p_1(\alpha)p_0'(\alpha) < 0$ si α est un zéro réel de p_0.
- Pour $k = 1, 2 \ldots, m-1$, $p_{k-1}(\alpha)p_{k+1}(\alpha) < 0$ si α est un zéro réel de $p_k(x)$.
- Le dernier polynôme, $p_m(x)$ n'a pas de zéros réels.

Comment construire cette suite ? À l'aide de la relation de récurrence assez simple que j'expose maintenant. Je pose

$$p_0(x) \equiv p(x) \qquad ; \qquad p_1(x) \equiv -p_0'(x)$$

et

$$p_{k-1}(x) = q_k(x)p_k(x) - p_{k+1}(x) \text{ avec } \deg p_k > \deg p_{k+1}. \tag{8}$$

Si les p_k étaient des nombres, vous auriez sûrement reconnu l'algorithme d'Euclide pour la construction du PGCD. Le concept de PGCD se transpose facilement aux polynômes. Comme le degré des polynômes décroît, l'algorithme s'arrête au bout de $m \leq n$ étapes :

$$p_{m-1}(x) = q_m(x)p_m(x), \quad p_m(x) \neq 0.$$

Le dernier polynôme, p_m est un plus grand commun diviseur de p_0 et p_1. Comme $p = p_0$ n'a que des zéros simples, p_0 et $p_1 = -p_0'$ n'ont pas de diviseurs communs (ne sont jamais nuls simultanément) et, par conséquent, p_m n'a pas de zéros réels. Si $p_k(\alpha) = 0$, la relation (8) implique que $p_{k-1}(\alpha) = -p_{k+1}(\alpha)$. Il pourrait se faire que $p_{k+1}(\alpha)$ soit nul ; mais alors, toujours d'après la relation de récurrence, tous les $p_k(\alpha)$ seraient nuls, y compris p_m, ce qui est contraire au résultat précédent. En tenant compte des définitions de p_0 et p_1, vous vous apercevez que les $\{p_k\}$ satisfont à toutes les propriétés qui définissent une suite de Sturm.

J'énonce maintenant le théorème de Sturm :

> **Théorème.** Le nombre de racines réelles de $p(x) \equiv p_0(x)$ dans l'intervalle $a \leq x < b$ est égal à $w(b) - w(a)$ où $w(x)$ est le nombre de changements de signe dans la suite $p_0(x), p_1(x), \ldots, p_m(x)$ au point x.

Je ne donnerai pas ici la démonstration, simple mais assez longue : il faut examiner en détail les changements de signe de chaque polynôme au voisinage d'un zéro de l'un d'eux.

Exemple. Soit $p(x) = x^5 - x^3 - x - 1$. La suite de Sturm de p est

$$\begin{aligned} p_0(x) &= x^5 - x^3 - x - 1, \\ p_1(x) &= -5x^4 - 3x^2 + 1, \\ p_2(x) &= \frac{2}{5}x^3 + \frac{4}{5}x + 1, \\ p_3(x) &= -13x^2 - \frac{25}{2}x - 1, \\ p_4(x) &= -\frac{385}{338}x - \frac{174}{169}, \\ p_5(x) &= \frac{47827}{148225}. \end{aligned}$$

Ici, $m = 5$. Je ne considère que les cas $x = -\infty, 0, \infty$; je construis un tableau de signes

x	$-\infty$	0	∞
p_0	-	-	+
p_1	-	+	-
p_2	-	+	+
p_3	-	-	-
p_4	+	-	-
p_5	+	+	+
$w(x)$	1	3	4

La suite présente un seul changement de signe lorsque x est très grand négatif, trois changements pour $x = 0$ et 4 pour x grand positif. Il y a donc deux zéros dans l'intervalle $[-\infty, 0[$ et un dans $[0, \infty[$.

8.3 Division des polynômes

Je reviens plus en détail, dans ce paragraphe, sur la division des polynômes, qui a été considérée comme bien connue au paragraphe précédent. Étant donné un polynôme $p(x)$ de degré n (le dividende) et un polynôme $b(x)$ de degré $m < n$ (le diviseur), on sait trouver deux polynômes, le quotient $q(x)$ et le reste $r(x)$ tels que

$$p(x) = b(x)q(x) + r(x), \text{ avec deg } r < \text{ deg } b. \tag{9}$$

Remarquez que le degré de $q(x)$ est imposé $(n - m)$ mais ne joue aucun rôle dans la suite. Avec cette définition, la division des polynômes est très semblable à celle des nombres, pour laquelle le reste doit être inférieur au diviseur. Je considère maintenant un cas particulier, celui où le diviseur est de la forme $b(x) = x - \alpha$. Le reste est alors une constante (polynôme de degré < 1). Je peux écrire

$$p(x) = (x - \alpha)q(x) + r$$

d'où je tire, en faisant $x = \alpha$,

$$p(\alpha) = r. \tag{10}$$

en d'autres termes, la valeur numérique de p en $x = \alpha$ est le reste de la division de p par $x - \alpha$. Ce résultat a une conséquence importante. Si a est un zéro de p, donc si $p(\alpha) = 0$, alors $r = 0$, ce qui signifie qu'un polynôme qui s'annule en $x = \alpha$ est divisible par $x - \alpha$. J'ai déjà expliqué (chapitre

1) comment calculer r ou $p(a)$, mais je reprendrai ce calcul plus bas. Je peux aussi obtenir une expression simple de la valeur numérique de la dérivée de p. Je sais que

$$p'(x) = q(x) + (x - \alpha)q'(x)$$

et, en faisant à nouveau $x = \alpha$,

$$p'(\alpha) = q(\alpha). \tag{11}$$

Comme vous le verrez, il est facile de calculer $q(\alpha)$. Des équations analogues existent pour les dérivées d'ordre supérieur.

Je vais maintenant construire le polynôme $q(x)$ par identification. Je note $c_0, c_1, \ldots, c_{n-1}$ les coefficients des puissances croissantes de x dans q. Ces nombres doivent satisfaire, quelque soit x, à la relation

$$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = (x - \alpha)(c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0).$$

En identifiant les coefficients des puissances successives de x, j'obtiens

$$a_n = c_{n-1} \quad ; \quad a_{n-1} = c_{n-2} - \alpha c_{n-1}; \ldots$$

$$a_1 = c_0 - \alpha c_1 \quad ; \quad a_0 = r - \alpha c_0$$

soit encore :

$$\begin{aligned} c_{n-1} &= a_n \\ c_{n-2} &= a_{n-1} + \alpha c_{n-1}, \\ \cdots &\quad \cdots \\ c_0 &= a_1 + \alpha c_1, \\ r &= a_0 + \alpha c_0. \end{aligned} \tag{12}$$

Vous voyez que r, qui est aussi la valeur numérique de $p(\alpha)$, s'obtient au moyen de $n-1$ multiplications. En fait la relation de récurrence précédente est équivalente à la règle suivante. J'écris le polynôme $p(x)$ sous la forme :

$$p = (((...(a_n x + a_{n-1})x + a_{n-2})x + ... + a_1)x + a_0, \tag{13}$$

puis je fais $x = \alpha$ avant d'effectuer les calculs. La parenthèse intérieure contient alors c_{n-2}, la suivante c_{n-3}, jusqu'à r. Vous avez reconnu la méthode de Horner. Je donne un exemple, qui fait un peu double emploi avec celui du chapitre 1.

Exemple. Je vais calculer la valeur de $p = 2x^5 - 71x^3 - 8x^2 + 12x + 3$ pour $x = 6$, ce qui revient à diviser p par $x - 6$.Il est commode de disposer le calcul comme ci-dessous. Seuls les coefficients des puissances décroissantes de x sont nécessaires, mais il faut les faire figurer tous, mêmes ceux qui sont nuls.

p	2	0	-71	-8	12	3
q	2	$6 \times 2 + 0 = 12$	$6 \times 12 - 71 = 1$	$6 \times 1 - 8 = -2$	$6 \times -2 + 12 = 0$	$6 \times 0 + 3 = 3$

Le reste est $r = 3$, le quotient est $q = 2x^4 + 12x^3 + x^2 - 2x$. La valeur numérique de $p'(6)$ est $q(6)$ qui s'obtient par le même procédé.

q	2	12	1	-2	0
	2	$6 \times 2 + 12 = 24$	$6 \times 24 + 1 = 145$	$870 - 2 = 868$	$6 \times 868 = 5208$

ce que vous pouvez vérifier par un calcul direct.

Lorsque l'on traite à la main un seul polynôme, les formes (12) et (13) sont équivalentes mais, lorsque l'on doit faire de nombreux calculs, il devient intéressant d'écrire un programme général. Je donne ci-dessous le squelette d'un programme qui calcule p et p' à partir du tableau `a` des coefficients de p et de la valeur de x.

```
p = a[n], dp = 0.0;
for i de n-1 à 0
     dp = dp*x + p;
     p = p*x+a[i];
```

Lorsque la boucle se termine, `p` contient la valeur numérique du polynôme (accumulée selon la méthode de Horner) et `dp` celle de la dérivée.

8.4 La méthode de Newton pour les polynômes

Je dispose maintenant d'un algorithme pratique de calcul des valeurs numériques d'un polynôme et de sa dérivée : c'est tout ce qu'il faut pour mettre en oeuvre la méthode de Newton. En voici un exemple : la recherche du zéro de $p = x^3 - x^2 + 2x + 5$ voisin de -1. Les coefficients ont été rangés en colonne pour gagner de la place.

x	p	q	p'	p/p'
-1	1	1	1	
	-1	-2	-3	
	2	4	**7**	
	5	<u>1</u>		
				0,142857
-1,142857	1	1	1	
	-1	-2,142857	-3,285714	
	2	4.448979	**8,204081**	
	5	<u>-0,084546</u>		
				-0,010305
-1,132552	1	1	1	
	-1	-2,132552	-3,265104	
	2	4,415226	**8,113126**	
	5	<u>-0.000473</u>		
				-0.000058
-1,132494				

Les nombres soulignés sont les valeurs de p, alors que les valeurs de p' figurent en caractères gras. On voit que la convergence (caractérisée par le terme correctif (p/p') est très rapide. Il est possible d'étendre cet algorithme aux racines complexes ou aux polynômes à coefficients complexes.

p est de degré 3 et admet donc 3 zéros ; comment pourrais-je les trouver ? Très simplement : il suffit de diviser le polynôme de départ par $x + 1,132494$. La division doit se faire exactement, puisque -1,132494 est racine de $p(x)$; le polynôme quotient est ici du second degré et ses zéros s'obtiennent sans problème. Ce procédé s'appelle la déflation. Dans le cas général d'un polynôme $p(x)$ de degré n, dès que j'ai déterminé une racine x^* (par la méthode de Newton par exemple), je divise $p(x)$ par $x - x^*$ pour obtenir un polynôme de degré $n - 1$. Je répète ce procédé jusqu'au

degré 2 ou 1. Si n est un tant soit peu grand, les erreurs d'arrondi risquent, en s'accumulant, de gâcher ce programme : le calcul doit être mené avec toute la précision possible.