Projet de programmation C++

Résolution de circuits

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

2 janvier 2012

Plan

1 Objectif

Nous devons réaliser un programme en C++ permetant de résoudre des équations différentielles du 1^{er} et du 2^{e} ordre à coefficients constants. Nous allons utliser une méthode de résolution numérique de type différences finies. Nous utliserons plus particulierement la méthode d'Euler.

L'utilisateur du programme pourra via un terminal :

- choisir le type de circuit (1^{er} et du 2^{e} ordre)
- choisir les valeurs des composants
- choisir le type de source (c'est a dire le second membre de l'equation différentielle)

Remarque : Le pas et la durée de la simulation sont réglés par défaut. L'utilisateur n'y a pas accès. Nous n'avons pas voulu implémenter cette fonctionnalité pour ne pas surcharger le programme, d'autant plus qu'elle n'apporte pas plus de concept orienté objet.

Le programme ecrira dans le terminal ou dans un fichier formaté la solution numérique trouvée. Il sera alors possible de la tracer (voir annexe Gnuplot).

Nous allons à présent détailler et expliquer la structure de notre programme et les choix que nous avons a fait. Enfin nous commenterons les résultat obtenus sur les différents circuits. Une fois notre code opérationel nous regarderons les limites de la méthode d'Euler.
En annexe nous détaillons notre méthode pour obtenir et tracer les résultats via un script élementaire Unix et le programme Gnuplot.

2 Organistion du code

2.1 Le concept général

Nous devons nous familiariser avec les notions de programmation orientée objet : classe, héritage, polymorphysme. En effet en fonction du type de circuit la méthode de résolution n'est pas la même, nous faisons donc appel a la notion de polymorphysme pour résoudre se problème. Dans notre cas les fonctions circuitSolve() et diffSolve() auront plusieurs versions possible en fonction du circuit.

2.2 L'objet circuit

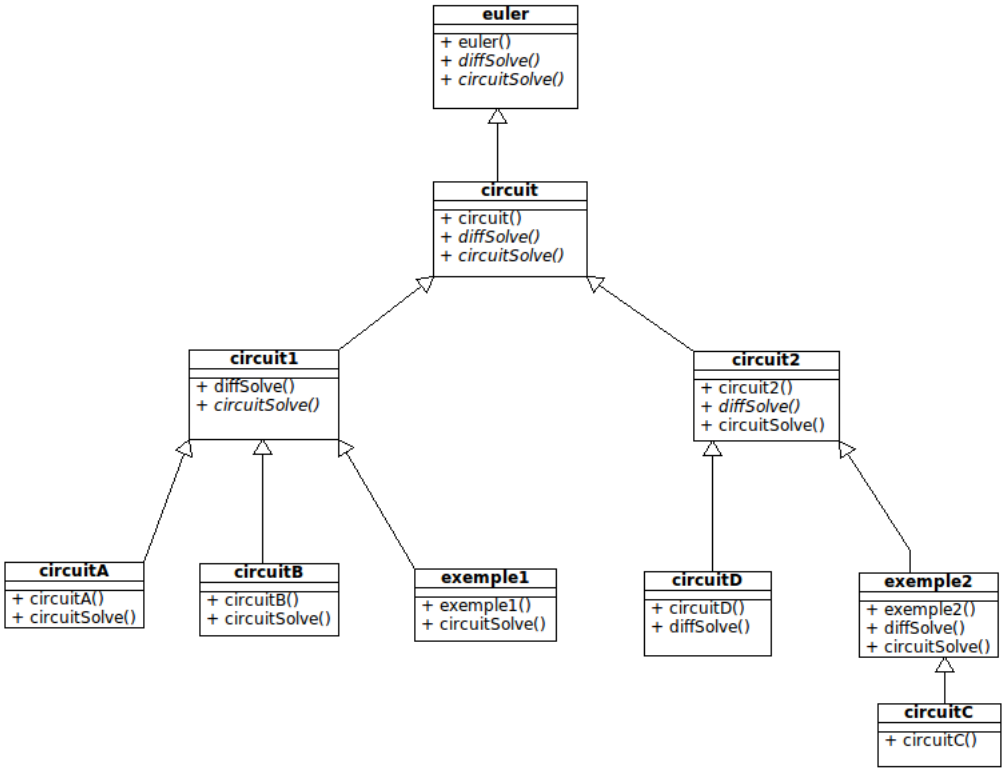


FIGURE 1 – Hièrarchie de la classe circuit

Les principales caractéristique de cet objet :

La classe Euler :
- permet de définir les paramètres de simulation
- les méthodes circuitSolve() et diffSolve() sont virtuelles et permettrons la résolution du problème en fonction de la classe instanciée

La classe Circuit :
- Hérite d'Euler
- Son constructeur permet le choix de la Source via un pointeur

La classe Circuit1 :
- Hérite de circuit
- définition de la fonction circuiSolve() qui permet de résoudre : $a \cdot u' + b \cdot u = E$

La classe Circuit1 :
- Hérite de circuit
- définition de la fonction circuiSolve(),qui permet de calculer résoudre : $a \cdot u'' + b \cdot u' + c \cdot u = E$

Les autres classes, héritant de ces 2 dernières, permettent de résoudre les circuit A,B,C,D, nous avons essayé de relier les exemples au cas général en considérant leurs seconds membres commes des sources particulières.

2.3 L'objet source

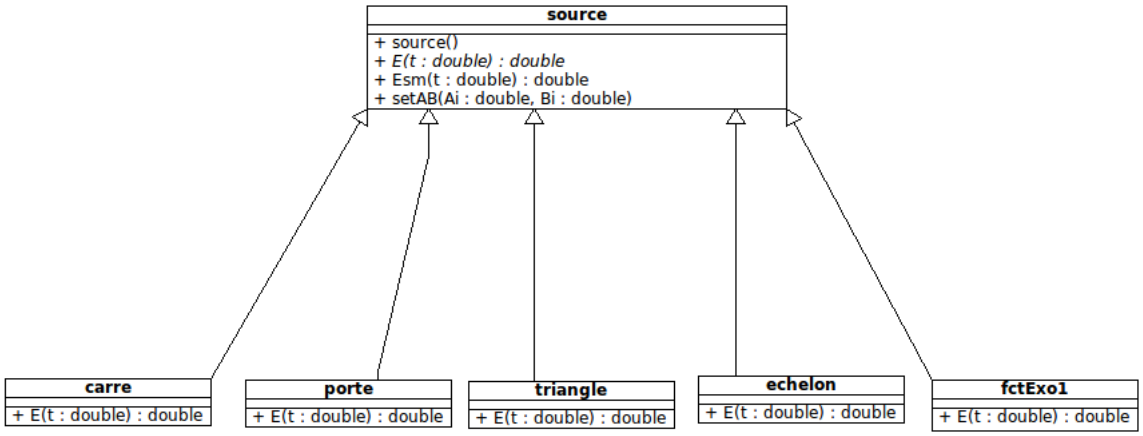


FIGURE 2 – Hièrarchie de la classe source

Les classes carré, échelon... héritent de la classe mère source. Elles définissent chacune une version de la fontion E(t) et règle certains attributs qui leurs sont propres comme l'amplitude, fréquences, l'offset...
La fonction E(t) renvoit la valeur à l'instant t de la source. L'objet Circuit va avoir comme attributs une source (ou plutôt un pointeur sur un source), car elle intervient dans la définition de l'équation différentielle à résoudre (seconde membre). La méthode Esm(t) est une transformation affine de la source : $Esm(t) = A \cdot E(t) + B$.
A et B sont des attributs privés de la méthode. Ils sont modifiables par l'accesseur setAB(,). Cette transformation permet de simuler une extinction de la source ou un offset (cas de la diode passante par exemple) sans modifier ses paramètres intrinsèques.

2.4 Le programme principal (main)

Notre *main* crée un pointeur vers un type de montage et l'initialise selon de choix de l'utilisateur. Il execute ensuite la methode de résolution et d'affichage de la solution du circuit qui est diffsolve().

Remarque : La seule sortie du programme est *stdout* non redéfinie. Pour alléger au maximum le programme, nous n'avons pas voulu utiliser une sortie supplémentaire (vers un fichier via *fpritf* par exemple). Lorsque nous voulons collecter les données de sortie du programme dans un fichier nous utilisons la syntaxe Unix :

```
1  ./programme > fichier.txt
```

3 Résultats

3.1 Exemple 1

Résolution de l'équation différentielle du 1er ordre :

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

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

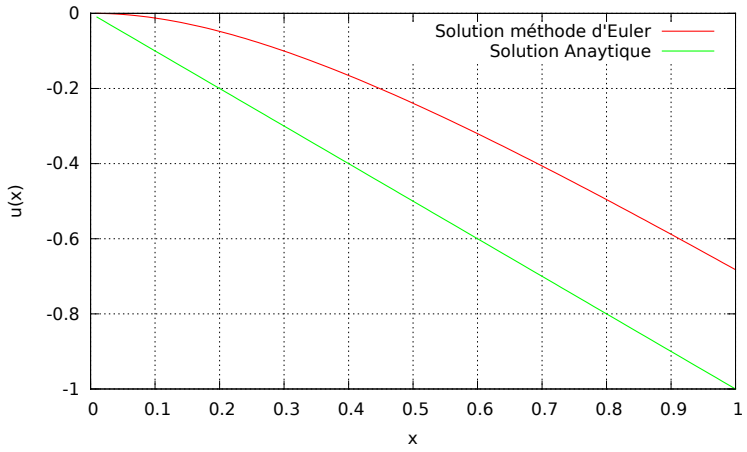


FIGURE 3 – Solution de l'exemple 1

3.2 Exemple 2

Résolution de l'équation différentielle du 1er ordre :

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

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

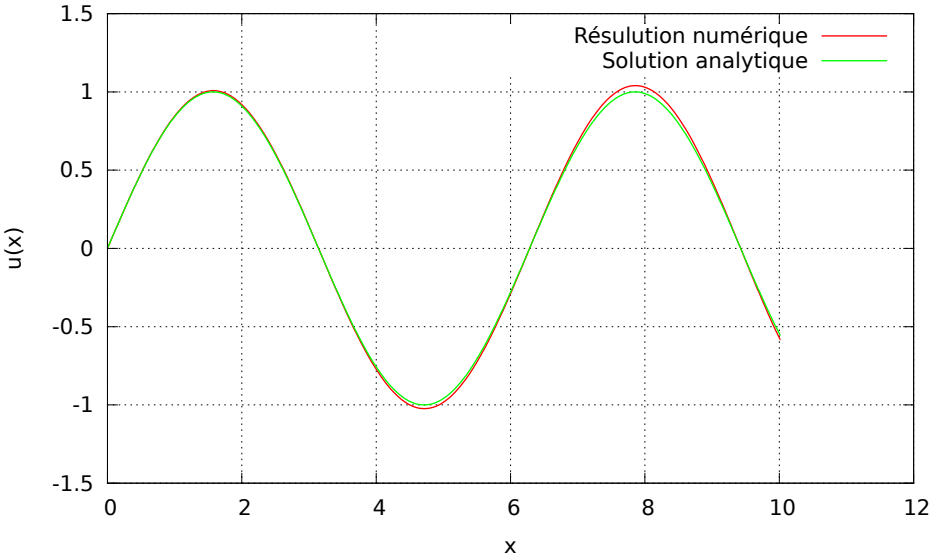
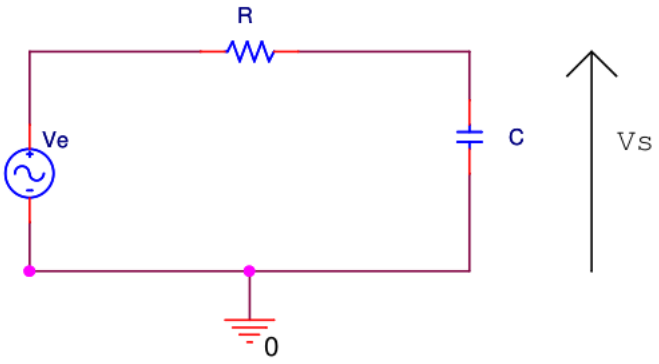


FIGURE 4 – Solution de l'exemple 2

3.3 Réponse du CircuitA



Le circuit A est un circuit RC du 1er ordre, régit par l'équation différentielle :

$$Vs + RC \cdot Vs' = Ve$$

Nous allons étudier la réponse du circuit A à plusieur type d'exitation, avec comme paramètres de simulation :

- $R = 1\Omega$ et $C = 1F$
- $pas = 0.01s$,$dure = 10s$

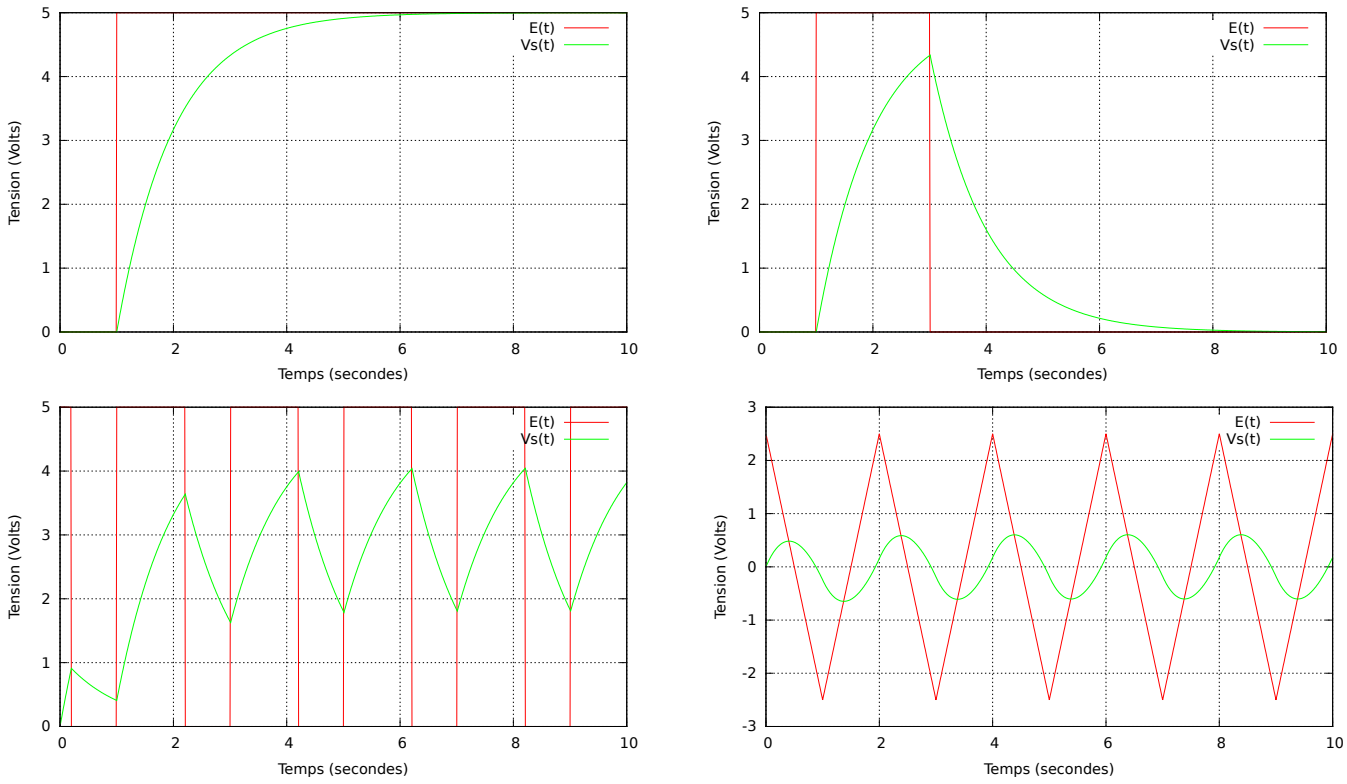
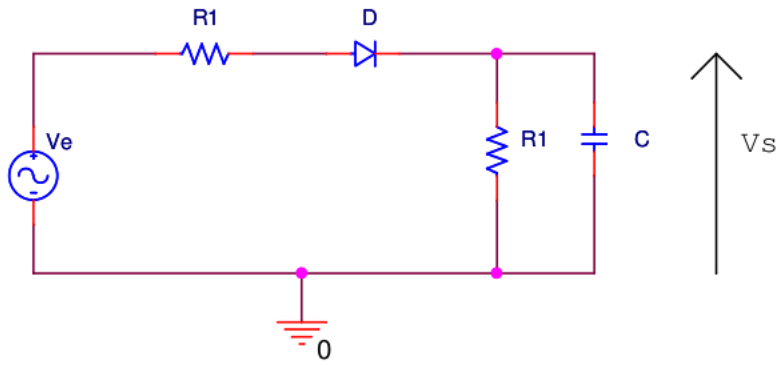


FIGURE 5 – Réponse du circuit A

3.4 Réponse du CircuitB



Le circuit B est un circuit RC du 1er ordre, il n'est pas linéaire à cause de la diode, nous devons donc l'étudier dans 2 cas.
1er cas : diode passante ($Vd > 0.6$), on a une équation de charge de la capacité.
2e cas : diode bloqué($Vd < 0.6$), on a une équation de décharge de la capacité dans la résistance.
Notre programme test la condition de Vd pour savoir quelle équation il doit résoudre. Ici on doit prendre en compte les conditions initiales, car on considère le changement d'équation comme un changement de condition initiales et une transformation affine de la source.

Nous allons étudier la réponse du circuit B à plusieurs type d'exitations, avec comme paramètres de simulation :

- $R = 1\Omega$ et $C = 1F$
- $pas = 0.01s$,$dure = 10s$

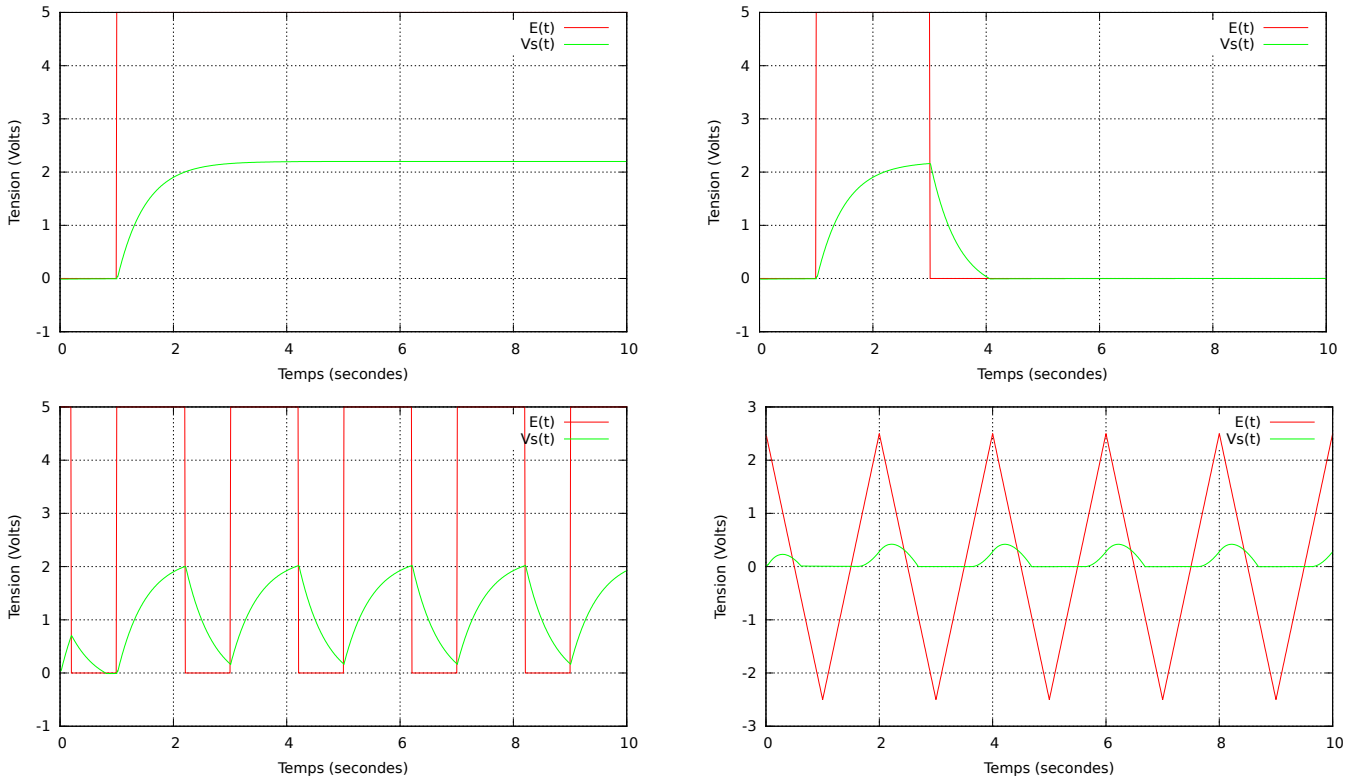
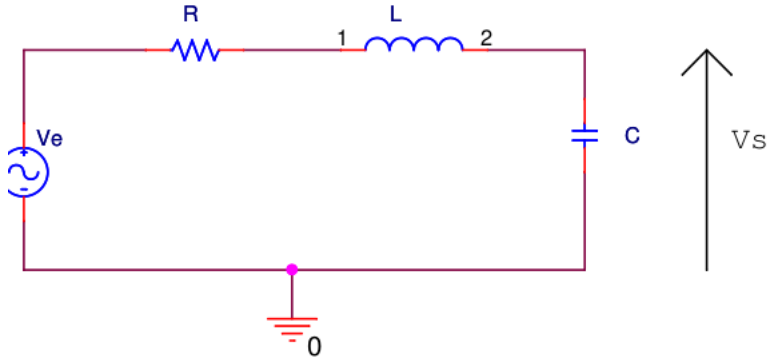


FIGURE 6 – Réponse du circuit B

3.5 Réponse du CircuitC



Le circuit C est un circuit RLC du 2e ordre, régit par l'équation différentielle :

Todo

Nous allons étudier la réponse du circuit C à plusieur types d'exitations, avec comme paramètres de simulation :

- $R = 1\Omega$ et $C = 1F$
- $pas = 0.01s$,$dure = 10s$

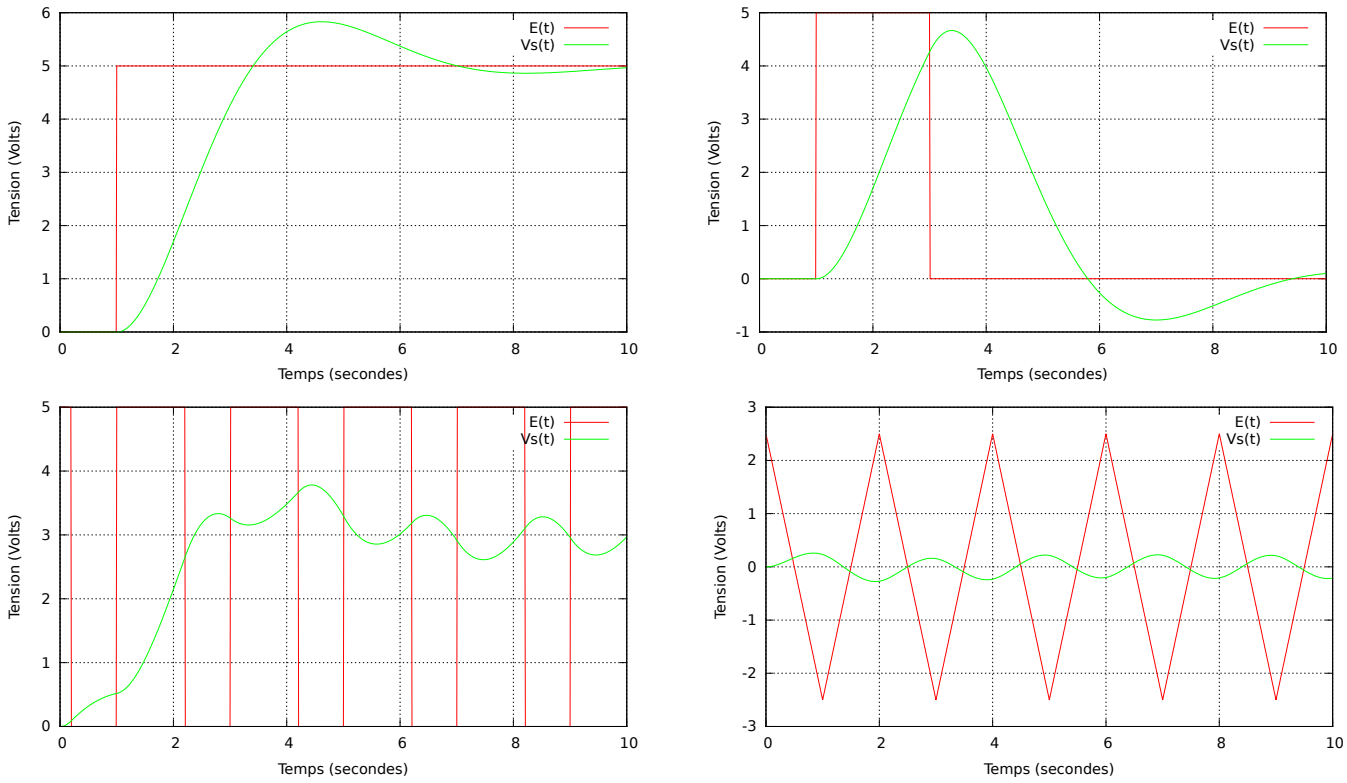
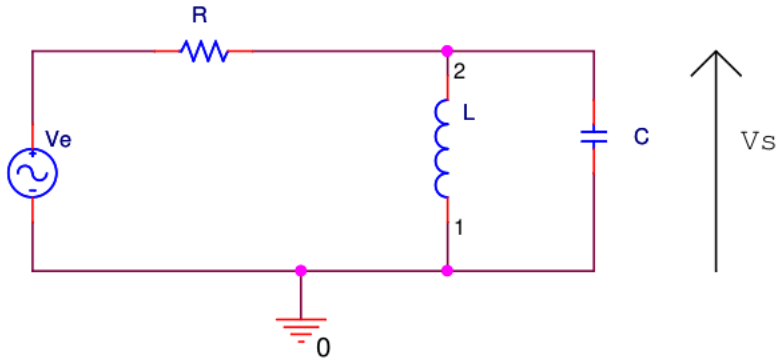


FIGURE 7 – Réponse du circuit C

3.6 Réponse du CircuitD



Le circuit D est un circuit RLC du 2e ordre, régit par l'équation différentielle :

Todo

Nous allons étudier la réponse du circuit D à plusieur types d'exitations, avec comme paramètres de simulation :

- $R = 1\Omega$ et $C = 1F$
- $pas = 0.01s$,$dure = 10s$

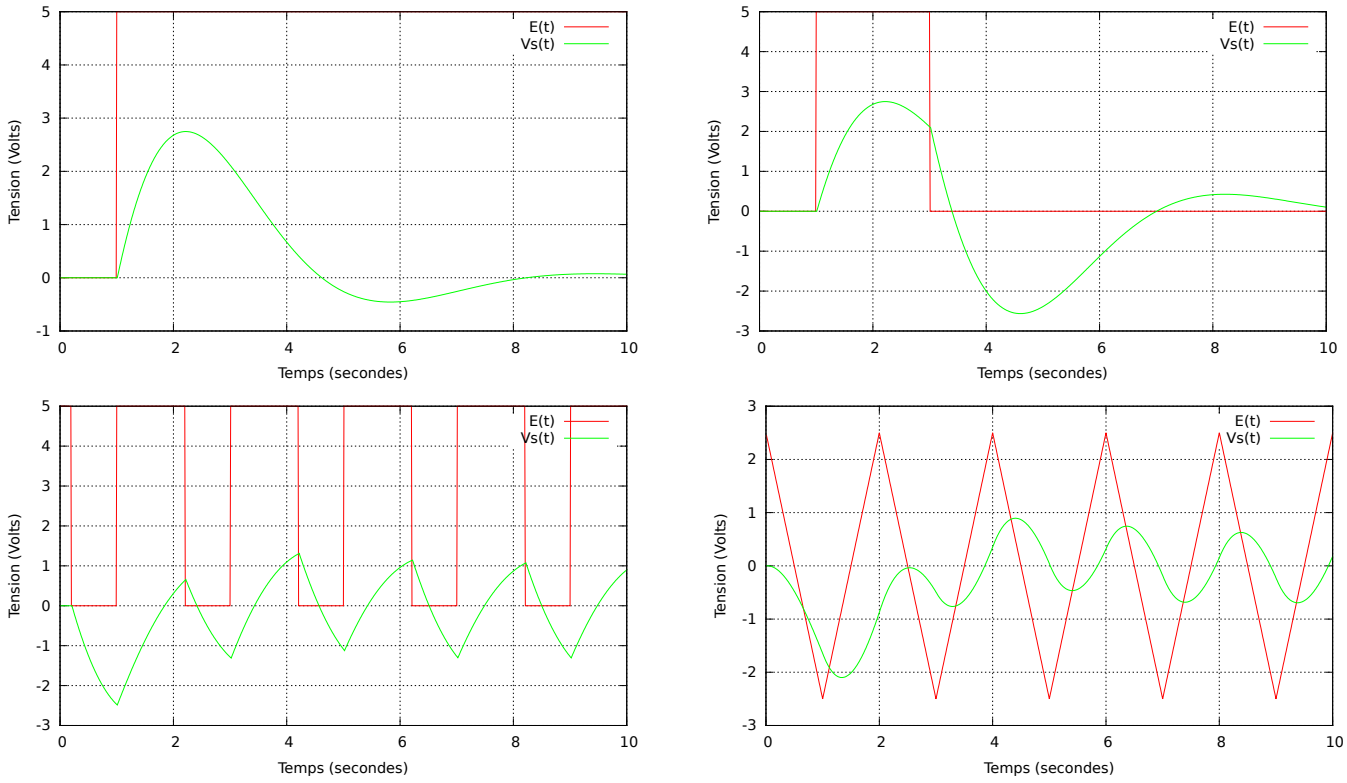


FIGURE 8 – Réponse du circuit D

4 Bilan

4.1 Conclusion

Nous sommes satisfaits de notre programme, les résultat obtenus lors des diverses simulations sont cohérents avec la réalité physique. La structure de notre programme et le formalisme de la POO nous a permis d'avoir un code léger, qui peut s'adapter et se modifier simplement.
Par exemple si on voulait rajouter un nouveau type de source, seul l'objet source serait à modifier (*principe de l'encapsulation*). *L'héritage* et *le polymorphysme* nous on permis de réduire considérablement la taille de notre code tout en gagnant en lisibilité.

En ce qui concerne la méthode numérique utilisée (méthode d'Euler), on remarque qu'elle s'implémente très simplement et donne de bon résultat. On remarque la précision de la résolution dépend du pas de calcul.
On utilise l'approximation suivante :

$$u'(x) = \frac{u(x+h) - u(x)}{h} + O(h)$$

$$u'(x) \approx \frac{u(x+h) - u(x)}{h}$$

La méthode numérique est donc consistante en h, c'est à dire que l'erreur est divisé par **a** quand le pas est divisé par **a** le nombre d'opération est quand a lui multiplié par **a** .

4.2 Amélioration possible

4.2.1 Programme

Certaines options pourraient encore être impélmentées sur le programme, notament l'accès aux paramètres de simulation. Cependant on est loin d'un "logiciel", en commençant par l'interface graphique et l'affichage automatique des tracés, la liste des améliorations est encore longue et n'est pas du domaine des électroniciens.

4.2.2 Sur le plan numérique

Si on change de schéma numérique, repartons du devellopement limité de u (on concidère que u estC^3) :

$$u(x+h) = u(x) + h \cdot u'(x) + \frac{h^2}{2} \cdot u''(x) + O(h^3)$$

$$u(x-h) = u(x) - h \cdot u'(x) + \frac{h^2}{2} \cdot u''(x) + O(h^3)$$

On obtient l'approximation suivante :

$$u'(x) = \frac{u(x+h) - u(x-h)}{2h} + O(h^3)$$

Ce schéma serait consistant en h^3 ! la precision serait grandement améliorée comparait à la méthode précédente tout en gardant la meme nombre de calculs.

A Listing du programme

A.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 << "#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;
}
```

A.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 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
```

A.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(){
    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 }
```

A.4 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
```

A.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 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;
   }
```

B Script de collecte des données et Gnuplot

En fainéants et pauvres étudiants que nous sommes, nous cherchions une solution rapide, gratuite et disponible chez soi pour générer les tracés demandés. Notre programme prend en entrée une combinaison correspondant au choix de l'utilisateur et renvoi les données dans la sortie standart. Par exemple pour demander au programme de nous afficher les résultats de l'exemple 1 on utilise le *pipe* qui dirige la sortie d'un programme vers l'entrée d'un autre :

```
echo 1 5 | ./main
```

En se plaçant évidemment au préalable dans le répertoire contenant notre fichier objet apellé *main*.
Pour envoyer ces résultats dans un fichier on utilise le chevron :

```
1  echo 1 5 | ./main > donnees.dat
```

Automatiser la procedure revient à exécuter cette commande pour toutes les données demandées, c'est quasiment du copier-coller :

B.1 collecte.sh

```
1
3  echo 2 1 1 1 | ../main > ../plot/CAechelon.dat;
   echo 2 2 1 1 | ../main > ../plot/CAporte.dat;
5  echo 2 3 1 1 | ../main > ../plot/CAcarre.dat;
   echo 2 4 1 1 | ../main > ../plot/CAtriangle.dat;
7
   echo 3 1 1 1 1 | ../main > ../plot/CBechelon.dat;
9  echo 3 2 1 1 1 | ../main > ../plot/CBporte.dat;
   echo 3 3 1 1 1 | ../main > ../plot/CBcarre.dat;
11 echo 3 4 1 1 1 | ../main > ../plot/CBtriangle.dat;

13 echo 4 6 | ../main > ../plot/exemple2.dat;

15 echo 5 1 1 1 1 | ../main > ../plot/CCechelon.dat;
   echo 5 2 1 1 1 | ../main > ../plot/CCporte.dat;
17 echo 5 3 1 1 1 | ../main > ../plot/CCcarre.dat;
   echo 5 4 1 1 1 | ../main > ../plot/CCtriangle.dat;
19
   echo 6 1 1 1 1 | ../main > ../plot/CDechelon.dat;
21 echo 6 2 1 1 1 | ../main > ../plot/CDporte.dat;
   echo 6 3 1 1 1 | ../main > ../plot/CDcarre.dat;
23 echo 6 4 1 1 1 | ../main > ../plot/CDtriangle.dat;

25 gnuplot aTrace;
   rm images/concat.pdf
27 pdftk images/*.pdf cat output images/concat.pdf;
```

Les dernières lignes apellent GNUplot. Le fichier *aTrace* dit au logiciel ou se trouvent les données, où il doit les tracer, leur format de sortie, les légendes, etc... Lui aussi est répétitif.

B.2 aTrace

```
2  set terminal pdf
   #set key off
4  a=2 #regle epesseur des traces
   set key top right #positionne la legende
6  set grid

8  set xlabel "x"
   set ylabel "u(x)"
10
   set output "images/exemple1.pdf"
```

```
plot "../plot/exemple1.dat" using 1:2 title "Solution mthode d'Euler" with lines linewidth a , "../plot/exemple1.dat" using 1:3 title "Solution Anaytique" with lines linewidth a
unset output

set output "images/exemple2.pdf"
plot "../plot/exemple2.dat" using 1:3 title "Rsulution numrique" with lines linewidth a,"../plot/exemple2.dat" using 1:4 title "Solution analytique" with lines linewidth a
unset output

set xrange[0:10]

set xlabel "Temps (secondes)"
set ylabel "Tension (Volts)"

set output "images/CAechelon.pdf"
plot "../plot/CAechelon.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CAechelon.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CAporte.pdf"
plot "../plot/CAporte.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CAporte.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CAcarre.pdf"
plot "../plot/CAcarre.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CAcarre.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CAtriangle.pdf"
plot "../plot/CAtriangle.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CAtriangle.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output


set output "images/CBechelon.pdf"
plot "../plot/CBechelon.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CBechelon.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CBporte.pdf"
plot "../plot/CBporte.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CBporte.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CBcarre.pdf"
plot "../plot/CBcarre.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CBcarre.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CBtriangle.pdf"
plot "../plot/CBtriangle.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CBtriangle.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CCechelon.pdf"
plot "../plot/CCechelon.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CCechelon.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CCporte.pdf"
plot "../plot/CCporte.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CCporte.dat" using 1:3 title "Vs(t)" with lines linewidth a
unset output

set output "images/CCcarre.pdf"
```

```
66 plot "../plot/CCcarre.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CCcarre.dat" using 1:3 title
   "Vs(t)" with lines linewidth a
   unset output
68
   set output "images/CCtriangle.pdf"
70 plot "../plot/CCtriangle.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CCtriangle.dat" using 1:3
   title "Vs(t)" with lines linewidth a
   unset output
72

74 set output "images/CDechelon.pdf"
   plot "../plot/CDechelon.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CDechelon.dat" using 1:3
   title "Vs(t)" with lines linewidth a
76 unset output

78 set output "images/CDporte.pdf"
   plot "../plot/CDporte.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CDporte.dat" using 1:3 title
   "Vs(t)" with lines linewidth a
80 unset output

82 set output "images/CDcarre.pdf"
   plot "../plot/CDcarre.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CDcarre.dat" using 1:3 title
   "Vs(t)" with lines linewidth a
84 unset output

86 set output "images/CDtriangle.pdf"
   plot "../plot/CDtriangle.dat" using 1:2 title "E(t)" with lines linewidth a,"../plot/CDtriangle.dat" using 1:3
   title "Vs(t)" with lines linewidth a
88 unset output
```