Université Bordeaux 1 Licence S&T Mention Informatique S6 2007/2008

UE INF 353T- Programmation Fonctionnelle et Symbolique
Contrôle continu 1

Tous documents autorisés. Mercredi 13 Février 2008
Durée : 1h.
Le barême est donné à titre **indicatif**. Vous devez écrire vos solutions dans un fichier cc1.lisp et envoyer ce fichier à l'adresse : `ges@labri.fr` avec comme sujet "CCPFS".

Exercice 1 *(10pts)*

On reprend les expressions du TD2 :

```
(defparameter *pas* (/ 1 10000))
(defun DERIVE (f)
  (lambda (u) (/(-(funcall f (+ u *pas*))(funcall f (+ u 0))) *pas*)))
```

1- Ecrire une fonction `make-trinome(a b c)` qui retourne une fonction trinome associant à tout argument numérique x la valeur $ax^2 + bx + c$.
2- Ecrire une fonction `c0(trin)` qui accepte comme argument une fonction trinôme et retourne un nombre qui est le coefficient de degré 0 de ce trinôme. Par exemple, on doit avoir

```
CL-USER> (c0 (make-trinome 2 1 5))
5
```

3- Ecrire une fonction `c1(trin)` qui accepte comme argument une fonction trinôme et retourne un nombre qui est le coefficient de degré 1 de ce trinôme. Par exemple, on pourra avoir, vu les erreurs commises par la fonctionnelle `DERIVE`, un résultat approché tel que

```
CL-USER> (c1 (make-trinome 2 1 5))
500001/500000
```

4- Ecrire une fonction `c2(trin)` qui accepte comme argument une fonction trinôme et retourne un nombre qui est le coefficient de degré 2 de ce trinôme. On pourra avoir, par exemple :

```
CL-USER> (c2 (make-trinome 2 1 5))
2
```

5- En admettant que la fonction `c2` est exacte, écrire une fonction `affine-p(trin)` qui accepte comme argument une fonction trinôme et renvoie vrai (resp. faux) si ce trinôme est de degré 0 ou 1 (resp. s'il est de degré 2). Donner un exemple d'appel à `affine-p`.

Exercice 2 *(15pts)*
On reprend les expressions utilisées au TD4 :

```
(defun point-in-zone-p (point zone)
  (funcall zone point))
(defun make-disk (radius)
  (lambda (p)
    (<= (abs p) radius)))
(defun move-zone (zone vector)
  (lambda (p)
    (point-in-zone-p (- p vector) zone)))
```

On rappelle quelques notions de géométrie plane :
Etant donnés $u_1 = (x_1, y_1), u_2 = (x_2, y_2) \in \mathbb{R}^2$, le *produit scalaire* de u_1 par u_2 est donné par :

$$(u_1 \mid u_2) := x_1 \cdot x_2 + y_1 \cdot y_2.$$

La projection (orthogonale) de $P = (x, y)$ sur la droite passant par 0 et dirigée par u_1 est donnée par :

$$proj_{u_1}(P) := \frac{(P \mid u_1)}{(u_1 \mid u_1)} \cdot u_1.$$

1- Ecrire une fonction `scal(u v)` qui prenne en arguments deux nombres complexes et retourne le nombre réel qui est leur produit scalaire. Par exemple :

```
CL-USER> (scal #C(0 1) #C(2 1))
1
```

2- Ecrire une fonction `proj(v)` qui prenne en argument un nombre complexe et retourne la fonction qui associe à tout point du plan (représenté par un nombre complexe) sa projection orthogonale sur la droite passant par 0 et dirigée par v. Par exemple :

```
CL-USER> (funcall (proj #C(1 1)) #C(2 0))
#C(1 1)
```

3- Ecrire une fonction `sym(v)` qui prenne en argument un nombre complexe et retourne la fonction qui associe à tout point P du plan (représenté par un nombre complexe) le point symétrique de P par rapport à la droite passant par 0 et dirigée par v. Par exemple :

```
CL-USER> (funcall (sym #C(1 1)) #C(5 0))
#C(0 5)
```

Aide : Si l'on note $proj_v$ (resp. s_v) la projection sur (resp. la symétrie par rapport à) la droite dirigée par v, on a, pour tout point P,

$$s_v(P) = 2proj_v(P) - P$$

(voir la figure 1).
4- Ecrire une fonction `symz(v zone)` qui retourne la zone Z' symétrique par rapport à la droite passant par 0 et dirigée par v, de la zone passé en second argument. Donner un exemple d'appel à cette fonction `symz`.

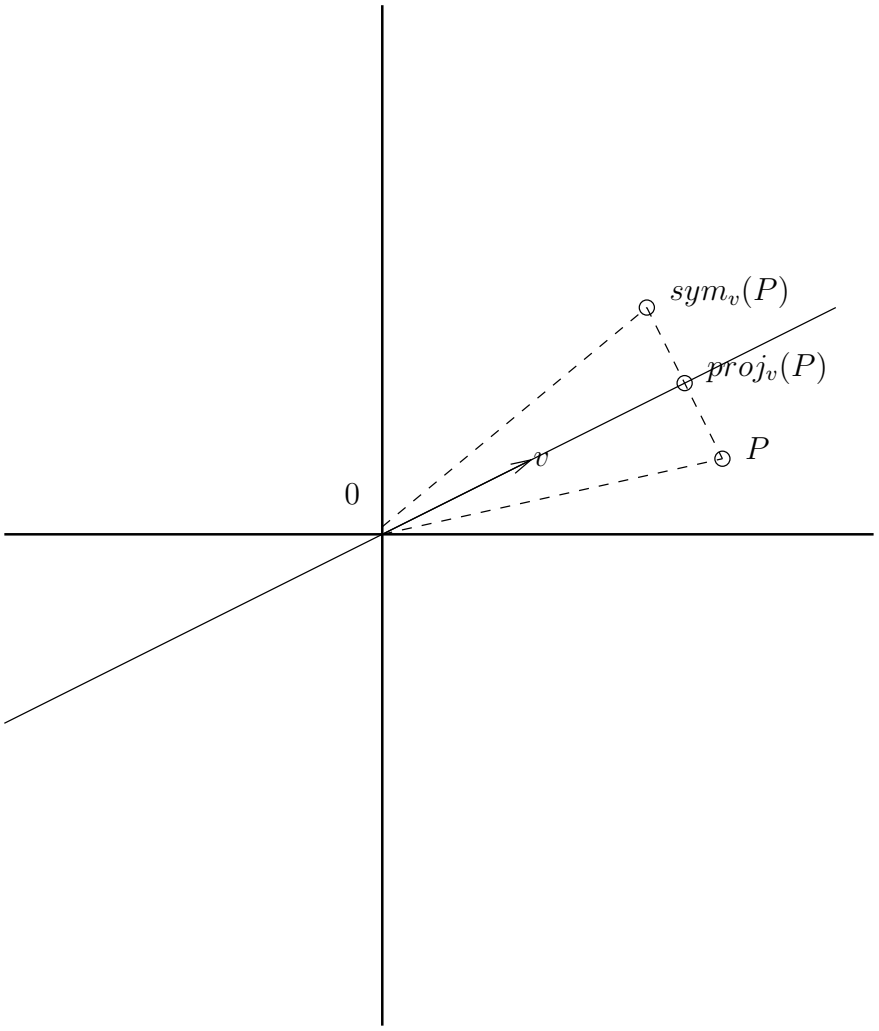

FIG. 1 – Projection et symétrie.