Inversion de fonction I

On résout ici le problème de l'inversion d'une fonction réelle de variables réelles. Inverser la fonction

$$f : \mathbb{R}^3 \to \mathbb{R} : (x_1, x_2, x_3) \mapsto f(x_1, x_2, x_3)$$

par rapport à x_2 consiste à construire une fonction

$$g : \mathbb{R}^3 \to \mathbb{R} : (x_1, u, x_3) \mapsto g(x_1, u, x_3)$$

telle que

$$g(x_1, f(x_1, x_2, x_3), x_3) = x_2 \quad \text{et} \quad f(x_1, g(x_1, u, x_3), x_3) = u\,,$$

ou encore

$$g(x_1, u, x_3) = x_2 \quad \text{et} \quad f(x_1, x_2, x_3) = u\,,$$

pour toutes valeurs adéquates de x_2 et u. Le problème est mathématiquement difficile, puisque l'inverse n'existe pas toujours, mais devient plus simple dans le cas où la fonction à inverser est continue et strictement monotone (croissante ou décroissante) par rapport à l'argument (ici x_2) sur lequel porte l'inversion.

Inversion de fonction II

On ramène le cas général (n arguments, $p$ième argument) au cas particulier d'un premier argument sur lequel porte l'inversion, et d'une liste d'autres arguments. Des opérateurs comme `in` et `out` font la conversion, ici dans le cas $n = 3$, $p = 2$:

```
(define in
  (lambda (f)
    (lambda (x l) (f (car l) x (cadr l)))))

(define out
  (lambda (h)
    (lambda (x1 x2 x3) (h x2 (list x1 x3)))))

(define f (lambda (a b c) (+ a (* b c))))

(f 34 56 23)                    1322
((in f) 56 '(34 23))            1322
((out (in f)) 34 56 23)         1322
```

Inversion de fonction III

Fonction $(x, \ell) \mapsto f(x, \ell)$, fonction inverse $(u, \ell) \mapsto g(u, \ell)$
Les deux fonctions sont croissantes en leur premier argument.

Approximations successives : soit $(x_0, x_1, \ldots) \longrightarrow g(u, \ell)$
Si $f(x_n, \ell) > u$, alors $x_n > g(u, \ell)$ et on choisit $x_{n+1} < x_n$;
si $u > f(x_n, \ell)$, on choisit $x_{n+1} > x_n$.

Deux variantes : $\mathbb{R} \to \mathbb{R}$ et $\mathbb{R}^+ \to \mathbb{R}^+$. On passe de l'une à l'autre par transformation exponentielle ou logarithmique. La seconde variante est développée ici.

Technique de la bissection. On maintient un intervalle $[m, M]$ dans lequel la valeur recherchée se trouve, on le rétrécit à chaque itération. Au départ, l'intervalle est très grand, par exemple $m = 10^{-6}$ et $M = 10^7$. A chaque étape, on calcule la moyenne (géométrique) μ des bornes de l'intervalle puis la valeur $f(\mu, \ell)$. En fonction de cette valeur, on décide de s'arrêter, si l'écart entre $f(\mu, \ell)$ et u n'excède pas une certaine quantité ε_y ou de continuer soit avec l'intervalle $[m, \mu]$, soit avec l'intervalle $[\mu, M]$. Chaque étape a pour effet de réduire la longueur de l'intervalle (de moitié, pour la première variante) et on peut s'arrêter dès que cette longueur devient moindre qu'une certaine quantité ε_x.

Inversion de fonction IV

Variables globales, fonctions auxiliaires :

```
(define *min* 1.e-6)            (define *eps-x* 1.e-12)
(define *max* 1.e+7)            (define *eps-y* 1.e-12)
(define mu (lambda (a b) (sqrt (* a b))))
(define prox (lambda (a b eps) (< (abs (- a b)) eps)))

(define inv+    ;; Cas où la fonction f est croissante
  (lambda (f)
    (lambda (u l)
      (i+ f u l *min* *max* *eps-x* *eps-y*))))

(define i+
  (lambda (f u l x0 x1 epsx epsy)
    (cond ((prox x0 x1 epsx) (mu x0 x1))
          ((prox (f (mu x0 x1) l) u epsy) (mu x0 x1))
          ((< (f (mu x0 x1) l) u) (i+ f u l (mu x0 x1) x1 epsx epsy))
          (else (i+ f u l x0 (mu x0 x1) epsx epsy)))))
```

Inversion de fonction V

L'opérateur `inv+` prend comme argument une fonction `f` croissante en son premier argument et renvoie la fonction inverse.

Le prédicat `prox` prend comme arguments deux réels a et b et un réel positif ε ; il renvoie vrai si $|a - b| < \varepsilon$.

Fonction `i+`

> *Si la fonction* $f : (\mathbb{R} \times \mathbb{R}^k) \to \mathbb{R}$ *est croissante en son premier argument et si l'unique solution* x *de l'équation* $f(x, \ell) = u$ *appartient à l'intervalle* $[x_0 : x_1]$, *alors* $i^+(f, u, \ell, x_0, x_1, \varepsilon_x, \varepsilon_y)$ *est un nombre* x' *proche de* x, *au sens que* x' *ne s'écarte pas de* x *de plus de* ε_x *ou que* $f(x', \ell)$ *ne s'écarte pas de* u *de plus de* ε_y.

On définit de manière analogue un opérateur `inv-` pour inverser les fonctions décroissantes, qui fera appel à l'opérateur auxiliaire `i-` ; ce dernier ne diffère de `i+` que par la permutation des comparateurs `<` et `>` dans les conditions des clauses contenant les appels récursifs.

Inversion de fonction VI

Un dernier point intéressant consiste à modifier `i+` de manière à éviter l'évaluation multiple des expressions `(mu x0 x1)` et `(f (mu x0 x1) l)`.

```
(define i+         ;; avec évaluation multiple
  (lambda (f u l x0 x1 epsx epsy)
    ((lambda (aux1)
       (cond ((prox x0 x1 epsx) aux1)
             ((prox (f aux1 l) u epsy) aux1)
             ((< (f aux1 l) u) (i+ f u l aux1 x1 epsx epsy))
             (else (i+ f u l x0 aux1 epsx epsy))))
     (mu x0 x1))))

(define i+         ;; sans évaluation multiple de aux1
  (lambda (f u l x0 x1 epsx epsy)
    (let ((aux1 (mu x0 x1)))
       (cond ((prox x0 x1 epsx) aux1)
             ((prox (f aux1 l) u epsy) aux1)
             ((< (f aux1 l) u) (i+ f u l aux1 x1 epsx epsy))
             (else (i+ f u l x0 aux1 epsx epsy))))))
```

Inversion de fonction VII

On peut réutiliser la même technique pour éviter la double évaluation de l'expression `(f aux1 l)` ; on obtient ainsi, sans utiliser `let` :

```
(define i+       ;; sans évaluation multiple de aux1 et aux2
  (lambda (f u l x0 x1 epsx epsy)
    ((lambda (aux1)
       ((lambda (aux2)
          (cond ((prox x0 x1 epsx) aux1)
                ((prox aux2 u epsy) aux1)
                ((< aux2 u) (i+ f u l aux1 x1 epsx epsy))
                (else (i+ f u l x0 aux1 epsx epsy))))
        (f aux1 l)))
     (mu x0 x1))))
```

Inversion de fonction VIII

Version utilisant `let*` :

```
(define i+        ;; sans évaluation multiple de aux1 et aux2
  (lambda (f u l x0 x1 epsx epsy)
    (let* ((aux1 (mu x0 x1))
           (aux2 (f aux1 l)))
      (cond ((prox x0 x1 epsx) aux1)
            ((prox aux2 u epsy) aux1)
            ((< aux2 u) (i+ f u l aux1 x1 epsx epsy))
            (else (i+ f u l x0 aux1 epsx epsy))))))
```

Inversion de fonction IX

$$f_1 : (x, [a, b]) \mapsto x(a + x(b + x))$$

$$f_1 : (x, [a, b]) \mapsto ax + bx^2 + x^3$$

```
(define (f1 x l) (* x (+ (car l) (* x (+ (cadr l) x)))))

(map (lambda (x) (f1 x '(1 1))) '(0 1 2 3 4 5 6))    (0 3 14 39 84 155 258)

(map (lambda (u) ((inv+ f1) u '(1 1))) '(0 3 14 39 84 155 258))
     (1.0000004460455906e-6 ;; précision médiocre
      1.000000000001634     ;; précision excellente
      2.00000000000033      ;; précision excellente
      3.00000000000123      ;; précision excellente
      3.9999999999999036    ;; précision excellente
      5.00000000000184      ;; précision excellente
      6.00000000000003)     ;; précision excellente
```