

Projet de programmation C++

Résolution de circuit

JÉRÉMIE FOURMANN (Promo 2013 - Électronique - Enseeiht)

MAXIME MORIN (Promo 2013 - Électronique - Enseeiht)

5 décembre 2011

Plan

1	Objectif	2
2	Organisation du code	2
2.1	L'objet circuit	2
2.2	L'objet source	2
3	Résultats	3
3.1	Exemple 1	3
3.2	Exemple 2	3
3.3	Réponse du CircuitA	5
3.4	Réponse du CircuitB	6
3.5	Réponse du CircuitC	7
3.6	Réponse du CircuitD	8
A	main.cpp	9
B	circuits.h	10
C	circuits.cpp	12
D	sources.h	16
E	sources.cpp	17

1 Objectif

2 Organistion 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 Résultats

3.1 Exemple 1

Résolution de l'équation différentielle du 1^{er}ordre :

$$\begin{cases} u'(t) = -3 \cdot u(t) - 3 \cdot t \\ u(0) = 0 \end{cases}$$

La solution exacte étant $u(t) = -1/3 \cdot \exp(-3t) - 1/3$

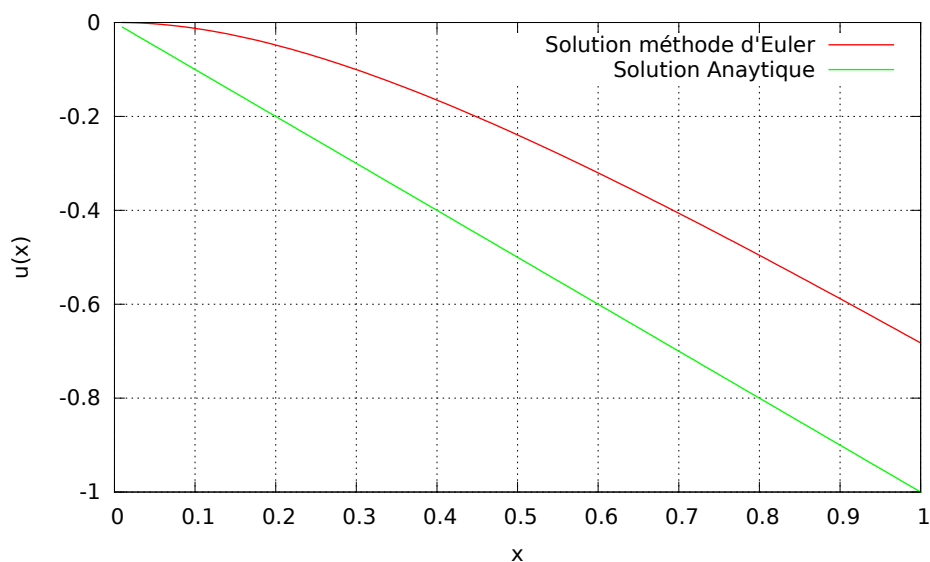


FIGURE 3 – Solution de l'exemple 1

3.2 Exemple 2

Résolution de l'équation différentielle du 1^{er}ordre :

$$\begin{cases} u''(t) = -\lambda \cdot u(t) \\ u(0) = 0 \\ u'(0) = 1 \end{cases}$$

La solution exacte étant $u(t) = \sin(t)$

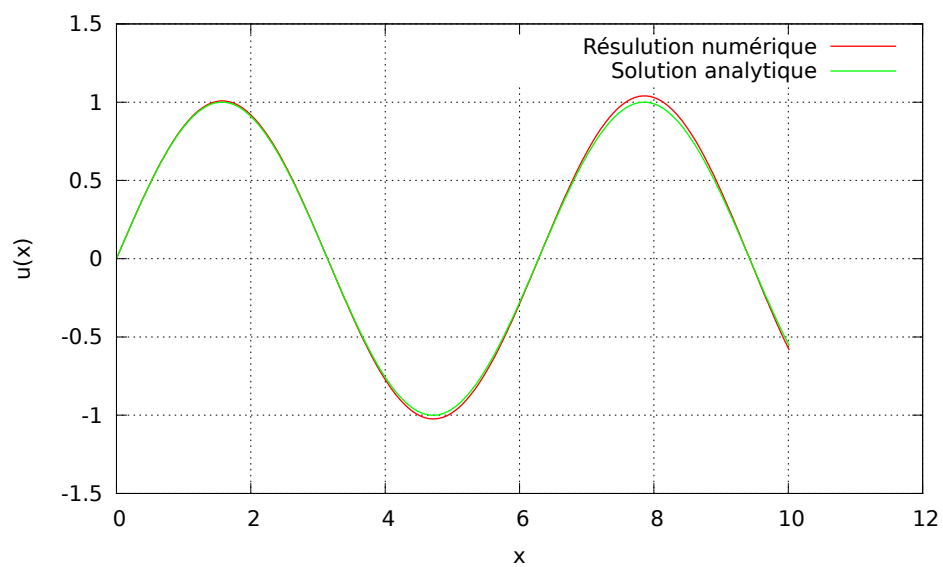


FIGURE 4 – Solution de l'exemple 2

3.3 Réponse du CircuitA

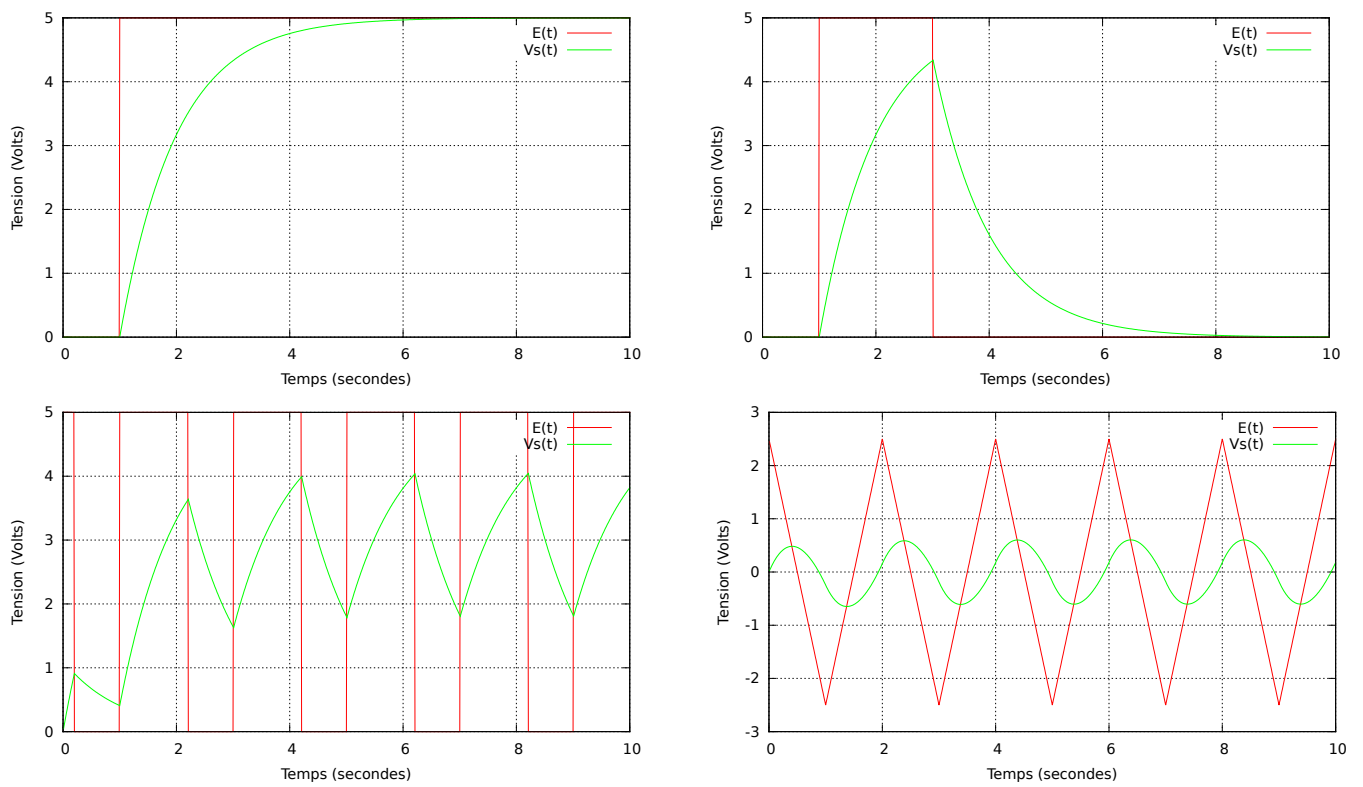


FIGURE 5 – Réponse du circuit A

3.4 Réponse du CircuitB

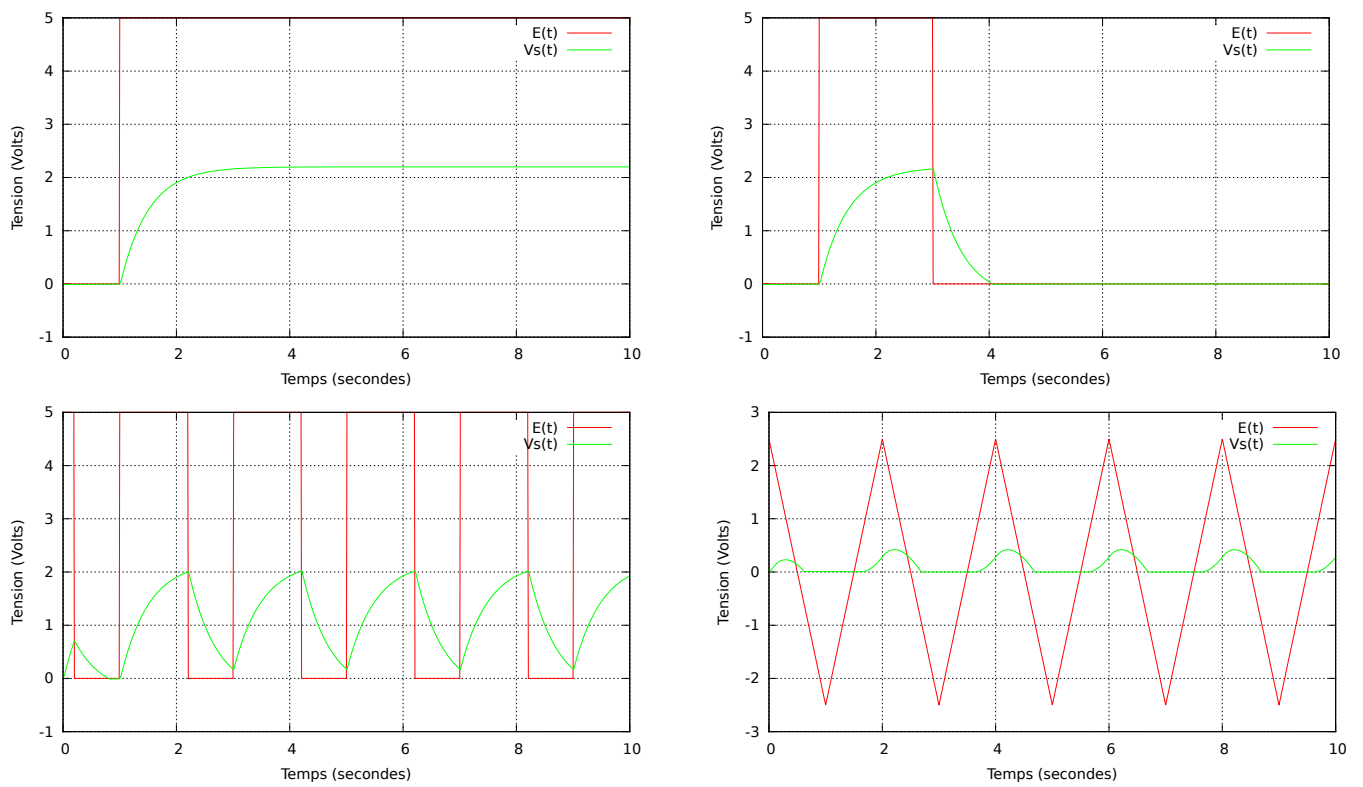


FIGURE 6 – Réponse du circuit B

3.5 Réponse du CircuitC

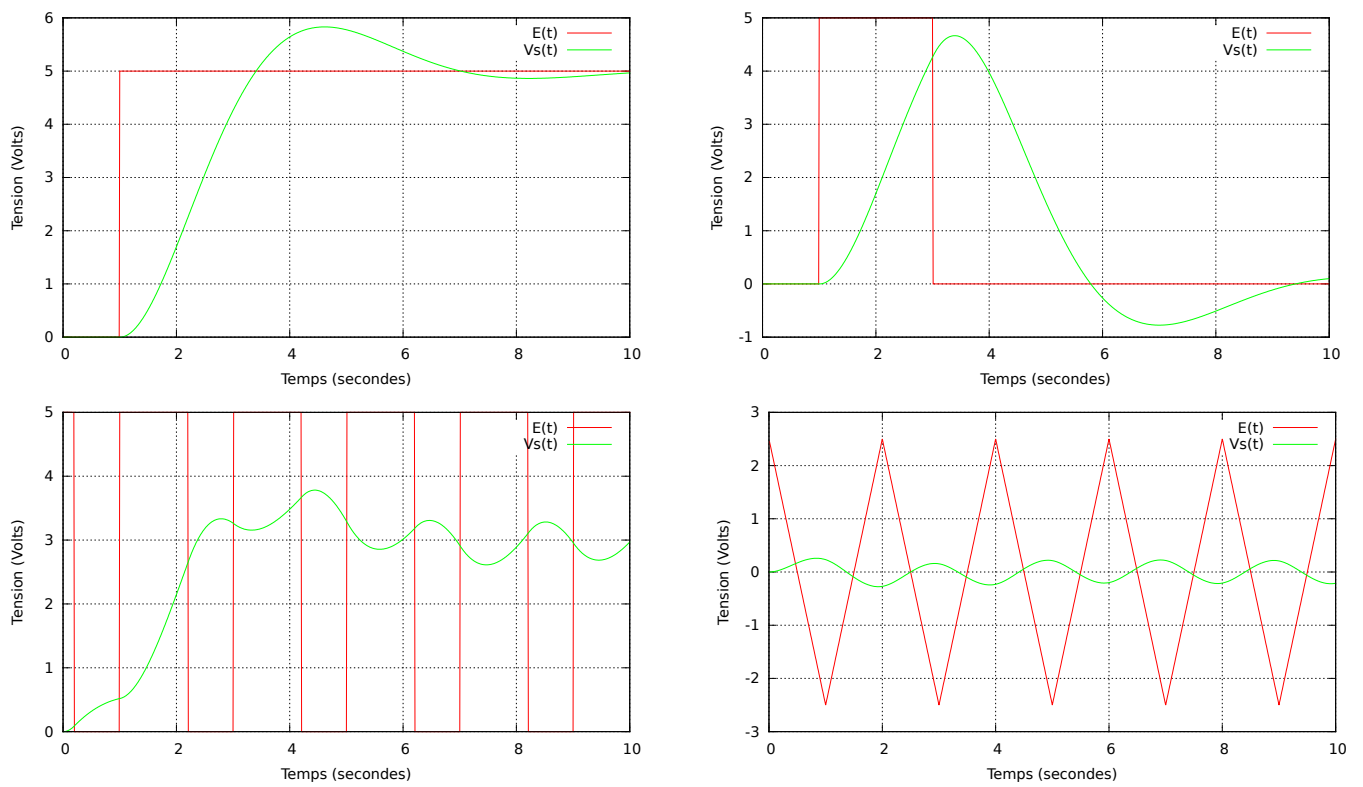


FIGURE 7 – Réponse du circuit C

3.6 Réponse du CircuitD

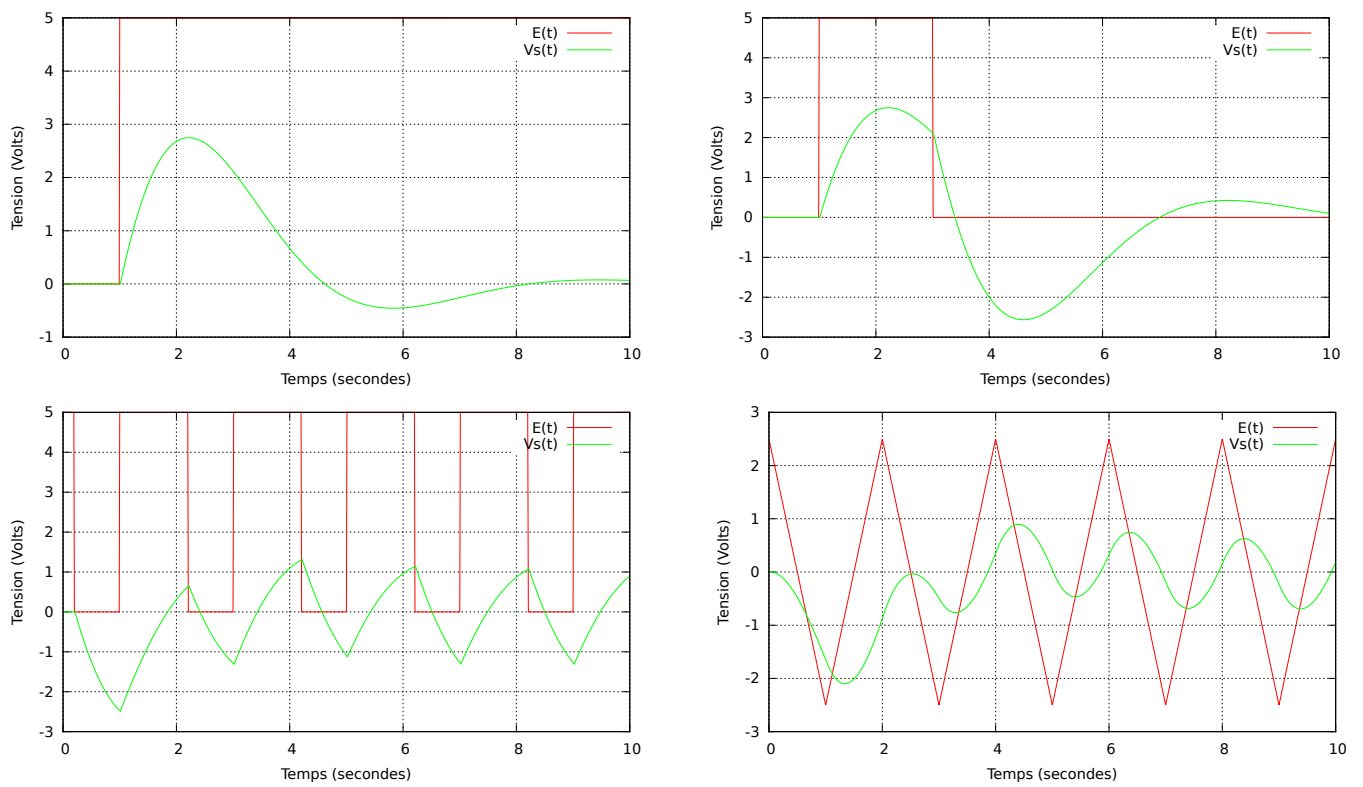


FIGURE 8 – Réponse du circuit D

A 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 << "#Premier Ordre : " << endl;
22     cout << "#1 - Exemple 1" << endl;
23     cout << "#2 - Circuit A" << endl;
24     cout << "#3 - Circuit B" << endl;
25     cout << "#Deuxime Ordre : " << endl;
26     cout << "#4 - Exemple 2" << endl;
27     cout << "#5 - Circuit C" << endl;
28     cout << "#6 - Circuit D" << endl;
29     cin >> choix;
30
31     switch(choix){
32     case 1:
33         montage = new exemple1;
34         break;
35     case 2:
36         montage = new circuitA;
37         break;
38     case 3:
39         montage = new circuitB;
40         break;
41     case 4:
42         montage = new exemple2;
43         break;
44     case 5:
45         montage = new circuitC;
46         break;
47     case 6:
48         montage = new circuitD;
49         break;
50     default:
51         cout << "#Mauvaix choix" << endl;
52         return 0;
53     }
54
55
56     montage->circuitSolve();
57
58     return 0;
59 }
```

B circuits.h

```
1  /* Programmation orientee objet : BE2 */
2  /* Jeremie Fourmann et Maxime Morin */
3  /* circuits.h */
4  /* Declaration des classes circuits */
5
6  #ifndef DEF_circuits
7  #define DEF_circuits
8  #include "sources.h"
9
10 /* Classe "euler" pour la resolution de au'+bu=f. */
11 class euler{
12     protected:
13         double pas,duree,t ;
14         source *generateur;
15     public:
16         euler();
17         virtual void diffSolve()=0;
18         virtual void circuitSolve()=0;
19 };
20
21 /* Classe "circuit" (permet le choix de la source) */
22 class circuit : public euler{
23     protected:
24         double a,b,ci,u,up;
25     public:
26         circuit();
27         virtual void diffSolve()=0;
28         virtual void circuitSolve()=0;
29 };
30
31 /* Classe "circuit1" (1er ordre) */
32 class circuit1 : public circuit{
33     public:
34         void diffSolve();
35         virtual void circuitSolve() =0; //defini en fct du circuit
36 };
37
38 /* Classe "exemple1". */
39 class exemple1 : public circuit1{
40     public:
41         exemple1();
42         void circuitSolve();
43 };
44
45
46 /* Classe "circuitA". */
47 class circuitA : public circuit1{
48     protected:
49         double R,C;
50     public:
51         circuitA();
52         void circuitSolve();
53 };
54
55 /* Classe "circuitB". */
56 class circuitB : public circuit1{
57     protected:
58         double Rd,C,R;
59     public:
60         circuitB();
61         void circuitSolve();
62 };
63 }
```

```

65  /* Classe "circuit2" (2eme Ordre)*/
    class circuit2 : public circuit{
67      protected:
          double ci2,u2,u2p;
69      public:
          circuit2();
71          virtual void diffSolve()=0;
          virtual void circuitSolve();
73  };
    /* Classe "exemple2" (2eme Ordre)*/
75  class exemple2 : public circuit2{
        public:
77          exemple2();
          void diffSolve();
79          void circuitSolve(); //Redefinition pour les besoins de l'exemple
    };
81
    /* Classe "circuitC". */
83  class circuitC : public exemple2{
        protected:
85          double R,C,L;
        public:
87          circuitC();
    };
89
    /* Classe "circuitD". */
91  class circuitD : public circuit2{
        protected:
93          double R,C,L;
        public:
95          circuitD();
          void diffSolve();
97  };
99  #endif

```

C circuits.cpp

```
1  /* Programmation orientee objet : BE2 */
2  /* Jeremie Fourmann et Maxime Morin */
3  /* circuits.cpp */
4  /* Definition des classes circuits */
5
6  #include <iostream>
7  #include <math.h>
8  #include "circuits.h"
9
10 using namespace std;
11
12 euler::euler(){
13     pas=0.01;
14     duree=10;
15     t=0.0;
16 }
17
18 /* Choix de la source lors de la creation d'un circuit. */
19 circuit::circuit(){
20     int choix=0;
21     a=0.0;
22     b=0.0;
23     ci=0.0;
24     u=0.0;
25     up=0.0;
26
27     cout << "#Choisir la source ?" << endl;
28     cout << "#1 - Echelon" << endl;
29     cout << "#2 - Porte" << endl;
30     cout << "#3 - Carre" << endl;
31     cout << "#4 - Triangle" << endl;
32     cout << "#5 - Rampe f(t)=-3*t (Exemple1)" << endl;
33     cout << "#6 - Nulle (Exemple 2)" << endl;
34     cin >> choix;
35
36     switch(choix){
37     case 1:
38         generateur=new echelon;
39         break;
40     case 2:
41         generateur=new porte;
42         break;
43     case 3:
44         generateur=new carre;
45         break;
46     case 4:
47         generateur=new triangle;
48         break;
49     case 5:
50         generateur=new fctExo1;
51         break;
52     case 6:
53         generateur=new echelon; /* Generateur quelconque. */
54         generateur->setAB(0,0); /* Coupe le generateur. */
55         break;
56     default:
57         break;
58     }
59 }
60
61 void circuit1::diffSolve(){
62     up=u;
63     u=(pas/a)*(generateur->Esm(t)+up*(-b+a/pas));
```

```

65         t=t+pas;
66     }
67
68     exemple1::exemple1(){ //Cas "mathematique" de l'exercice 1
69         a=1;
70         b=3;
71         ci = 0;
72     }
73
74     void exemple1::circuitSolve(){
75         cout << "#Temps" << " " << "SolEuler" << " " << "SolExacte" << " " << endl;
76         while(t<= duree){
77             diffSolve();
78             cout << t << " " << u << " " << -(1/3)*exp(-3*t) -t + (1/3) << endl;
79         }
80     }
81
82     /* Circuit A avec comme parametres R et C */
83     circuitA::circuitA(){
84         cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
85         cout << "#_/_/_/_/_/_/_/_/_/_/_/__ " << endl ;
86         cout << "#|          R          |_" << endl ;
87         cout << "#E          C ---" << endl ;
88         cout << "#|_/_/_/_/_/_/__|" << endl ;
89
90         cout << "#Valeur de R (Ohm) : " << endl;
91         cin >> R ;
92         cout << "#Valeur de C (Farad) : " << endl ;
93         cin >> C ;
94
95         a=R*C;
96         b=1;
97         generateur->setAB(1,0); // Esm(t) = E(t)
98     }
99
100     /* Resolution de l'equation differentielle du circuitA pour la source choisie. */
101     void circuitA::circuitSolve(){
102
103         cout << "#Temps" << " " << "Ve" << " " << "Vs" << " " << endl;
104         while(t<= duree){
105             diffSolve();
106             cout << t << " " << generateur->E(t) << " " << u << endl;
107         }
108     }
109 }
110
111 /* Circuit B avec comme paramtres Rd, R et C. */
112 circuitB::circuitB(){
113     cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
114     cout << "#_/_/_/_/_/_/_/_/_/_/_/__|_/_/_/_/_/_/__ " << endl ;
115     cout << "#|          Rd          |/" << endl ;
116     cout << "#|          D          /      |_" << endl ;
117     cout << "#E          R \\\      --- C " << endl ;
118     cout << "#|          /          |" << endl ;
119     cout << "#|_/_/_/_/_/_/_/_/_/_/__|_/_/__|" << endl ;
120
121     cout << "#Valeur de Rd (Ohm) : " << endl;
122     cin >> Rd ;
123     cout << "#Valeur de R (Ohm) : " << endl;
124     cin >> R ;
125     cout << "#Valeur de C (Farad) : " << endl ;
126     cin >> C ;
127 }
128
129 /*Resolution des equations differentielles circuitB pour la source

```

```

131 choisie, pour les deux differents etats de la diode */
void circuitB::circuitSolve(){
133     bool bloquee=1; //Flag d'etat de la diode
    double vd=.7; // A t=0, C dechargee donc D passante (vd>0.6)
135     ci=0; // C dechargee

137     cout << "#Temps" << " " << "Ve" << " " << "Vs" << " " << "Vd" << endl;
    while(t<=duree){
139         if(vd>=.6 && bloquee ){
            a=Rd*C;
141            b=1+Rd/R;
            generateur->setAB(1,-0.6); // Offset pour le second membre
143            ci=u;
            cout << "#Diode passante"<<endl;
145            bloquee=0;
        }
147         if(vd<.6 && !bloquee )
        {
149             a=R*C;
            b=1;
151             generateur->setAB(0,0); // Second membre nul, decharge de C dans R
            ci=u;
153             cout << "#Diode bloquee"<<endl;
            bloquee=1;
155         }
        diffSolve();
157         vd=generateur->E(t)-u-Rd*C*(u-up)/pas+u/R;
        cout << t << " " << generateur->E(t) << " " << u << " " << vd << endl;
159     }
}

161 /*Circuit 2 ordre*/
163 circuit2::circuit2(){
165     u2=0.0;
    u2p=0.0;
167     ci2=0;

169 }

171 void circuit2::circuitSolve(){
    cout << "#Temps" << " " << "ESM" << " " << "Vs" << " " << endl;
173     u=ci;
    u2=ci2;
175     while(t<=duree){
        diffSolve();
177         cout << t << " " << generateur->Esm(t) << " " << u << endl;
    }
179 }

181 /*Resolution de l'exemple numero 2 */
exemple2::exemple2(){ //Cas mathematique de l'exercice 2
183     a=0.0;
    b=-1.0;
185     ci2=1;
}

187 void exemple2::diffSolve(){
189     up=u;
    u2p=u2;
191     u=up+pas*u2p;
    u2=u2p+pas*(b*up+a*u2p+generateur->Esm(t));
193     t=t+pas;
}

195 void exemple2::circuitSolve(){

```

```

197     cout << "#Temps" << " " << "Entree" << " " << "Sortie-Solution" << " " << "Sinus(sol exact)" << "
    " << endl;
    u=ci;
199     u2=ci2;
    while(t<=duree){
201         diffSolve();
        cout << t << " " << generateur->Esm(t) << " " << u << " " << sin(t) << endl;
203     }
}
205
/*Constructeur du circuitC*/
207 circuitC::circuitC(){ //Cas special de l'exercice 2
    cout << "#Valeur de R (Ohm) : " << endl;
209     cin >> R ;
    cout << "#Valeur de L (Henry) : " << endl;
211     cin >> L ;
    cout << "#Valeur de C (Farad) : " << endl ;
213     cin >> C ;

215     a=-R/L;
    b=-1/(L*C);
217     ci=0.0;
    ci2=0.0;

219     generateur->setAB(1,0);
221 }

223 circuitD::circuitD(){
    cout << "#Valeur de R (Ohm) : " << endl;
225     cin >> R ;
    cout << "#Valeur de L (Henry) : " << endl;
227     cin >> L ;
    cout << "#Valeur de C (Farad) : " << endl ;
229     cin >> C ;

231     a=-1/(R*C);
    b=-1/(L*C);
233     ci=0.0;
    ci2=0.0;

235     generateur->setAB(-a,0);
237 }

239 void circuitD::diffSolve(){
    up=u;
241     u2p=u2;
    u=up+pas*u2p;
243     u2=u2p+pas*(b*up+a*u2p)+(generateur->Esm(t)-generateur->Esm(t-pas)); //on code la deriv de la fct
    second membre
    t=t+pas;
245 }

```

D sources.h

```
1  /* Programmation orientee 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 mere : 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 differents signaux d'entree. */
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
45 class carre : public source{
46     public:
47         double E(double t);
48 };
49 #endif
```


E sources.cpp

```
1  /* Programmation orientee objet : BE2 */
2  /* Jeremie Fourmann et Maxime Morin   */
3  /* sources.cpp                         */
4  /* Definition des classes sources      */
5
6  #include <iostream>
7  #include "sources.h"
8  #include <math.h>
9
10 using namespace std;
11
12
13 /* Methodes de la classe mere "source". */
14 source::source(){
15     T=2;
16     phi=1;
17     offset=0;
18     ampli=5;
19     alpha=.6;
20     A=1, B=0;
21 }
22
23 double source::Esm(double t) // Transformation affine du signal de la source
24 {
25     return A*E(t)+B;
26 }
27
28
29 /* Definitions des sources filles pour differents types de signaux ou fonctions. */
30
31 double fctExo1::E(double t){
32
33     return -3*t;
34 }
35
36 void source::setAB(double Ai, double Bi)
37 {
38     A = Ai;
39     B = Bi;
40 }
41
42
43 double echelon::E(double t){
44     double fx;
45     if(phi <=t ) fx= offset+ampli;
46     else fx= offset;
47     return fx;
48 }
49
50
51 double porte::E(double t){
52     double fx;
53     if(phi < t && t <phi+T) fx=offset+ampli;
54     else fx=offset;
55     return fx;
56 }
57
58 double carre::E(double t){
59     double fx;
60     if((t-phi)-floor((t-phi)/T)*T<T*alpha) fx=offset+ampli;
61     else fx=offset;
62     return fx;
63 }
```

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