

EXERCICE

```

1. ED:=diff(q(t),t,t)+R/L*diff(q(t),t)+1/(L*C)*q(t)=0;
   EC:=r^2+R/L*r+1/L/C;
   # ou bien EC:=r**2+R/L*r+1/L/C

2. Delta:=(R/L)^2-4/L/C;
   # ou bien Delta:=discrim(EC,r);

3. solve(EC,r);
4. dsoive(ED,q(t));

5. R:=3;C:=1;L:=1;
   dsoive({ED,q(0)=1,D(q)(0)=0},q(t),numeric);

6. s:=""; # ou bien s:=%;

7. with(plots); # ou bien with(plots,odeplot);
   odeplot(s,[t,q(t)],0..20);

8. restart;
9. H:=1/(1+I/Q*omega/omega0-omega^2/omega0^2);
   omega0:=1/sqrt(L*C);
   Q:=1/R*sqrt(C/L);

10. L:=1;C:=10^(-4);R:=0.01;
11. plot(abs(H),omega=0..300);
12. plot(argument(H),omega=0..300);
    
```

PROBLEME

```

1)
fonction DEGRE( ) : entier
  variable n : entier
  début
  répéter
    écrire('saisir un entier')
    lire( n )
  jusqu'à (n>0 et n<=NMAX)
  DEGRE ← n
fin

2)
procédure SAISIVECT(n: entier ; variable B : VECT)
  variable i : entier
  début
  pour i de 1 à n faire
    écrire('donner l'élément d'indice ',i)
    lire( B[i] )
  fin pour
fin

3)
procédure SAISIMAT(n: entier ; variable X : MAT)
  variable i,j : entier
  début
  pour i de 1 à n faire
    pour j de 1 à n faire
      écrire('donner l'élément d'indice ',i,j)
      lire( X[i,j] )
    fin pour
  fin pour
fin
    
```

```

4)
procédure MINUS(n: entier ; X : MAT ; variable Y :
MAT; i : entier; j : entier)
  variable L,C : entier
  début
    
```

```

  pour L de 1 à i-1 faire
    pour C de 1 à j-1 faire
      Y[L,C] ← X[L,C]
    fin pour
  fin pour
    
```

```

  pour L de i à n-1 faire
    pour C de 1 à j-1 faire
      Y[L,C] ← X[L+1,C]
    fin pour
  fin pour
    
```

```

  pour L de 1 à i-1 faire
    pour C de j à n-1 faire
      Y[L,C] ← X[L,C+1]
    fin pour
  fin pour
    
```

```

  pour L de i à n-1 faire
    pour C de j à n-1 faire
      Y[L,C] ← X[L+1,C+1]
    fin pour
  fin pour
fin
    
```

```

5)
procédure REMPLACE(n: entier ; X : MAT ; B : VECT ;
variable Y : MAT; j : entier)
  variable L,C : entier
    
```

```

  début
  pour L de 1 à n faire
    pour C de 1 à j-1 faire
      Y[L,C] ← X[L,C]
    fin pour
  fin pour
    
```

```

  pour L de 1 à n faire
    pour C de j+1 à n faire
      Y[L,C] ← X[L,C]
    fin pour
  fin pour
    
```

```

  pour L de 1 à n faire
    Y[L,j] ← B[L]
  fin pour
fin
    
```



6)
procédure SELECTLIGNE(n: entier ; X : MAT ;
variable i : entier; variable nb : entier)
variable L, C, S : entier

début
i ← 1
S ← 0
pour C de 1 à n faire
si X[i,C] = 0 alors
S ← S+1
fin
fin pour
nb ← S

pour L de 2 à n faire
S ← 0
pour C de 1 à n faire
si X[L,C] = 0 alors
S ← S+1
fin
fin pour
si nb < S alors
nb ← S
i ← L
fin
fin pour
fin

7)
procédure SELECTCOLONNE(n: entier ; X : MAT ;
variable j : entier; variable nb : entier)
variable L, C, S : entier

début
j ← 1
S ← 0
pour L de 1 à n faire
si X[L,j] = 0 alors
S ← S+1
fin
fin pour
nb ← S

pour C de 2 à n faire
S ← 0
pour L de 1 à n faire
si X[L,C] = 0 alors
S ← S+1
fin
fin pour
si nb < S alors
nb ← S
j ← C
fin
fin pour
fin

8)
procédure METHODE1(n: entier ; A : MAT ; B : VECT
; variable X : VECT)
variable
C : entier
Z : MAT

début
pour C de 1 à n faire
REPLACE (n; A ; B ; Z ; C)
X[C] ← DETERMINANT(r, Z) / DETERMINANT(n, A)
fin pour
fin

9)
procédure CALCULAD(n: entier ; X : MAT ; variable Y
: MAT)
variable
L, C : entier
Z : MAT

début
pour L de 1 à n faire
pour C de 1 à n faire
MINUS(n ; X ; Z ; L; C)
si (L+C) MOD 2 = 0 alors
Y[L,C] ← X[L,C] * DETERMINANT(n-1, Z)
sinon
Y[L,C] ← - X[L,C] * DETERMINANT(n-1, Z)
fin
fin pour
fin pour
fin

10)
procédure TRANSPOSE(n: entier ; X : MAT ; variable
Y : MAT)
variable
L, C : entier

début
pour L de 1 à n faire
pour C de 1 à n faire
Y[L, C] ← X[C, L]
fin pour
fin pour
fin

11)
procédure INVERSE(n: entier ; X : MAT ; variable Y :
MAT)
variable
L, C : entier
Z, T : MAT

début
CALCULAD(n; X ; Z)
TRANSPOSE(n; Z ; T)
pour L de 1 à n faire
pour C de 1 à n faire
Y[L, C] ← T[L, C] / DETERMINANT(n, X)
fin pour
fin pour
fin

12)

procédure METHODE2 (n: entier ; A : MAT ; B : VECT
; variable X : VECT)
variable

L, C : entier

IA : MAT

début

INVERSE(n; A; IA)

pour L de 1 à n faire

X[L] ← 0

pour C de 1 à n faire

X[L] ← X[L] + IA[L, C] * B[C]

fin pour

fin pour

fin

13)

procédure TRIANGLE(n: entier; variable A : MAT)
variable

i, j, k : entier

x : réel

début

pour k de 2 à n faire

pour i de k à n faire

x ← A[i, k-1] / A[k-1, k-1]

pour j de k-1 à n faire

A[i, j] ← A[i, j] - x * A[k-1, j]

fin pour

fin pour

fin pour

fin

14)

procédure METHODE3 (n: entier ; A : MAT ; B : VECT
; variable X : VECT)
variable

i, j : entier

S : réel

début

TRIANGLE(n; A)

X[n] ← B[n] / A[n, n]

pour i de n-1 à 1 pas = -1 faire

S ← 0

pour j de i+1 à n faire

S ← S - A[i, j] * X[j]

fin pour

X[i] ← (B[i] - S) / A[i, i]

fin pour

fin

15)

a)

S := { 1/2*x1 + 1/3*x2 + 1/4*x3 = 1,

1/3*x1 + 1/4*x2 + 1/5*x3 = 2, 1/4*x1 + 1/5*x2 + 1/6*x3 = 3 };

solve(S, {x1, x2, x3});

b)

with(linalg):

A := matrix(3,3,(i,j)->1/(i+j));

c)

b := vector([1,2,3]);

d)

METHODE4 := proc(A, b, x):

x := linsolve(A, b);

end;

e)

COMPARAISON := proc(A, b, x, n):

local TM1, TM2, TM3, TM4, min, A1;

A1 := eval(A);

TM1 := time();

METHODE1(n, A, B, x):

TM1 := time() - TM1;

A := eval(A1);

TM2 := time();

METHODE2(n, A, B, x):

TM2 := time() - TM2;

A := eval(A1);

TM3 := time();

METHODE3(n, A, B, x):

TM3 := time() - TM3;

A := eval(A1);

TM4 := time();

METHODE4(A, B, x):

TM4 := time() - TM4;

min := TM1;

if(TM2 < min) then min := TM2: fi;

if(TM3 < min) then min := TM3: fi;

if(TM4 < min) then min := TM4: fi;

if (TM1 = min) then

print(' La méthode 1 est la meilleure');

elif (TM2 = min) then

print(' La méthode 2 est la meilleure');

elif (TM3 = min) then

print(' La méthode 3 est la meilleure');

else print(' La méthode 4 est la meilleure');

fi;

end;

