

Projet de programmation C++

Résolution de circuit

JÉRÉMIE FOURMANN (Promo 2013 - Électronique - Enseeiht)

MAXIME MORIN (Promo 2013 - Électronique - Enseeiht)

20 novembre 2011

Plan

1	Objectif	2
2	Organisation du code	2
2.1	L'objet circuit	2
2.2	L'objet source	2
3	Code Source	3
3.1	main.cpp	3
3.2	circuits.h	4
3.3	circuits.cpp	5
3.4	sources.h	8
3.5	sources.cpp	9
4	Résultats	11
4.1	Réponse du l'exemple 1	11
4.2	Réponse du CircuitA	11
4.3	Réponse du CircuitB	11
4.4	Réponse du CircuitC	12
4.5	Réponse du CircuitD	12

1 Objectif

2 Organisation du code

2.1 L'objet circuit

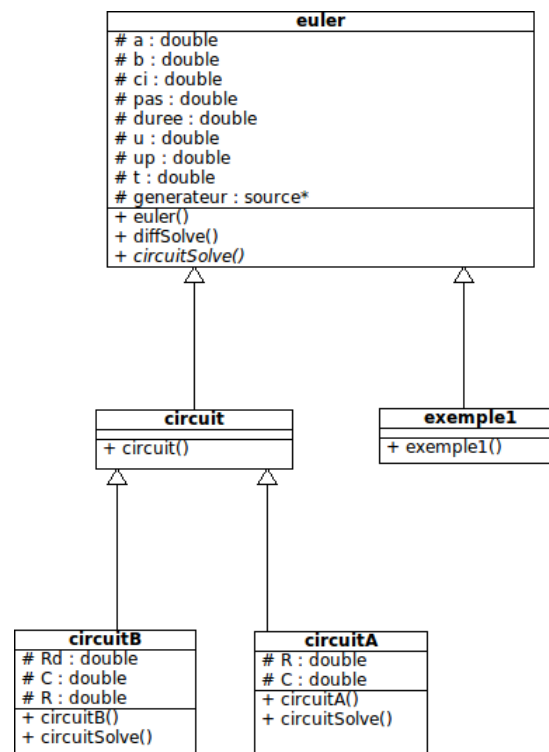


FIGURE 1 – Hiérarchie de la classe circuit

2.2 L'objet source

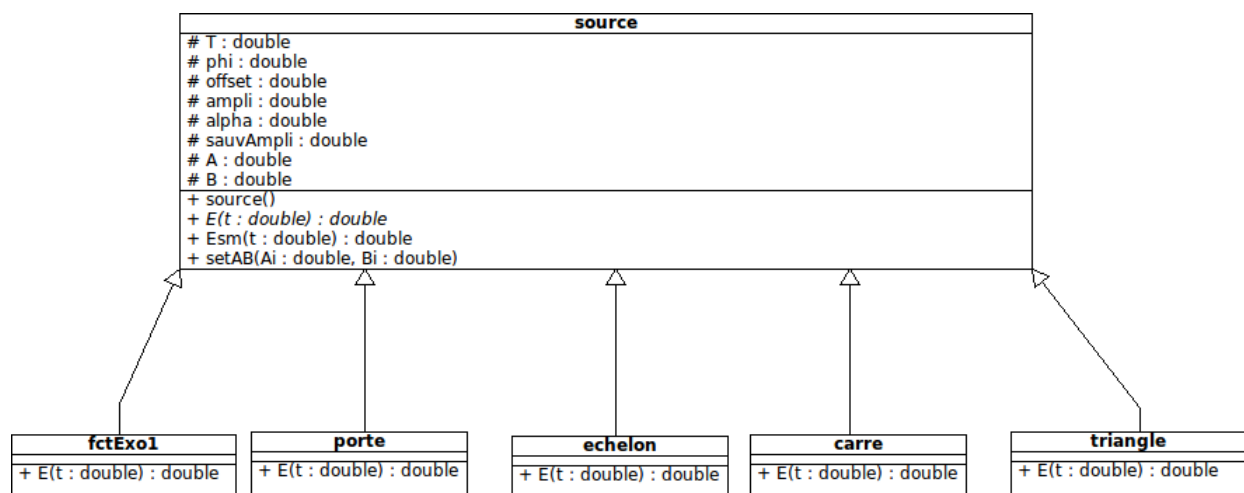


FIGURE 2 – Hiérarchie de la classe source

3 Code Source

3.1 main.cpp

```
1  /* Programmation orientee objet : BE2 */
2  /* Jeremie Fourmann et Maxime Morin */
3  /* main.cpp */
4  /* Programme principal */
5
6
7  #include <iostream>
8  #include "circuits.h"
9  #include "sources.h"
10
11 using namespace std;
12
13 int main(int argc, char **argv)
14 {
15     cout.width(6);
16     cout.precision(4);
17
18     circuit * montage;
19     int choix=0;
20
21     cout << "#Quel montage ?" << endl;
22     cout << "#1 - Circuit A" << endl;
23     cout << "#2 - Circuit B" << endl;
24     cin >> choix;
25
26     switch(choix){
27     case 1:
28         montage = new circuitA;
29         break;
30     case 2:
31         montage = new circuitB;
32         break;
33     default:
34         cout << "#Mauvais choix" << endl;
35         return 0;
36     }
37
38
39     montage->circuitSolve();
40
41     return 0;
42 }
```

3.2 circuits.h

```
1  /* Programmation orientee objet : BE2 */
   /* Jeremie Fourmann et Maxime Morin */
3  /* circuits .h */
   /* Declaration des classes circuits */
5
   #ifndef DEF_circuits
7  #define DEF_circuits
   #include "sources.h"
9
   /* Classe "euler" pour la resolution de au'+bu=f. */
11  class euler{
       protected:
13         double a,b,ci,pas,duree,u,up,t ;
           source *generateur;
15     public:
           euler();
17         void diffSolve();
           virtual void circuitSolve() =0;
19 };

21 /* Classe "exemple1". */
   class exemple1 : public euler{
23     public:
           exemple1();
25 };

27 /* Classe "circuit" (permet le choix de la source) */
   class circuit : public euler{
29     public:
           circuit();
31 };

33 /* Classe "circuitA". */
   class circuitA : public circuit{
35     protected:
           double R,C;
37     public:
           circuitA();
39         void circuitSolve();
       };
41
   /* Classe "circuitB". */
43  class circuitB : public circuit{
       protected:
45         double Rd,C,R;
       public:
47         circuitB();
           void circuitSolve();
49 };

51 #endif
```

3.3 circuits.cpp

```
1  /* Programmation orientee objet : BE2 */
   /* Jeremie Fourmann et Maxime Morin  */
3  /* circuits .cpp                      */
   /* Definition des classes circuits    */
5
   #include <iostream>
7  #include <math.h>
   #include "circuits.h"
9
   using namespace std;
11
   euler :: euler () {
13       a=0.0;
       b=0.0;
15       ci=0.0;
       pas=0.1;
17       duree=10.0;
       u=0.0;
19       up=0.0;
       t=0.0;
21   }

23   void euler :: diffSolve () {
       up=u;
25       u=(pas/a)*(generateur->Esm(t)+up*(-b+a/pas));
       t=t+pas;
27   }

29   exemple1::exemple1() { //Cas special de l'exercice 1
       a=1;
31       b=3;
       generateur = new fctExo1;
33   }

35   /* Choix de la source lors de la creation d'un circuit . */
   circuit :: circuit () {
37       int choix=0;

39       cout << "#Choisir la source ?" << endl;
       cout << "#1 - Echelon" << endl;
41       cout << "#2 - Porte" << endl;
       cout << "#3 - Carre" << endl;
43       cout << "#4 - Triangle" << endl;
       cin >> choix;

45
       switch(choix){
47       case 1:
           generateur=new echelon;
49           break;
       case 2:
           generateur=new porte;
51           break;
```

```

53     case 3:
        generateur=new carre;
55     break;
    case 4:
        generateur=new triangle;
        break;
59     default:
        break;
61 }
}

63
/* Circuit A avec comme paramtres R et C */
64 circuitA :: circuitA(){
    cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
65     cout << "#_/_/_/_/_/_/_/_/_/_/_/_/_/_/__" << endl ;
    cout << "#|      R      _|_" << endl ;
67     cout << "#E      C ----" << endl ;
    cout << "#|_/_/_/_/_/_/_/_/_/__|" << endl ;
71
    cout << "#Valeur de R (Ohm) : " << endl;
73     cin >> R ;
    cout << "#Valeur de C (Farad) : " << endl ;
75     cin >> C ;

77     a=R*C;
    b=1;
79     generateur->setAB(1,0); // Esm(t) = E(t)
}

81
/* Resolution de l'equation differentielle du circuitA pour la source choisie . */
82 void circuitA :: circuitSolve (){

83
84     cout << "#Temps" << " " << "Ve" << " " << "Vs" << " " << endl;
    while(t<= duree){
85         diffSolve ();
86         cout << t << " " << generateur->E(t) << " " << u << endl;
87     }
88 }

91

92
/* Circuit B avec comme paramtres Rd, R et C. */
93 circuitB :: circuitB(){
94     cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
    cout << "#_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/__|" << endl ;
95     cout << "#|      Rd      |/" << endl ;
    cout << "#|      D      /      _|_" << endl ;
97     cout << "#E      R \\\      ---- C " << endl ;
    cout << "#|      /      |" << endl ;
99     cout << "#|_/_/_/_/_/_/_/_/_/__|" << endl ;
101
102     cout << "#Valeur de Rd (Ohm) : " << endl;
    cin >> Rd ;
103     cout << "#Valeur de R (Ohm) : " << endl;
105

```

```

    cin >> R ;
107    cout << "#Valeur de C (Farad) : " << endl ;
    cin >> C ;
109 }

111 /*Resolution des equations differentielles circuitB pour la source
choisie , pour les deux differents etats de la diode */
113 void circuitB :: circuitSolve () {
    bool bloquee=1; //Flag d'etat de la diode
115    double vd=.7; // A t=0, C dechargee donc D passante (vd>0.6)
    ci=0; // C dechargee

117    cout << "#Temps" << " " << "Ve" << " " << "Vs" << " " << "Vd" << endl;
119    while(t<=duree){
        if(vd>=.6 && bloquee ){
121            a=Rd*C;
            b=1+Rd/R;
123            generateur->setAB(1,-0.6); // Offset pour le second membre
            ci=u;
125            cout << "#Diode passante" << endl;
            bloquee=0;

127        }
        if(vd<.6 && !bloquee )
129        {
            a=R*C;
131            b=1;
            generateur->setAB(0,0); // Second membre nul, decharge de C dans R
133            ci=u;
            cout << "#Diode bloquee" << endl;
135            bloquee=1;
        }
137        diffSolve () ;
        vd=generateur->E(t)-u-Rd*C*(u-up)/pas+u/R;
139        cout << t << " " << generateur->E(t) << " " << u << " " << vd << endl;
    }
141 }

```

3.4 sources.h

```
1  /* Programmation orientée objet : BE2 */
2  /* Jeremie Fourmann et Maxime Morin */
3  /* sources.h */
4  /* Declaration des classes sources */
5
6  #ifndef DEF_sources
7  #define DEF_sources
8
9  /* Classe mre : source. */
10 class source{
11
12     protected:
13         double T,phi,offset,ampli,alpha,sauvAmpli;
14         double A,B;
15     public:
16         source();
17         virtual double E(double t)=0;//fct virtuelle de la source
18         double Esm(double t); // Transformation affine de E pour changer amplitude
19                                 // ou ajouter un offset dans le second membre
20         void setAB(double Ai, double Bi); //accesseur pour les valeurs A et B
21 };
22
23 /* Classe fille permettant de traiter l'exemple 1. */
24 class fctExo1 : public source{
25     public:
26         double E(double t);
27 };
28
29 /* Classes filles pour les différents signaux d'entrée. */
30 class echelon : public source{
31     public:
32         double E(double t);
33 };
34
35 class porte : public source{
36     public:
37         double E(double t);
38 };
39
40 class triangle : public source{
41     public:
42         double E(double t);
43 };
44 class carre : public source{
45     public:
46         double E(double t);
47 };
48
49 #endif
```


3.5 sources.cpp

```
1  /* Programmation orientee objet : BE2 */
   /* Jeremie Fourmann et Maxime Morin */
3  /* sources.cpp */
   /* Definition des classes sources */
5
   #include <iostream>
7  #include "sources.h"
   #include <math.h>
9
   using namespace std;
11
13  /* Methodes de la classe mre "source". */
   source::source(){
15     T=2;
     phi=1;
17     offset=0;
     ampli=5;
19     alpha=.6;
     A=1, B=0;
21 }

23 double source::Esm(double t) // Transformation affine du signal de la source
   {
25     return A*E(t)+B ;
   }
27
   /* Definitions des sources filles pour differents types de signaux ou fonctions. */
29
   double fctExo1::E(double t){
31
33     return -3*t;
   }

35 void source::setAB(double Ai, double Bi)
   {
37     A = Ai;
     B = Bi;
39 }

41 double echelon::E(double t){
     double fx;
43     if(phi <=t ) fx= offset+ampli;
     else fx= offset ;
45     return fx;
   }
47
   double porte::E(double t){
49     double fx;
     if(phi < t && t < phi+T) fx=offset+ampli;
51     else fx=offset ;
     return fx;
```

```

53 }

55 double carre::E(double t){
    double fx;
57     if((t-phi)-floor((t-phi)/T)*T<T*alpha) fx=offset+ampli;
        else fx=offset;
59     return fx;
    }

61 double triangle::E(double t){
63     double fx;
    if((t-phi)-floor((t-phi)/T)*T<=T/2) fx=((t-phi)-floor((t-phi)/T)*T-.5)*ampli+offset;
65     else fx=(-((t-phi)-floor((t-phi)/T)*T)+2-.5)*ampli+offset;
    return fx;
67 }

```

4 Résultats

4.1 Réponse du l'exemple 1

4.2 Réponse du CircuitA

4.3 Réponse du CircuitB

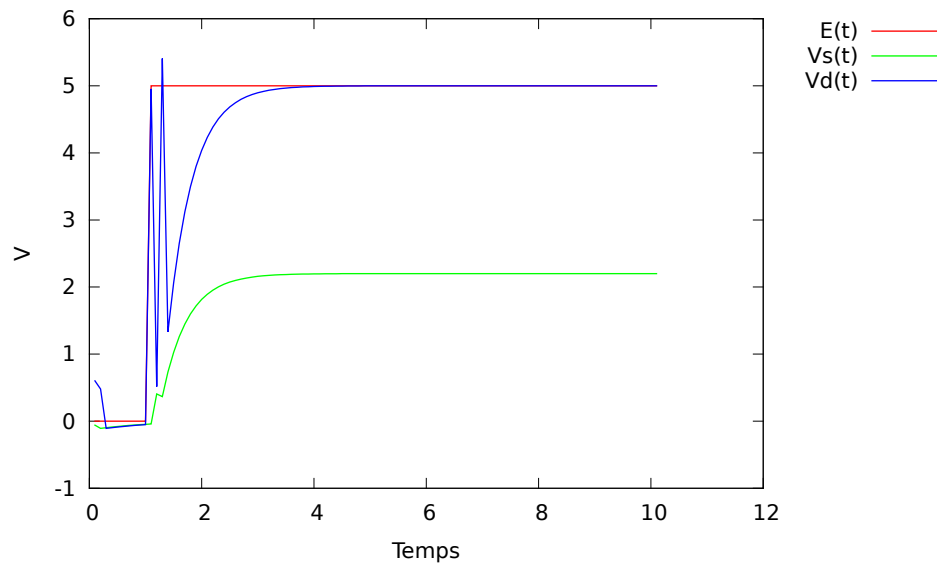


FIGURE 3 – Réponse à un echelon de tension

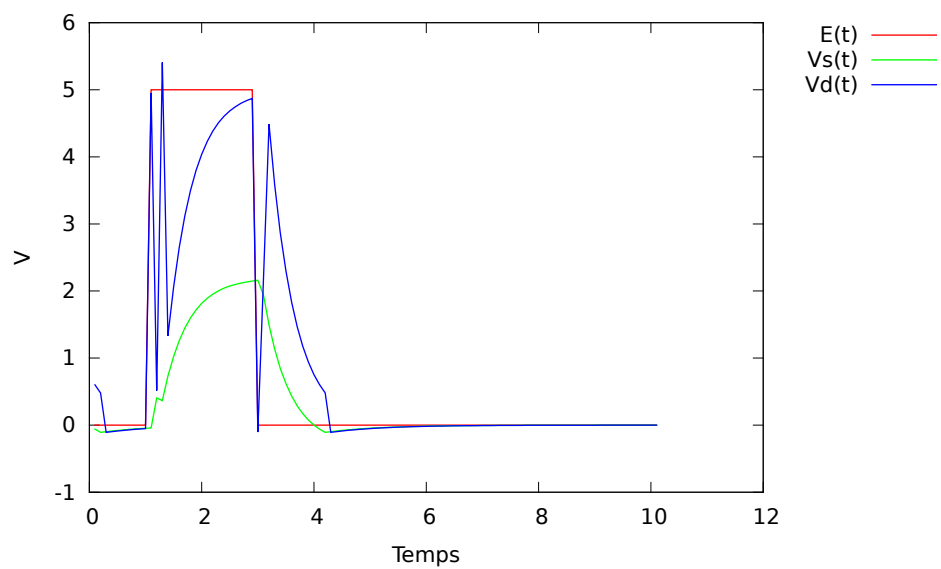


FIGURE 4 – Réponse à une porte

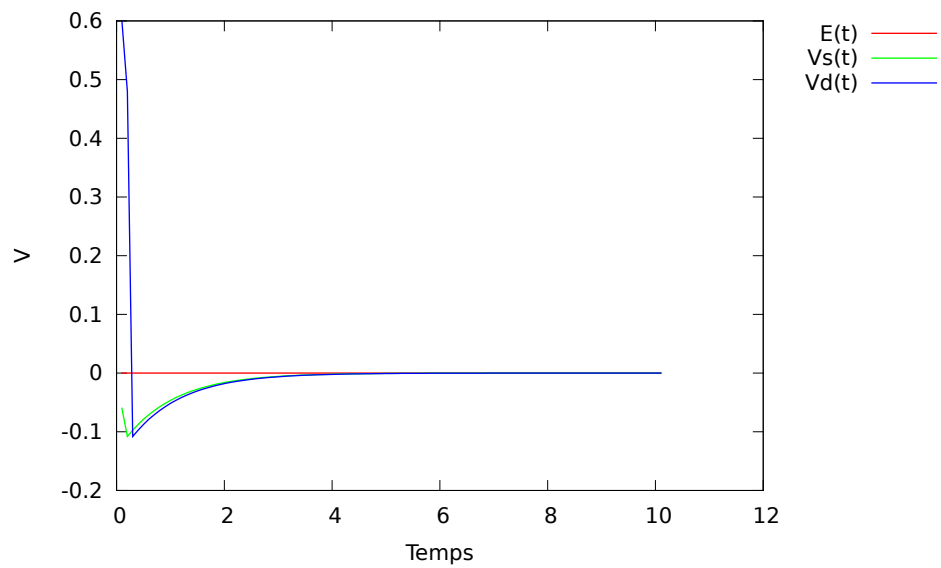


FIGURE 5 – Réponse à signal carré

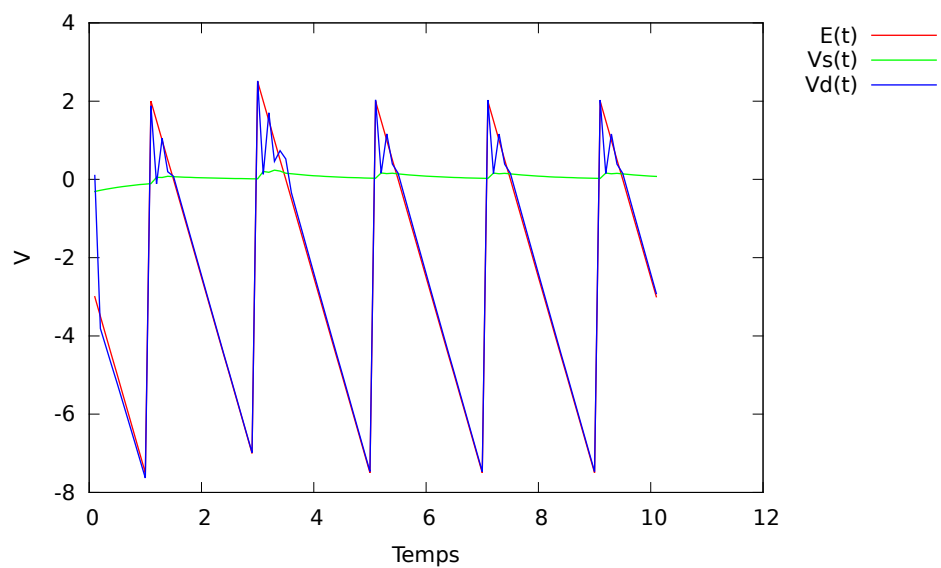


FIGURE 6 – Réponse à signal triangle

4.4 Réponse du CircuitC

4.5 Réponse du CircuitD