Projet de programmation C++

Résolution de circuit

Jérémie Fourmann (Promo 2013 - Eléctronique - Enseeiht)
Maxime Morin (Promo 2013 - Eléctronique - Enseeiht)

23 novembre 2011

Plan

1 Objectif

2 Organistion du code

2.1 L'objet circuit

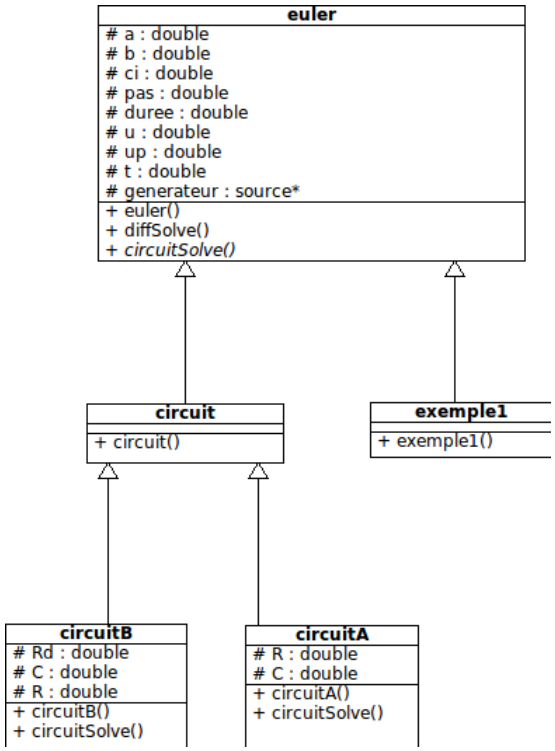

FIGURE 1 – Hièrarchie de la classe circuit

2.2 L'objet source

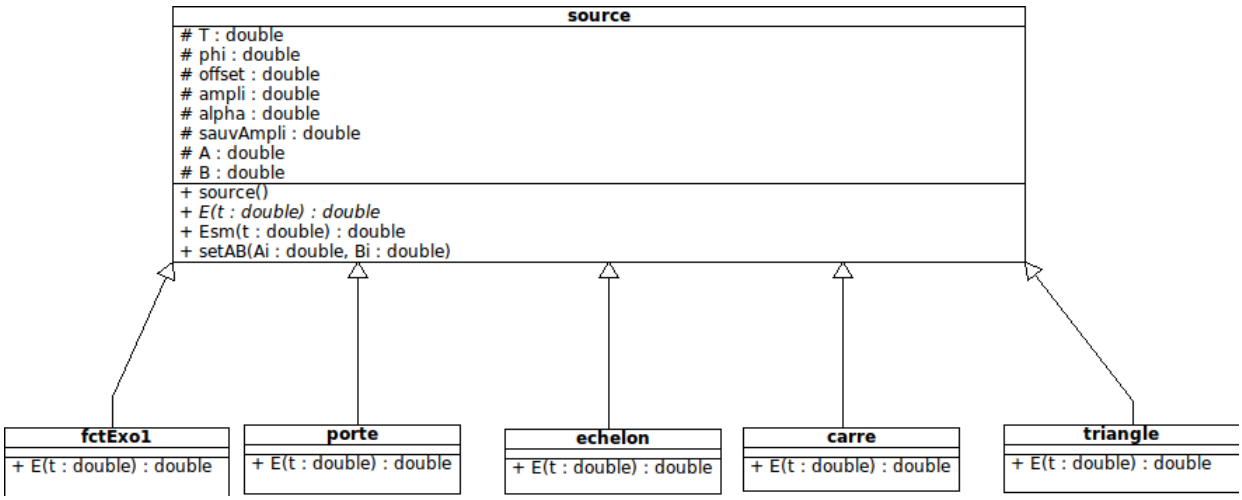

FIGURE 2 – Hièrarchie de la classe source

3 Code Source

3.1 main.cpp

```
/* Programmation ortientee objet : BE2 */
/* Jeremie Fourmann et Maxime Morin    */
/* main.cpp                            */
/* Programme principal                 */


#include <iostream>
#include "circuits.h"
#include "sources.h"

using namespace std;

int main(int argc, char **argv)
{
    cout.width(6);
    cout.precision(4);

    circuit * montage;
    int choix=0;

        cout << "#Quel montage ?" << endl;
        cout << "#1 - Circuit A" << endl;
        cout << "#2 - Circuit B" << endl;
        cin >> choix;

        switch(choix){
        case 1:
            montage = new circuitA;
            break;
        case 2:
            montage = new circuitB;
            break;
        default:
            cout << "#Mauvaix choix" << endl;
            return 0;
        }


    montage->circuitSolve();

    return 0;
}
```

3.2 circuits.h

```
/* Programmation ortientee objet : BE2 */
/* Jeremie Fourmann et Maxime Morin    */
/* circuits.h                          */
/* Declaration des classes circuits    */

#ifndef DEF_circuits
#define DEF_circuits
#include "sources.h"

/* Classe "euler" pour la resolution de au'+bu=f. */
class euler{
    protected:
        double a,b,ci,pas,duree,u,up,t ;
        source *generateur;
    public:
        euler();
        void diffSolve();
        virtual void circuitSolve() =0;
};

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

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

/* Classe "circuitA". */
class circuitA : public circuit{
    protected:
        double R,C;
    public:
        circuitA();
        void circuitSolve();
};

/* Classe "circuitB". */
class circuitB : public circuit{
    protected:
        double Rd,C,R;
    public:
        circuitB();
        void circuitSolve();
};

#endif
```

3.3 circuits.cpp

```
/* Programmation ortientee objet : BE2 */
/* Jeremie Fourmann et Maxime Morin    */
/* circuits.cpp                        */
/* Definition des classes circuits     */

#include <iostream>
#include <math.h>
#include "circuits.h"

using namespace std;

euler::euler(){
    a=0.0;
    b=0.0;
    ci=0.0;
    pas=0.1;
    duree=10.0;
    u=0.0;
    up=0.0;
    t=0.0;
}

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

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

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

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

        switch(choix){
        case 1:
            generateur=new echelon;
            break;
        case 2:
            generateur=new porte;
            break;
        case 3:
            generateur=new carre;
            break;
        case 4:
            generateur=new triangle;
            break;
        default:
            break;
        }
}

/* Circuit A avec comme paramtres R et C */
```

```
65  circuitA::circuitA(){
        cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
67      cout << "#_____/\\/\\/\\____  " << endl ;
        cout << "#|      R      _|_" << endl ;
69      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. */
83  void circuitA::circuitSolve(){

85      cout << "#Temps" << "   " << "Ve" <<"   " << "Vs" << "   " << endl;
        while(t<=   duree){
87          diffSolve();
            cout << t << "   " << generateur->E(t) <<"   " << u << endl;
89      }
    }
91

93  /* Circuit B avec comme paramtres Rd, R et C. */
    circuitB::circuitB(){
95      cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
        cout << "#_____/\\/\\/\\____|\\_____________  " << endl ;
97      cout << "#|      Rd      |/       |     |" << endl ;
        cout << "#|                  D     /    _|_" << endl ;
99      cout << "#E                      R \\    --- C " << endl ;
        cout << "#|                        /     |" << endl ;
101     cout << "#|________________________|_____|" << endl ;

103     cout << "#Valeur de Rd (Ohm) : " << endl;
        cin >> Rd ;
105     cout << "#Valeur de R (Ohm) : " << endl;
        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

```
/* Programmation ortiente objet : BE2 */
/* Jeremie Fourmann et Maxime Morin    */
/* sources.h                           */
/* Declaration des classes sources     */

#ifndef DEF_sources
#define DEF_sources

/* Classe mre : source. */
class source{

    protected:
        double T,phi,offset,ampli,alpha,sauvAmpli;
        double A,B;
    public:
        source();
        virtual double E(double t)=0;//fct virtuelle de la source
        double Esm(double t); // Transformation affine de E pour changer amplitude
                              // ou ajouter un offset dans le second membre
        void setAB(double Ai, double Bi); //accesseur pour les valeurs A et B
};

/* Classe fille permettant de trater l'exemple 1. */
class fctExo1 : public source{
    public:
         double E(double t);
};

/* Classes filles pour les differnets signaux d'entree. */
class echelon : public source{
    public:
         double E(double t);
};

class porte : public source{
    public:
         double E(double t);
};

class triangle : public source{
    public:
         double E(double t);
};
class carre : public source{
    public:
         double E(double t);
};

#endif
```

3.5 sources.cpp

```
/* Programmation ortientee objet : BE2 */
/* Jeremie Fourmann et Maxime Morin    */
/* sources.cpp                         */
/* Definition des classes sources      */

#include <iostream>
#include "sources.h"
#include <math.h>

using namespace std;


/* Methodes de la classe mre "source". */
source::source(){
    T=2;
    phi=1;
    offset=0;
    ampli=5;
    alpha=.6;
    A=1, B=0;
}

double source::Esm(double t) // Transformation affine du signal de la source
{
    return A*E(t)+B ;
}

/* Definitions des sources filles pour differents types de signaux ou fonctions. */

double fctExo1::E(double t){

    return -3*t;
}

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

double echelon::E(double t){
    double fx;
    if(phi <=t ) fx= offset+ampli;
    else fx= offset;
    return fx;
}

double porte::E(double t){
    double fx;
    if(phi < t && t <phi+T) fx=offset+ampli;
            else fx=offset;
    return fx;
}

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

double triangle::E(double t){
    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

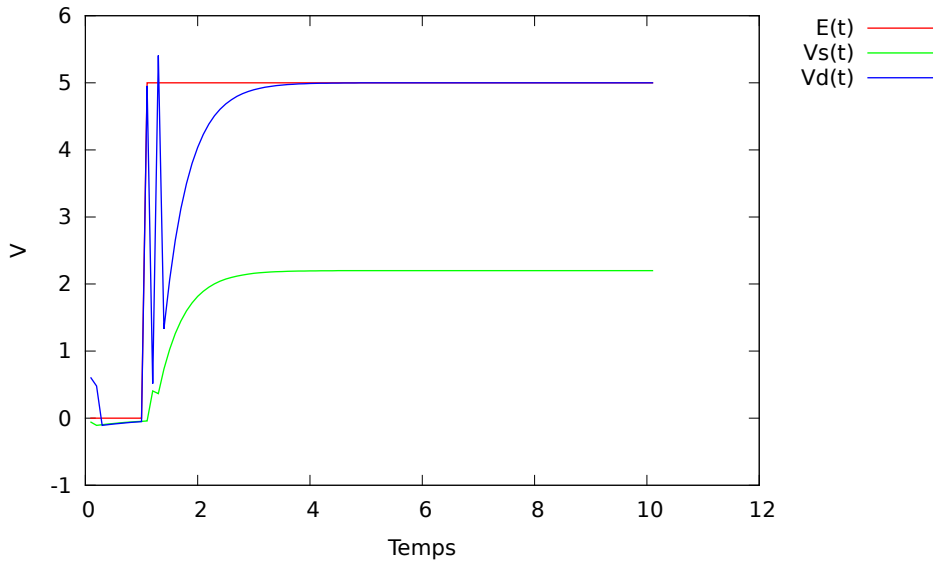

FIGURE 3 – Réponse à un echelon de tension

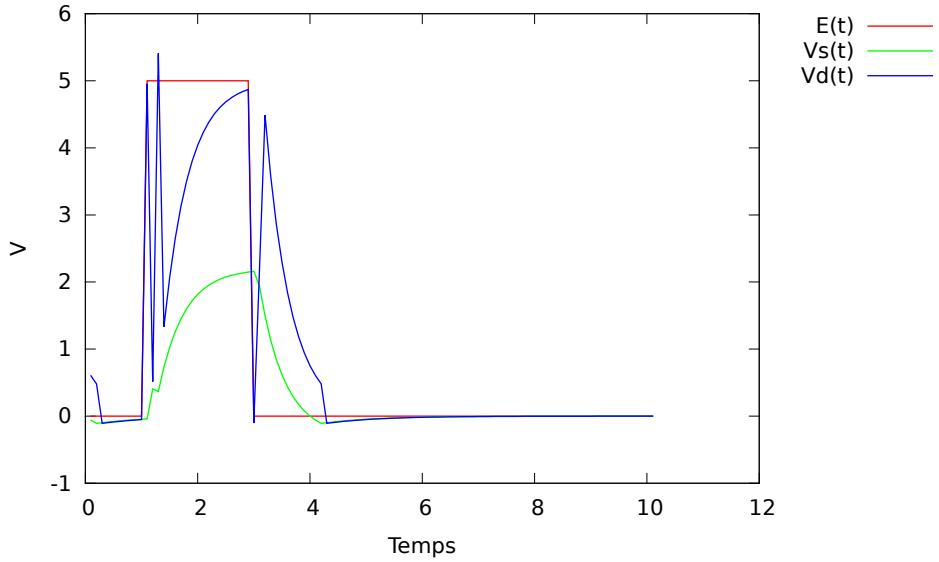

FIGURE 4 – Réponse à une porte

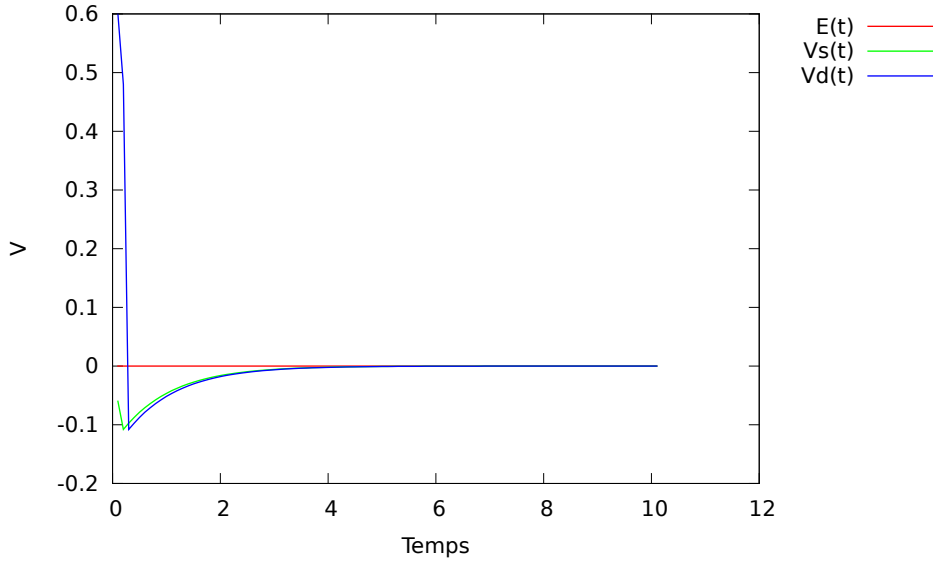

FIGURE 5 – Réponse à signal carré

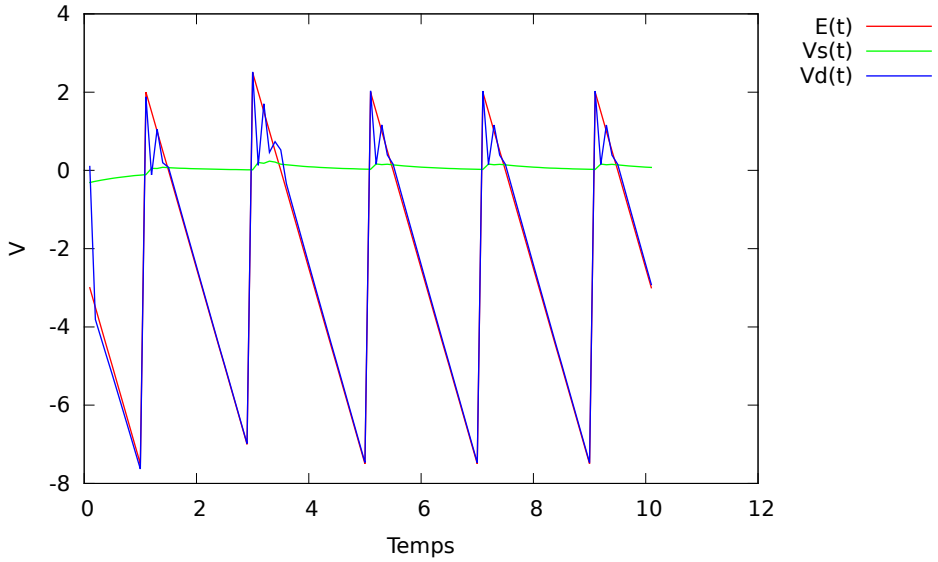

FIGURE 6 – Réponse à signal triangle

4.4 Réponse du CircuitC

4.5 Réponse du CircuitD