

**Corrigé du sujet 1 du Concours Mathématiques et Physique, Physique et  
Chimie, Biologie et Géologie & Technologie  
Epreuve informatique**

**EXERCICE1 (3 points)**

1) (0,25 point)

```
> restart;  
> f:=x->2*int(exp(-t^2), t=0..x)/sqrt(Pi);
```

2) (0,5 point)

```
> paire := evalb(f(-x)=f(x)):impaire :=evalb(f(-x)=-f(x)):  
if paire then print (`f est paire`) elif impaire then print  
(`f est impaire`) else print(`f est sans parite`); fi;
```

#ou encore

```
> if f(-x)=f(x) then print (`f est paire`) elif f(-x)=-f(x)  
then print (`f est impaire`) else print(`f est sans parite`);  
fi;
```

3) (0,25 point)

```
> L:=limit(f(x), x=infinity);
```



4) (0,25 point)

```
> g:=D(f);
```

5) (0,5 point)

```
> DV:=series(f(x), x=0, 10); #ou encore DV:=taylor(f(x), x=0, 10);
```

6) (0,25 point)

```
> DA:=asympt(f(x), x);
```

7) (0,5 point)

```
> H:=sqrt(Pi)/2*exp(x^2)*f(x);
```

```
> h:=unapply(H, x);
```

8) (0,5 point)

```
> h1:=D(h);
```

```
> z:=simplify(h1(x)-2*x*h(x));
```

```
> evalb(z=1);
```

**EXERCICE2 (4 points)**

```
> restart:
```

> with(linalg):

Warning, the protected names norm and trace have been redefined and unprotected

1) (0,25 point)

> A:=matrix(3,3,[0,2,2,-2,5,3,2,-4,-2]);

#ou bien > A:=matrix([[0,2,2],[-2,5,3],[2,-4,-2]]);

2) (0,25 point)

> DT:=det(A);

3) (0,25 point)

> NOY:=kernel(A);

4) (0,25 point)

> IM:=colspace(A);

5) (0,25 point)

> POL:=charpoly(A,lambda);

6) (0,25 point)

> VALP1:=solve(POL=0,lambda):# ou bien

VALP1:=solve(POL=0):# ou bien

VALP1:=solve(POL);

7) (0,25 point)

> VALP2:=eigenvals(A);

8) (0,5 point)

> VECTP:=eigenvectors(A):# ou bien VECTP:=eigenvects(A);

C'est une séquence de liste dont le nombre est égal au nombre de valeurs propres. Chacune de ces listes est constituée de trois termes. Le premier est la valeur propre, le second est la multiplicité associée à cette valeur propre, et le troisième est un ensemble constitué par un ou plusieurs vecteurs propres associés à cette valeur propre.

9) (0,5 point)

> V1:=op( op(1,[VECTP ] ) [3] );

> V2:=op( op(2,[VECTP ] ) [3] );

> V3:=op( op(3,[VECTP ] ) [3] );

#ou bien V1:=op(op(3,VECTP[1]));

V2:=op(op(3,VECTP[2]));V3:=op(op(3,VECTP[3]));

10) (0,25 point)

```

> P:=concat(V1,V2,V3); #ou bien
> P:=transpose(matrix([eval(V1),eval(V2),eval(V3)]));#ou bien
> P:=transpose(matrix([V1,V2,V3]));#ou P1:=augment(V1,V2,V3);

```

11) (0,25 point)

```

> INVP:=inverse(P);# ou bien INVP:=evalm(P^(-1));

```

12) (0,25 point)

```

> d:=diag(VALP1); #ou bien d:=diag(VALP2);

```

13) (0,25 point)

```

> dn:=map(x->x^n,d);

```

14) (0,25 point)

```

> An :=multiply(P,dn,INVP):# ou bien
> An :=evalm(P&dn*INVP):# ou bien
> An:=evalm(P&dn*P^(-1));

```

## PROBLEME

### PARTIE A (7 points)

#### 1) (0,5 point)

```

FONCTION f(x : REEL) : REEL
DEBUT

```

```

    f ← tan(x) - x { ou encore Retourner(tan(x)-x) }

```

```

FIN

```

#### 2) (0,5 point)

```

FONCTION fprime (x : REEL) : REEL
CONSTANTE

```

```

    h=0.001

```

```

DEBUT

```

```

    fprime ← ( f(x+h)-f(x-h) ) / ( 2*h ) {ou encore Retourner( ( f(x+h)-f(x-h) ) / ( 2*h ) )}

```

```

FIN

```

#### 3) (1,5 points)

##### Variante 1

```

PROCEDURE dicho(a,b,eps : REEL ; kmax : ENTIER ;

```

```

    VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)

```

```

VARIABLE sa,sb : REEL

```

```

DEBUT

```

```

    nb ← 0  sa ← a  sb ← b

```

```

    REPETER

```

```

        nb ← nb+1

```

```

        xn ← (sa+sb)/2

```

```

        SI f(sa)*f(xn)<0 ALORS sb ← xn SINON SI f(sa)*f(xn)>0 sa ← xn FINSI

```

```

        FINSI

```

```

    JUSQU'A (ABS(f(xn))<=eps) OU (nb>=kmax)

```

```

    SI (nb>=kmax)      ALORS ibf ← 1

```

```

SINON ibf ← 0
FINSI
FIN

```

#### Variante2

```

PROCEDURE dichotomie(a,b,eps : REEL ; kmax : ENTIER ;
VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)
VARIABLE sa,sb : REEL
DEBUT
  nb ← 1      sa ← a  sb ← b
  ibf ← -1
  TANTQUE (ibf = -1) FAIRE
    xn ← (sa+sb)/2
    SI f(sa)*f(xn)<0 ALORS sb ← xn SINON SI f(sa)*f(xn)>0 sa ← xn FINSI
  FINSI
  SI (ABS(f(xn))<=eps)
    ALORS      ibf ← 0
    SINON      SI nb=kmax
                  ALORS ibf ← 1
                  SINON nb ← nb+1
    FINSI
  FINSI
FINTANTQUE
FIN

```

4) (1,5 points)

#### Variante1

```

PROCEDURE newton(a,b,eps : REEL ; kmax : ENTIER ;
VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)
VARIABLE
  x0 ,erreur: REEL
DEBUT
  x0 ← (a+b)/2
  nb ← 0
  REPETER
    nb ← nb+1
    SI fprime(x0) > 0 ALORS  xn ← x0 - f(x0)/fprime(x0)
                          erreur ← ABS(xn-x0)
                          x0 ← xn
    SINON nb ← kmax+1
  FINSI
  JUSQU'A (erreur <= eps) OU (nb>=kmax)
  SI (nb>=kmax)      ALORS      ibf ← 1
                     SINON      ibf ← 0
  FINSI
FIN

```

#### Variante2

```

PROCEDURE newton(a,b,eps : REEL ; kmax : ENTIER ;
VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)

```

VARIABLE

x0 ,erreur: REEL

DEBUT

x0  $\leftarrow$  (a+b)/2

nb  $\leftarrow$  0

ibf  $\leftarrow$  -1

TANTQUE (ibf = -1) FAIRE

SI fprime(x0)=0 OU (nb=kmax) ALORS ibf  $\leftarrow$  1

SINON

nb  $\leftarrow$  nb+1

xn  $\leftarrow$  x0 -f(x0)/fprime(x0)

erreur  $\leftarrow$  ABS(xn-x0)

x0  $\leftarrow$  xn

SI erreur  $\leq$  eps ALORS ibf  $\leftarrow$  0 FINSI

FINSI

FINTANTQUE

FIN

5) (1,5 points)

a) (0,25 point)

FONCTION g(x : REEL) : REEL

DEBUT

g  $\leftarrow$  x + f(x) { ou encore Retourner(x + f(x)) }

FIN

b) (0,25 point)

FONCTION gprime(x : REEL) : REEL

CONSTANTE

h=0.001

DEBUT

gprime  $\leftarrow$  (g(x+h)-g(x-h)) / (2\*h) {ou encore Retourner( ( g(x+h)-g(x-h) ) / ( 2\*h ) )}

FIN

(1,0 point)

Variante 1

PROCEDURE success(a,b,eps : REEL ; kmax : ENTIER ;

VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)

VARIABLE

x0 ,erreur: REEL

DEBUT

x0  $\leftarrow$  (a+b)/2

nb  $\leftarrow$  0

REPETER

SI ABS((gprime(x0))<1 ALORS

nb  $\leftarrow$  nb+1

xn  $\leftarrow$  g(x0)

erreur  $\leftarrow$  ABS(xn-x0)

x0  $\leftarrow$  xn

SINON nb  $\leftarrow$  kmax

```

        FINSI
    JUSQU'A (erreur <= eps) OU (nb>=kmax)
    SI (nb>=kmax)      ALORS      ibf ← 1
                      SINON      ibf ← 0
    FINSI
FIN

```

#### Variante2

```

PROCEDURE success (a,b,eps : REEL ; kmax : ENTIER ;
                  VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)
VARIABLE
    x0 ,erreur: REEL
DEBUT
    x0 ← (a+b)/2
    nb ← 0
    ibf ← -1
    TANTQUE (ibf = -1) FAIRE
        SI ABS((gprime(x0))>=1 OU (nb=kmax) ALORS ibf ← 1
        SINON
            nb ← nb+1
            xn ← g(x0)
            erreur ← ABS(xn-x0)
            x0 ← xn
            SI erreur <=eps ALORS ibf ← 0 FINSI
        FINSI
    FINTANTQUE
FIN

```

6) (1,5 points)

#### Variante1

```

PROCEDURE falsi(a,b, eps : REEL ; kmax : ENTIER ;
               VAR xn : REEL ; VAR nb : ENTIER, VAR ibf : ENTIER)
VARIABLE  sa,sb : REEL
DEBUT
    nb ← 0      sa ← a  sb ← b
    REPETER
        nb ← nb+1
        SI sa > sb ALORS
            xn ← (sa*f(sb)-sb*f(sa))/(f(sb)-f(sa))
            SI f(sa)*f(xn)<0 ALORS sb ← xn SINON sa ← xn FINSI
        SINON nb ← kmax
    FINSI
    JUSQU'A (ABS(f(xn))<=eps) OU (nb>=kmax)
    SI (nb>=kmax)      ALORS      ibf ← 1
                      SINON      ibf ← 0
    FINSI
FIN

```

**Variante2**

```

PROCEDURE falsi(a,b,eps : REEL ; kmax : ENTIER ;
                VAR xn : REEL ; VAR nb, ibf : ENTIER)
VARIABLE sa,sb : REEL
DEBUT
    nb ← 1      sa ← a  sb ← b
    ibf ← -1
    TANTQUE (ibf = -1) FAIRE
        SI sa < sb ALORS
            xn ← (sa*f(sb)-sb*f(sa))/(f(sb)-f(sa))
            SI f(sa)*f(xn) < 0 ALORS sb ← xn SINON sa ← xn FINSI
        SINON nb ← kmax
    FINSI
    SI (ABS(f(xn)) <= eps)
        ALORS      ibf ← 0
        SINON      SI nb >= kmax
                    ALORS ibf ← 1
                    SINON nb ← nb+1
        FINSI
    FINSI
FINTANTQUE
FIN

```

 **PARTIE B (6 points)** **7) (2 points)**

```

> recherche := proc(a ::numeric, b::numeric, eps::numeric, kmax::posint)
> local i,j, tab1,tab2,tab3,tab4,TAB:
> TAB:=matrix(4,4):
> tab1:= falsi (a, b, eps, kmax) :
> tab2:= newton (a, b, eps, kmax):
> tab3:= success (a, b, eps, kmax) :
> tab4:= dicho (a, b, eps, kmax) :
> for i to 4 do
> TAB[i,1] :=i
> od:
> for j to 3 do
> TAB[1,j+1]:=tab1[j]:
> TAB[2,j+1]:=tab2[j]:
> TAB[3,j+1]:=tab3[j]:
> TAB[4,j+1]:=tab4[j]:
> od:
> RETURN(eval(TAB)):
> end:

```

 **8) (2,5 points)** **Variante1** **Version1**

```

>organisation:=proc(TAB::matrix, nl::posint)
> local i, k, li:

```

 **Version2**

```

>organisation:=proc(TAB::matrix)
> local i, k, n, li: with(linalg):

```



```

> li:=0:
> for i to n do
>   for k from i to n while TAB[k,4]=1 do od:
>   if k<=n then TAB:=swaprow(TAB,i,k): li:=li+1: fi:
> od:
> for i to li-1 do
>   for k from i+1 to li while TAB[k,2]>TAB[i,2] do od:
>   if k<=n then TAB:=swaprow(TAB,i,k) fi:
> od:
> end:

```

### **Variante 2**

```

> with(linalg):
> organisation:= proc(TAB ::matrix, nl::posint) # même chose pour l'entête de la variante2
> local i, j, n, z, p min :
> i:=1:
> n:=nl: #nombre total de lignes
> while i<n do
  #avancer jusqu'a une ligne "suivante" dont ibf=0
  while i<n and TAB[i,4]= 0 do
    i:= i+1:
  od:
  #reculer jusqu'a une ligne "suivante" dont ibf=1
  while ( TAB[n,4]=1 and i< n) do
    n:=n-1:
  od:
  #permutation
  if i<n then
    TAB:=swaprow(TAB, n , i):
    i:=i+1:
    n:=n-1:
  fi :
od:

# calcul du nombre de lignes dont ibf=0
z:=0: i:=1:
while i<=n1 and TAB[i,4]=0 do
z:=i:
i:=i+1:
od:
#tri
p:=1:
while p<z do
  #recherche max
  min:=TAB[p,2]:
  j:=p:
  for i from p+1 to z do
    if TAB[i,2] < min then min:=TAB[i,2]:
    j:=i:
  fi:

```



```

    od:
if j=p then
    p:=p+1
else
    TAB:=swaprow(TAB, p , j):
    p:=p+1:
fi:
od:
eval(TAB):
end:

```

**9) (1,5 points)**

```

TAB:=recherche(a, b, eps, kmax) ;
TAB:=organisation(TAB,4); #ou organisation(TAB);
if TAB[1,4]=1 then
    print(' aucune méthode n'a donné de résultat '):
else
    print(' la méthode faisant le minimum d'itérations est ` , TAB[1,1] , `le résultat est` ,
    TAB[1,3] ):
fi:

```