# Projet de programmation C++

Résolution de circuit

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

5 décembre 2011

## Plan

# 1 Objectif

# 2 Organistion du code

## 2.1 L'objet circuit

FIGURE 1 – Hièrarchie de la classe circuit

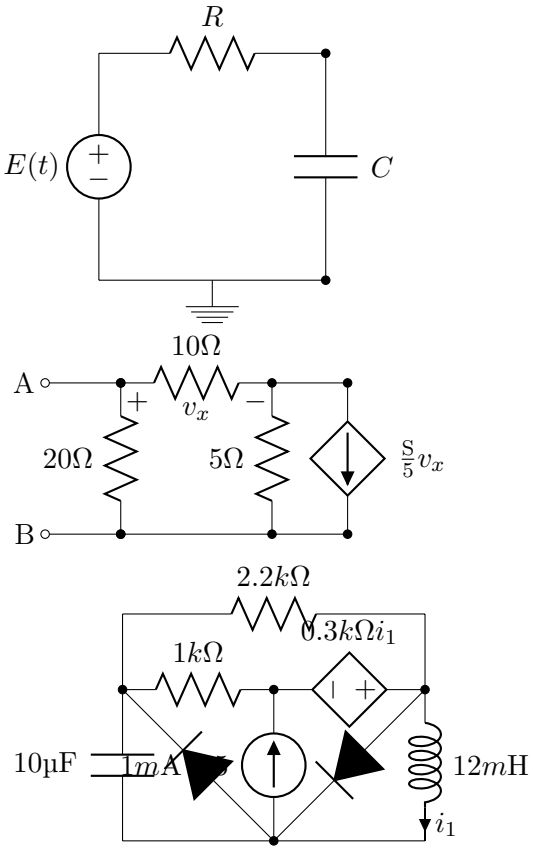
## 2.2 L'objet source

FIGURE 2 – Hièrarchie de la classe source

# 3 Résultats

## 3.1 Réponse du l'exemple 1

## 3.2 Réponses du CircuitA




## 3.3 Réponse du CircuitB

## 3.4 Réponse du CircuitC

## 3.5 Réponse du CircuitD

# A 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 << "#Premier Ordre :" << endl;
        cout << "#1 - Exemple 1" << endl;
        cout << "#2 - Circuit A" << endl;
        cout << "#3 - Circuit B" << endl;
        cout << "#Deuxime Ordre :" << endl;
        cout << "#4 - Exemple 2" << endl;
        cout << "#5 - Circuit C" << endl;
        cout << "#6 - Circuit D" << endl;
        cin >> choix;

        switch(choix){
        case 1:
            montage = new exemple1;
            break;
        case 2:
            montage = new circuitA;
            break;
        case 3:
            montage = new circuitB;
            break;
        case 4:
            montage = new exemple2;
            break;
        case 5:
            montage = new circuitC;
            break;
        case 6:
            montage = new circuitD;
            break;
        default:
            cout << "#Mauvaix choix" << endl;
            return 0;
        }


    montage->circuitSolve();

    return 0;
}
```

## B 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 pas,duree,t ;
        source *generateur;
    public:
        euler();
        virtual void diffSolve()=0;
        virtual void circuitSolve()=0;
};


/* Classe "circuit" (permet le choix de la source) */
class circuit : public euler{
    protected:
        double a,b,ci,u,up;
    public:
        circuit();
        virtual void diffSolve()=0;
        virtual void circuitSolve()=0;
};

/* Classe "circuit1" (1er ordre) */
class circuit1 : public circuit{
    public:
        void diffSolve();
        virtual void circuitSolve() =0; //defini en fct du circuit
};

/* Classe "exemple1". */
class exemple1 : public circuit1{
    public:
        exemple1();
        void circuitSolve();
};


/* Classe "circuitA". */
class circuitA : public circuit1{
    protected:
        double R,C;
    public:
        circuitA();
        void circuitSolve();
};

/* Classe "circuitB". */
class circuitB : public circuit1{
    protected:
        double Rd,C,R;
    public:
        circuitB();
        void circuitSolve();
};
```

```
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

```
/* 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(){
    pas=0.01;
    duree=10;
    t=0.0;
}

/* Choix de la source lors de la creation d'un circuit. */
circuit::circuit(){
    int choix=0;
    a=0.0;
    b=0.0;
    ci=0.0;
    u=0.0;
    up=0.0;

        cout << "#Choisir la source ?" << endl;
        cout << "#1 - Echelon" << endl;
        cout << "#2 - Porte" << endl;
        cout << "#3 - Carre" << endl;
        cout << "#4 - Triangle" << endl;
        cout << "#5 - Rampe f(t)=-3*t (Exemple1)" << endl;
        cout << "#6 - Nulle (Exemple 2)" << 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;
        case 5:
            generateur=new fctExo1;
            break;
        case 6:
            generateur=new echelon; /* Generateur quelconque. */
            generateur->setAB(0,0); /* Coupe le generateur. */
            break;
        default:
            break;
        }

}

void circuit1::diffSolve(){
        up=u;
        u=(pas/a)*(generateur->Esm(t)+up*(-b+a/pas));
```

```
65          t=t+pas;
    }
67
    exemple1::exemple1(){   //Cas "mathematique" de l'exercice 1
69      a=1;
        b=3;
71      ci = 0;
    }
73
    void exemple1::circuitSolve(){
75          cout << "#Temps" << "    " << "SolEuler" <<"    " << "SolExacte" << "    " << endl;
        while(t<=   duree){
77          diffSolve();
            cout << t << "    " << u <<"    " << -(1/3)*exp(-3*t) -t + (1/3) << endl;
79      }
    }
81

83  /* Circuit A avec comme parametres R et C */
    circuitA::circuitA(){
85      cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
        cout << "#_____/\\/\\/\\____  " << endl ;
87      cout << "#|      R      _|_" << endl ;
        cout << "#E           C ---" << endl ;
89      cout << "#|_______________|" << endl ;

91      cout << "#Valeur de R (Ohm) : " << endl;
        cin >> R ;
93      cout << "#Valeur de C (Farad) : " << endl ;
        cin >> C ;
95
        a=R*C;
97      b=1;
        generateur->setAB(1,0); // Esm(t) = E(t)
99  }

101 /* Resolution de l'equation differentielle du circuitA pour la source choisie. */
    void circuitA::circuitSolve(){
103
        cout << "#Temps" << "    " << "Ve" <<"    " << "Vs" << "    " << endl;
105     while(t<=   duree){
            diffSolve();
107         cout << t << "    " << generateur->E(t) <<"    " << u << endl;
        }
109 }

111
    /* Circuit B avec comme paramtres Rd, R et C. */
113 circuitB::circuitB(){
        cout << "#Choix des valeurs pour le circuit suivant :" << endl ;
115     cout << "#_____/\\/\\/\\____|\\_____________  " << endl ;
        cout << "#|      Rd      |/       |     |" << endl ;
117     cout << "#|                  D     /    _|_" << endl ;
        cout << "#E                      R \\    --- C " << endl ;
119     cout << "#|                        /     |" << endl ;
        cout << "#|________________________|_____|" << endl ;
121
        cout << "#Valeur de Rd (Ohm) : " << endl;
123     cin >> Rd ;
        cout << "#Valeur de R (Ohm) : " << endl;
125     cin >> R ;
        cout << "#Valeur de C (Farad) : " << endl ;
127     cin >> C ;
    }
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
    /*Ciruit 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

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

#ifndef DEF_sources
#define DEF_sources

/* Classe mere : 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 traiter l'exemple 1. */
class fctExo1 : public source{
    public:
         double E(double t);
};

/* Classes filles pour les differents 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
```

# E 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 mere "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;
}
```

```
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;
   }
```