

MA411-TP Linéarisation d'une diode

DESGRANGES Jean-Philippe DAVID Rajendra
SCHAPIRA Boris

le 17 Octobre 2005

Introduction

Le but de ce TP est d'implémenter sur Matlab un algorithme de linéarisation d'une diode.

Le courant de sortie de la diode est donné par :

$$I_d = I_s \exp\left(\frac{V_d}{V_t}\right)$$

avec $I_s = 1e^{-6}A$ et $V_t = 26mV$

Tracé de la courbe $I_d = f(V_d)$:

```
function [out] = diode( Vd );
global Is Vt;
Vt = 26e-3;
Is = 1e-6 ;
out = Is * exp( Vd / Vt );
```

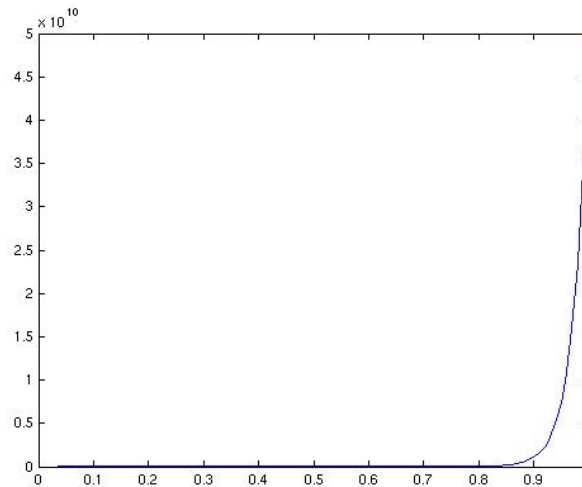


Figure 1: $I_d = f(V_d)$

Linéarisation à pas fixe

On calcule la simulation avec un signal discret V_d de type rampe : $V=[0 :0.01 :1]$;
ou de type sinusoïdal $W=\cos(V)$;

Le principe : on part d'une valeur d'origine V_0 et d'un pas fixe (ici, $V_0=0$ et pas=0.1).

On calcule la tangente au point $(V_0, I_d(V_0))$ puis on calcule la différence entre cette tangente et la courbe I_d à l'abscisse $V_0 + \text{pas}$ (différence qu'on appelle erreur).

Si cette différence est suffisamment faible, on avance d'un pas sans enregistrer la valeur $V_0 + \text{pas}$ précédente.

Lorsqu'on atteint la dernière valeur telle que cette différence soit suffisamment faible sans avancer d'un nouveau pas, on archive cette valeur dans un vecteur (ici u).

Ce vecteur contient l'ensemble des abscisses des points nécessaires à une linéarisation "correcte" de la courbe (à l'erreur tolérée).

Voici le code Matlab correspondant :

```
V=[0:0.01:1];
W = diode(V)
Vo = 0;
u = [0];
pas = 0.1;
err = 5e8;
while(Vo<=1)
    pente = diode(Vo) + (Vo+pas - Vo)*diode(Vo)/Vt;
    Vo = Vo+pas;
    while(abs( pente - diode(Vo)) <err \&\& Vo<=1)
        Vo = Vo+pas;
    end;
    u = [u Vo];
end;
J = diode(u);
plot(V,W)
axis([0 1 0 5e10])
```

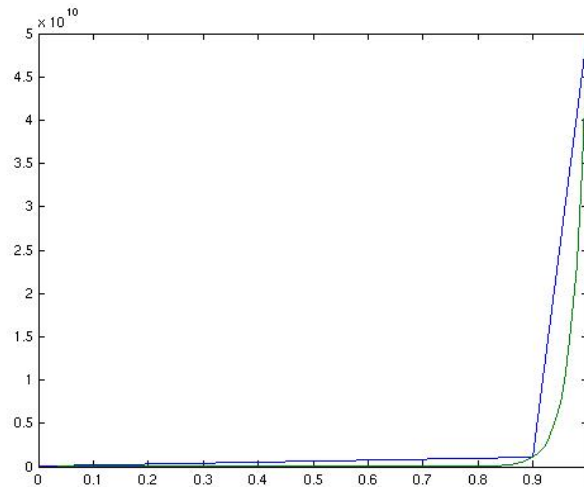


Figure 2: Linéarisation à pas fixe d'une diode avec en entrée un signal rampe

Linéarisation à pas variable

On commence la simulation avec un signal discret V_d de type rampe : $V=[0 :0.01 :1]$; ou de type sinusoïdal $W=\cos(V)$;

Le principe : on part d'une valeur d'origine V_0 et d'un pas fixé (ici, $V_0=0$ et $\text{pas}=0.5$).

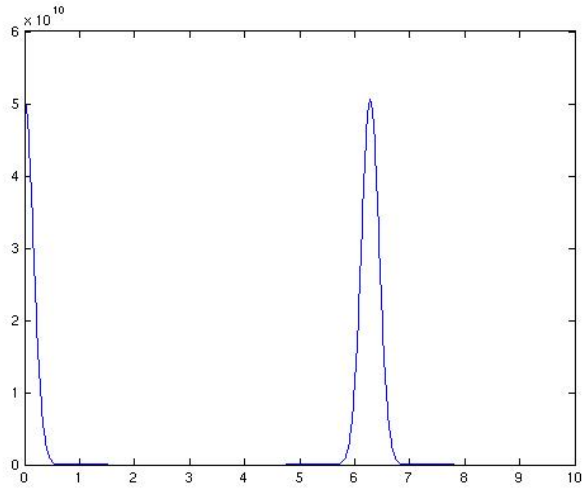


Figure 3: Signal de sortie d'une diode avec en entrée un signal sinusoïdal

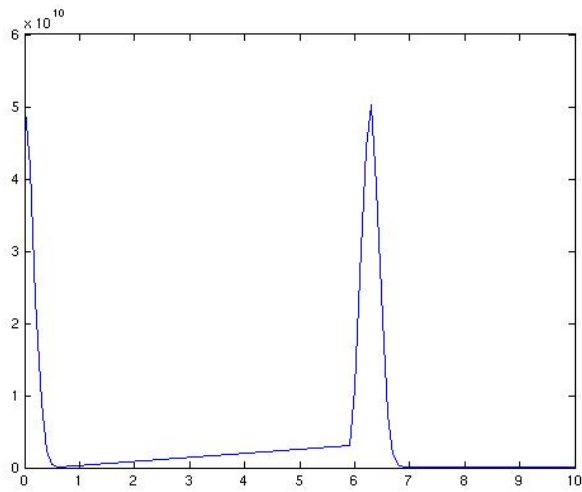


Figure 4: Linéarisation à pas fixe d'une diode avec en entrée un signal sinusoïdal

On calcule la tangente au point $(V_o, I_d(V_o))$ puis on calcule la différence entre cette tangente et la courbe I_d à l'abscisse $V_o + (\text{pas fixe})$ (différence qu'on appelle erreur).

Si cette différence est suffisamment faible, on avance d'un pas en enregistrant la valeur où l'on se trouve dans le vecteur (u) . Sinon, on divise le pas précédent par deux de manière à se retrouver plus près du V_o d'origine afin de minimiser l'écart, que l'on recalcule puis on réitère...

A la fin, le vecteur u contient l'ensemble des abscisses des points nécessaires à une linéarisation "correcte" de la courbe (à l'erreur tolérée). Dans notre code, nous avons également un vecteur p chargé d'enregistrer les différents pas successifs pour vérifier leur variation.

Voici le code Matlab correspondant :

```
Vo = 0;
```

```

u = [Vo];
pas = 0.5;
p = [pas];
err = 1e+13;
pastemp = pas;
i=0;
while(Vo<=10)
    pastemp = pas;
    pente = diode(cos(Vo)) + (Vo+pastemp - Vo)*diode(cos(Vo))/Vt;
    Vo = Vo+pastemp;
    diff = abs( pente - diode(cos(Vo)));
    while(diff>err && i<100000)
        i=i+1;
        pastemp = pastemp/2;
        Vo = Vo - pastemp;
        pente = diode(cos(Vo)) + (Vo+pastemp - Vo)*diode(cos(Vo))/Vt;
        diff = abs( pente - diode(cos(Vo)));
    end;
    u = [u Vo];
    p = [p pastemp];
end;

V=[0:0.01:10];
W = diode(cos(V))

J = diode(cos(u));
plot(V,W)
axis([0 10 0 6e10])

```

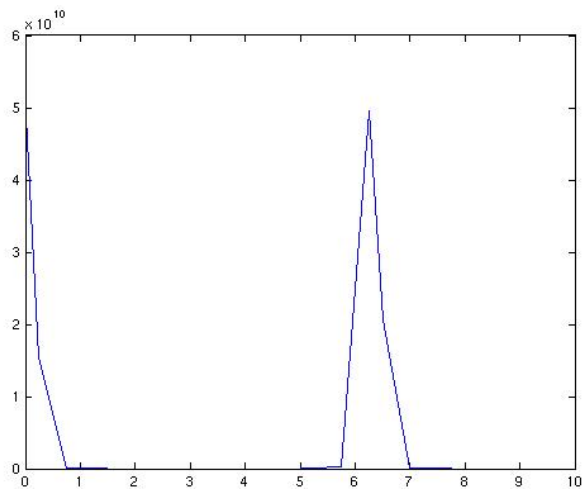


Figure 5: Linéarisation à pas variable d'une diode avec en entrée un signal sinusoïdale pour une erreur de $1 * 10^{12}$

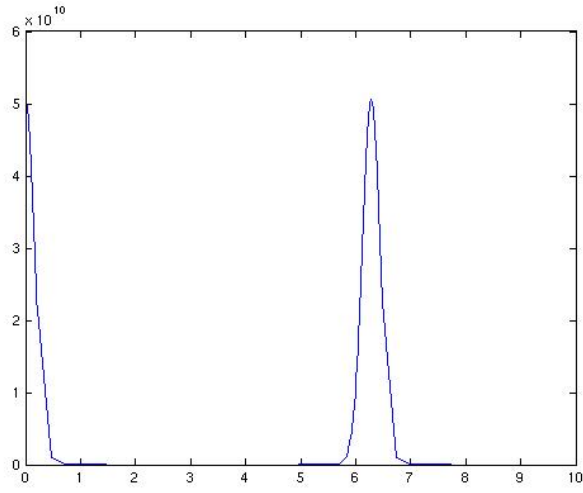


Figure 6: Linéarisation à pas variable d'une diode avec en entrée un signal sinusoïdale pour une erreur de $1 * 10^{10}$

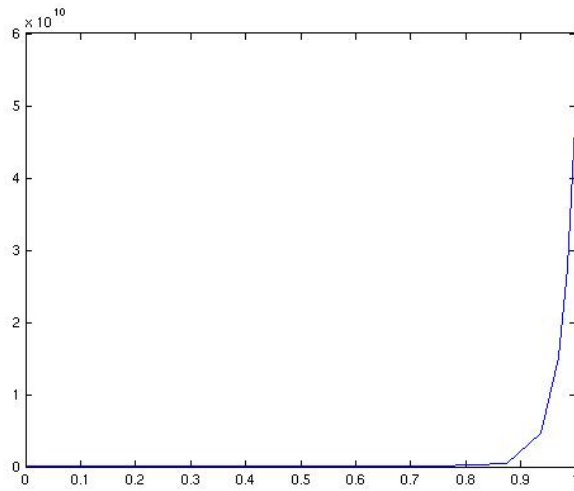


Figure 7: Linéarisation à pas variable d'une diode avec en entrée un signal rampe pour une erreur de $5 * 10^{10}$

Conclusion

Suivant la méthode utilisée (pas fixe ou pas variable), les erreurs à implémenter n'ont rien de commun à linéarisation égale et se doivent d'être étalonnées par l'expérience.

Le nombre de points obtenu est lui aussi difficile à prévoir.

Il pourrait être intéressant d'encadrer nos fonctions Matlab par des optimisations de la valeur de l'erreur. Elles dépendraient de la différence entre la fonction linéarisée et la fonction initiale suivant un pourcentage d'erreur à minimiser.