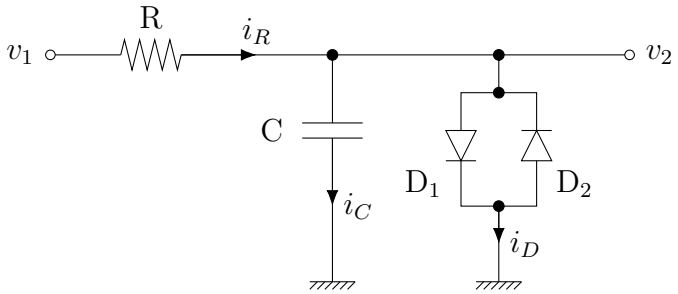
Introduction aux filtres numériques à topologie préservée

Laurent de Soras
http://ldesoras.free.fr

2020-04-23

Résumé

Cet article technique dans le domaine du traitement numérique du signal traite des procédés de filtrage sonore. Il aborde en particulier une certaine classe de filtres qui ont pour particularité de préserver la topologie de leur modèle analogique. Leurs propriétés les rendent intéressants entre autres pour imiter des filtres électroniques analogiques ou encore pour créer des filtres originaux.



Sommaire

1 Introduction

La réalisation d'un synthétiseur ou d'un processeur d'effets numérique demande souvent de copier des filtres analogiques électroniques existants possédant les qualités recherchées. Le terme filtre peut s'entendre au sens large, puisque par exemple un effet de distorsion peut aussi être vu comme un filtre sollicitant fortement ses composantes non-linéaires. En réalité, la grande majorité des circuits comportent des éléments à mémoire (capacitifs, inductifs...) — ne serait-ce que par l'imperfection des composants — et peuvent être considérés comme des filtres au sens de cet article.

Au-delà de la simulation de circuits électroniques existants, il peut être intéressant d'utiliser des techniques de filtrage numérique pour créer des effets originaux aux sonorités variées.

Habituellement, on réalise un filtre en passant par un prototype analogique en temps continu, en le décomposant en cellules du premier et du second ordre, et finalement en convertissant en temps discret (échantillonné) chacune de ces cellules à l'aide d'une transformation spécifique, généralement une transformée bilinéaire (*bilinear transform*, ou BLT). Il en résulte une brique de base, par exemple un filtre passe-bas du premier ordre, qu'on peut intégrer au sein d'un ensemble plus complexe. Le succès du *RBJ cookbook* [3], sorte de formulaire à composer des filtres, laisse à penser que cette pratique est très largement répandue dans le développement informatique audio-numérique.

Or cette approche part sur d'importants présupposés, qui sont finalement assez éloignés de la réalité que l'on cherche à copier : les filtres sont linéaires et temporellement invariants. Ces deux aspects sont très importants. Tout d'abord parce que les non-linéarités du traitement analogique participent entièrement à la chaleur et aux caractéristiques dynamiques du filtre. Il est d'ailleurs impossible de réaliser un filtre auto-oscillant à la fois stable et purement linéaire. Ensuite, les filtres sont rarement figés. Leurs caractéristiques bougent en fonction des réglages d'utilisation ou sont dynamiques par nature, selon le fonctionnement de l'appareil. Les implémentations discrètes classiques (formes directes, transposées, etc.) sont généralement inadaptées aux changements brusques des paramètres, qui aboutissent à des craquements ou autres bruits gênants.

Un des principaux écueils de l'approche par briques en BLT réside dans la gestion des contre-réactions. Comme le circuit est envisagé sous forme d'un graphe dans lequel « coule » le signal, les contre-réactions nécessitent l'introduction d'un échantillon de retard intercalaire. En effet, on a besoin de connaître la valeur de la sortie pour pouvoir la réinjecter en amont au cycle suivant. Il est globalement acquis que ce retard est un facteur majeur de défaut dans la modélisation ; la

contre-réaction devrait être immédiate. De nombreuses tentatives [2, 5] ont été faites pour contourner ces problèmes, mais les solutions existantes sont rarement satisfaisantes.

Depuis un certain nombre d'années, d'autres méthodes ont été introduites et peuvent être regroupées sous le nom de TPT (*topologicaly-preserving transform*, ou transformée préservant la topologie)[10]. Sans être forcément nouvelles, elles ont été adoptées grâce à une formalisation adéquate. D'autre part, la puissance de calcul des processeurs est devenue suffisante pour absorber le surcoût qu'elles peuvent entraîner par rapport aux bidouillages basés sur les méthodes traditionnelles.

Ces techniques sont diverses mais restent très similaires dans leurs principes. Elles prennent dès le départ une direction assez différente des méthodes traditionnelles et se rapprochent plutôt des algorithmes utilisés dans les logiciels simulateurs de circuits. Nous esquisserons deux de ces approches dans les paragraphes qui viennent.

Un survol rapide de l'article montre un nombre considérable de formules qui peuvent paraître compliquées. Précisons que le niveau en mathématiques requis n'est pas très élevé, disons à peu près du niveau fin de lycée. La grande majorité des formules font uniquement appel à des opérations simples. Il est préférable d'avoir déjà des notions de traitement numérique du signal, mais ce n'est pas absolument indispensable. Des connaissances de base en électronique (loi d'Ohm, loi des nœuds...) sont conseillées. En revanche, les abstractions mises en œuvre peuvent être plus rebutantes.

2 Discrétisation d'un filtre

Les condensateurs et les bobines sont des composants à la base de la plupart des filtres électroniques. On peut qualifier ces composants, ainsi que de nombreux autres, d'être « à stockage d'énergie ». Ils ont la particularité d'avoir des caractéristiques qui lient la tension et le courant par l'entremise d'une dérivée. Ainsi le courant qui traverse un condensateur est proportionnel à la dérivée de la tension à ses bornes, et c'est l'inverse pour une bobine.

Par conséquent, l'établissement de formules mathématiques régissant les circuits comportant ce genre de composant se fait sous forme d'équations différentielles, par rapport au temps. La complexité de ces équations croît vite avec le nombre de composants et, à part dans certains cas très spécifiques, il est impossible d'obtenir une formulation analytique exprimant les courants et tensions dans le circuit. Le filtrage consiste donc à simuler le circuit en résolvant les équations différentielles au pas par pas. Nous verrons comment modéliser de tels circuits puis comment résoudre les équations correspondantes par intégration numérique.

Tout d'abord, appelons F_S la fréquence d'échantillonnage du système, en Hz. À cette fréquence correspond une période T définie par :

$$T = \frac{1}{F_S} \tag{2.1}$$

T est donc le temps en secondes séparant deux échantillons successifs. Nous présumerons que ce temps reste constant.

2.1 La méthode traditionnelle

La méthode traditionnelle de création d'un filtre numérique consiste à partir de l'équation d'un filtre en temps continu [1]. Quand le filtre est une modélisation d'un circuit électronique, on utilise les impédances complexes pour exprimer sa fonction de transfert dans le domaine de Laplace (variable en s). Ensuite on utilise une formule d'intégration numérique pour passer de l'équation en temps continu fonction de s à une équation en temps discret fonction de z^{-1}. En effet l'opérateur $1/s$ correspond à une intégration qu'il faut discrétiser. La transformée bilinéaire (BLT) est le choix d'intégration le plus courant, effectuant la substitution suivante :

$$s \to \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{2.2}$$

T est la période d'échantillonnage définie plus haut. On obtient une fonction de transfert exprimée à partir d'unités z^{-1}, unités qui représentent le retard du signal

d'un échantillon. Cette fonction de transfert peut être convertie à son tour en une équation donnant la sortie du filtre comme combinaison linéaire des échantillons d'entrée et des précédents échantillons d'entrée et de sortie. L'équation obtenue représente l'étape itérative d'un algorithme qui peut être réalisé sous diverses formes (directes I et II, leurs transposées et d'autres encore).

Il est à noter que la transformée bilinéaire applique au passage une déformation de l'axe des pulsations, que l'on corrige au moyen d'un *prewarping* adéquat.

Malheureusement la BLT fait complètement disparaître la topologie initiale du filtre modélisé en intercalant une fonction de transfert abstraite en s, qui pourrait correspondre à de nombreuses topologies différentes. En outre, les opérations non-linéaires n'y sont pas représentables. Les signaux manipulés au sein de la réalisation d'une BLT n'ont finalement plus rien à voir avec le circuit dont ils sont dérivés, quelle que soit la forme de réalisation choisie. C'est une méthode qui peut être pratique pour réaliser des filtres purement statiques et linéaires, mais qui est inadaptée aux filtres dits « musicaux ».

2.2 Transformation préservant la topologie

L'idée de la TPT est de discrétiser non pas l'ensemble du filtre, mais uniquement l'opérateur d'intégration.

Une fois cet opérateur discrétisé, les différents éléments constitutifs peuvent être reliés entre eux par des équations simples. Ces équations peuvent ensuite être transformées de façon à faire apparaître des étapes algorithmiques. Le gros intérêt de la méthode, c'est que la topologie du filtre initial est préservée dans le filtre discret. Le signal suit le même chemin que dans le filtre analogique, bien que l'algorithme final ne le reflète pas forcément.

2.3 Intégration numérique

Avant de rentrer dans le vif du sujet, il nous faut aborder les techniques d'intégration numérique. Nous essaierons d'être concis et d'aller droit au but, veuillez nous excuser si le formalisme mathématique est parfois un peu écorné au passage.

Pour dire les choses très simplement, l'intégration peut être vue comme le calcul algébrique de l'aire située entre la courbe d'une fonction et l'axe du temps, entre deux instants. Cette définition s'applique en temps continu. En temps discret, il ne peut s'agir que d'une approximation. En effet, nous ne savons pas quelle courbe précise suit la fonction à intégrer entre deux échantillons.

Comme mentionné précédemment, la discrétisation du temps est constante et

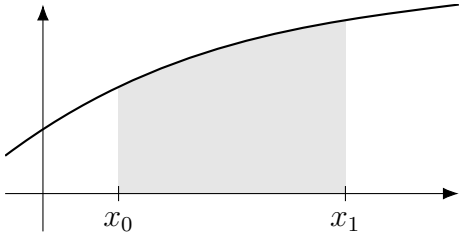


FIGURE 2.1 – *Intégration d'une fonction entre x_0 et x_1. La surface grisée en est le résultat. Quand la courbe passe sous l'axe des abscisses, la surface correspondante est à déduire.*

la période d'échantillonnage vaut T.

2.3.1 Les méthodes d'intégration

Une méthode d'intégration est une formule pour approximer en temps discret une intégration qui se fait normalement en temps continu. On peut aussi le voir dans l'autres sens, comme étant une approximation de la dérivée du signal que l'on souhaite obtenir. C'est d'ailleurs à partir de cette dernière formulation que ces méthodes sont établies.

Il existe deux principaux types de méthodes. Les méthodes explicites n'ont besoin que de la formule d'intégration pour bâtir l'instant $n + 1$. Autrement dit elles se basent uniquement sur les échantillons du passé jusqu'à n pour construire le résultat suivant. Pour cette raison, elles sont relativement peu coûteuses. En revanche, elles sont facilement instables et sont donc rarement utilisées.

Les méthodes implicites se basent en plus sur la valeur de la dérivée à l'instant $n + 1$, ce qui nécessite de résoudre d'autres équations pour connaître cette valeur. Bien que plus complexes, ces méthodes sont plus stables et sont les plus utilisées.

Pour l'implémentation de nos filtres, nous n'utiliserons pas de méthode utilisant des valeurs placées temporellement à mi-chemin des échantillons sources, comme celles du point milieu ou de Runge-Kutta. Cela reste cependant possible.

Voici trois méthodes élémentaires dans un formalisme très simplifié, illustrées sur la figure 2.2. x est suite échantillonnée à partir de la fonction que l'on cherche à intégrer, et y le résultat de son intégration.

Euler explicite
$$y[n] = y[n-1] + T \cdot x[n-1] \tag{2.3}$$

Euler implicite
$$y[n] = y[n-1] + T \cdot x[n] \tag{2.4}$$

Formule du trapèze, implicite
$$y[n] = y[n-1] + \frac{T}{2}\Big(x[n-1] + x[n]\Big) \tag{2.5}$$

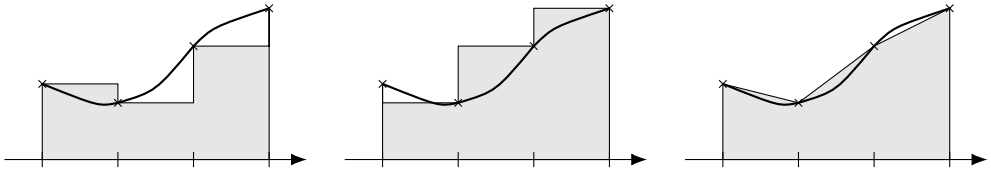


FIGURE 2.2 – *Méthodes d'intégration. De gauche à droite : Euler explicite, Euler implicite, trapèze. Les croix représentent les valeurs discrètes échantillonnées. La surface colorée correspond au résultat de l'intégration.*

Toutes ces formules sont récursive. Il faut connaître le résultat précédent pour calculer le suivant.

On voit bien dans les formules implicites que $y[n]$ dépend de $x[n]$. Cependant dans les cas qui nous intéressent — la résolution d'équations différentielles — ce dernier est souvent lui-même dépendant de $y[n]$ via d'autres formules. Cela nécessitera à chaque période de résoudre des équations préliminaires avant de pouvoir effectuer l'étape d'intégration proprement dite.

L'approximation trapézoïdale forme un bon compromis générique entre précision et complexité. Il se trouve que c'est la même qui sert à la transformée bilinéaire. Nous adopterons cette approximation tout au long de cet article, mais il est évidemment possible de changer de méthode quand le cas s'y prête ou le nécessite.

Si toutes ces considérations peuvent paraître un peu abstraites, ce n'est pas grave, nous illustrerons cela dans les parties qui suivent.

2.4 Un exemple : filtre passe-bas du 1$^{\text{er}}$ ordre

Commençons par l'analyse et la simulation d'un montage simple : un filtre RC passif, dont le schéma est dessiné sur la figure 2.3.

v_1 représente la tension du signal d'entrée. v_2 est la tension de sortie du filtre. Il s'agit d'un simple pont diviseur entre le condensateur et la résistance. Le courant

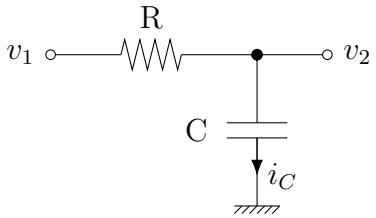


FIGURE 2.3 – *Un filtre passe-bas*

traversant un condensateur est lié à la tension à ses bornes par la formule suivante :

$$i_C(t) = C\frac{dv_C}{dt} \tag{2.6}$$

Si on isole dv_C/dt :

$$\frac{dv_C}{dt} = \frac{i_C(t)}{C} \tag{2.7}$$

En substituant $i_C(t)/C$ à x et v_C à y dans la formule trapézoïdale d'intégration (2.5), on obtient :

$$i_C[n] = \frac{2C}{T}v_C[n] - \frac{2C}{T}v_C[n-1] - i_C[n-1] \tag{2.8}$$

Cette formule peut être réécrite sous la forme d'une fonction linéaire :

$$i_C[n] = g_{Ceq} \cdot v_C[n] + i_{Ceq}[n-1] \tag{2.9}$$

avec :

$$g_{Ceq} = \frac{2C}{T} \tag{2.10}$$

$$i_{Ceq}[n] = -\frac{2C}{T}v_C[n] - i_C[n] \tag{2.11}$$

À un instant donné, g_{Ceq} et i_{Ceq} sont déjà connus et figés. Ils servent de base pour construire des équations uniquement formées de termes linéaires. Pour l'instant nous n'avons qu'un seul terme différentiel, mais des circuits plus complexes en auront plusieurs.

Toujours à un instant donné, le composant peut être représenté par un générateur équivalent Norton (figure 2.4) :

Attention au sens des broches et à l'orientation des grandeurs algébriques. La broche ① est du côté « positif ». Autrement dit, la tension associée au dipôle est la

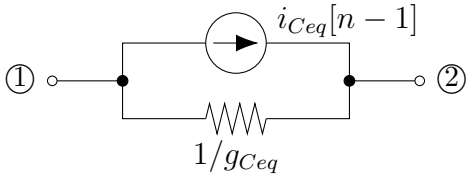


FIGURE 2.4 – *Équivalent Norton du condensateur à un instant n*

différence de potentiels ①−②, et le courant circule conventionnellement de ① vers ②. D'autre part g_{Ceq} étant homogène à une conductance, son inverse représente une résistance.

Notre équivalent va maintenant remplacer le condensateur dans le circuit dont nous écrivons l'équation :

$$v_1 = Ri_C + v_2 \tag{2.12}$$

En substituant i_C par la formule (2.9) :

$$v_1[n] = R(g_{Ceq} \cdot v_2[n] + i_{Ceq}[n-1]) + v_2[n] \tag{2.13}$$

Ce qui se résout en :

$$v_2[n] = \frac{v_1[n] - Ri_{Ceq}[n-1]}{Rg_{Ceq} + 1} \tag{2.14}$$

Une fois $v_2[n]$ connu, on peut effectuer l'itération d'intégration avec (2.9), v_2 remplaçant v_C. Et c'est reparti pour un tour.

2.5 Réalisation

En appliquant directement (2.14), (2.9), (2.10) et (2.11), on peut dériver les lignes de pseudo-code suivantes pour chaque itération temporelle :

```
iceq := -gceq * y - ic
y    := (x - R * iceq) / (1 + R * gceq)
ic   := gceq * y + iceq
```

Avec :

```
gceq := 2 * C * fs
```

Ce code peut cependant être optimisé et simplifié. `ic` représente l'état de l'intégrateur, qui doit d'ailleurs être initialisé à `0`. Cependant `ic` n'intervient pas directement dans le calcul de `y`. On peut commencer par remplacer `ic` par `iceq`

pour mémoriser l'état de l'intégrateur d'une itération à l'autre. Cela se fait simplement en redescendant en bas la ligne qui calcule `iceq`, désormais à l'ordre $n+1$. `ic` n'est plus qu'une variable temporaire qui peut être éliminée.

```
gceq  := 2 * C * fs
y     := (x - R * iceq) / (1 + R * gceq)
iceq  := -2 * gceq * y - iceq
```

Ensuite on pose $g_0 = R \cdot g_{Ceq}$, ce qui permet de retrouver le fameux produit RC dans le paramétrage du filtre et de sortir `R` de l'itération. Implicitement, `iceq` se retrouve aussi multiplié par R :

```
g0    := 2 * C * R * fs
y     := (x - iceq) / (1 + g0)
iceq  := -2 * g0 * y - iceq
```

On peut diviser `iceq` par `-g0`. On adapte `y` et le développe :

```
g0    := 2 * C * R * fs
y     := x * 1 / (1 + g0) + iceq * g0 / (1 + g0)
iceq  := 2 * y - iceq
```

La somme des deux facteurs en face de `x` et `iceq` font 1, ce qui fait apparaître `y` comme interpolation linéaire de `iceq` et de `x`. Celle-ci peut être reformulée pour éliminer une multiplication.

```
y     := iceq + (x - iceq) * g1
iceq  := 2 * y - iceq
```

Avec :

```
g0    := 2 * C * R * fs
g1    := 1 / (1 + g0)
```

Il est également possible de décliner l'itération de cette manière :

```
v     := (x - iceq) * g1
y     := iceq + v
iceq  := y + v
```

Il n'y maintenant plus qu'une multiplication et trois additions. Précisons que la fréquence de coupure en Hz à $-3\,\mathrm{dB}$ est données par $1/(2\pi RC)$. Comme nous avons utilisé une intégration trapézoïdale, il nous faut utiliser un *prewarping* pour garder cette fréquence juste. D'autre part, pour les mêmes raisons, la forme de la réponse sera compressée contre la fréquence de Nyquist. Cette situation est identique à celle du filtre équivalent en BLT. Ces inconvénients peuvent être palliés par suréchantillonnage ou par transformation progressive en un filtre-plateau.

3 Technique ZDF

Nous allons maintenant montrer une autre manière d'arriver au même résultat, en passant par une autre abstraction. Cette technique, appelée ZDF (pour *zero delay feedback*, ou contre-réaction instantanée) permet par ailleurs de créer plus intuitivement des filtres originaux qui ne correspondent pas à la modélisation d'un circuit précis [10].

On peut poser les relations entre les différentes tensions du circuit de la figure 2.3 :

$$v_1 = Ri_C + v_2 \tag{3.1}$$

$$i_C = C\frac{dv_2}{dt} \tag{3.2}$$

L'indexation en t a été supprimée pour simplifier l'écriture. Par substitution, on obtient :

$$v_1 = RC\frac{dv_2}{dt} + v_2 \tag{3.3}$$

Nous avons donc notre équation différentielle. On peut regrouper v_1 et v_2 puis intégrer tous les termes fonctions du temps :

$$v_2(t) = v_2(0) + \int_{\tau=0}^{t} \frac{1}{RC}\Big(v_1(\tau) - v_2(\tau)\Big)d\tau \tag{3.4}$$

Il n'est pas anodin d'avoir laissé $1/RC$ dans l'intégrale, plutôt que de l'avoir factorisé devant. En effet, ce terme affectant la fréquence de coupure du filtre n'est pas forcément constant. En le sortant de l'intégrale, on fausserait la formule si celui-ci venait à varier en fonction de t. On peut d'ailleurs lui assigner la pulsation de coupure ω_0 :

$$\omega_0 = \frac{1}{RC} \tag{3.5}$$

Cette équation peut être représentée sous forme de diagramme, comme l'illustre la figure 3.1.

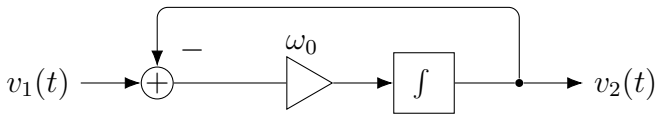


FIGURE 3.1 – *filtre passe-bas sous forme de diagramme*

À titre de vérification il est possible de retrouver la fonction de transfert équivalente en s. Cela se fait en remplaçant le bloc d'intégration par son équivalent

$1/s$ en termes de transformée de Laplace. Ainsi, en repartant du diagramme on obtient facilement l'équation en s :

$$v_2(s) = \frac{\omega_0}{s}\Big(v_1(s) - v_2(s)\Big) \tag{3.6}$$

Puis en posant

$$H(s) = \frac{v_2(s)}{v_1(s)} \tag{3.7}$$

on remanie l'équation précédente pour obtenir :

$$H(s) = \frac{1}{1 + \dfrac{s}{\omega_0}} \tag{3.8}$$

On reconnaît la fonction de transfert d'un filtre passe-bas du premier ordre, ayant ω_0 pour pulsation de coupure à $-3\,\mathrm{dB}$. Nous n'utiliserons cependant pas cette équation.

3.1 Intégrateur trapézoïdal

Nous avons défini l'intégrateur trapézoïdal dans la formule (2.5). La figure 3.2 le représente sous forme de diagramme, avec un T unitaire.

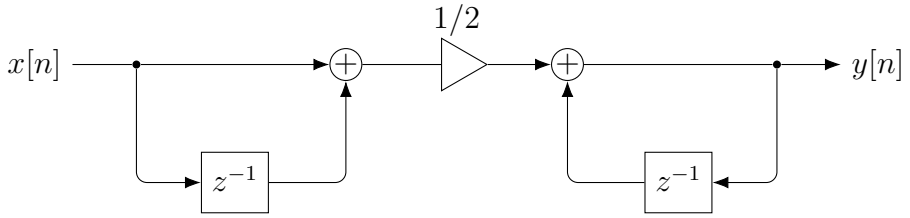


FIGURE 3.2 – *Intégrateur trapézoïdal, forme directe I*

Il s'agit de la forme directe I. Nous utiliserons plutôt la forme II transposée, qu'on peut voir sur la figure 3.3.

Cette forme a la particularité de n'avoir qu'une seule mémoire, et une seule combinaison par addition de cette mémoire avec le signal principal. Ces caractéristiques nous simplifieront les choses par la suite. Le multiplicateur étant placé avant les autres opérations, nous le factoriserons plus tard avec la pulsation de coupure ω_0.

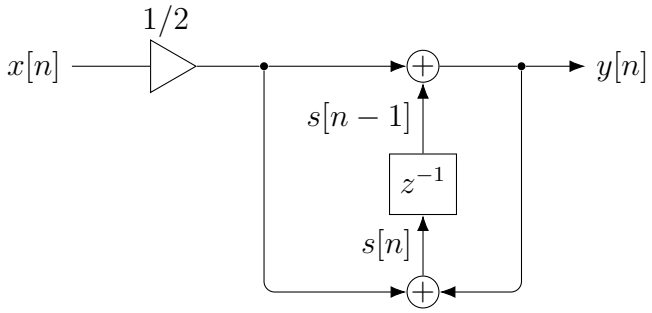


FIGURE 3.3 – *Intégrateur trapézoïdal, forme directe II transposée*

On peut poser :

$$g = \frac{\omega_0 T}{2} \tag{3.9}$$

Si on attribue la lettre s (à ne pas confondre avec la variable de Laplace qui n'a pas sa place ici) à l'état courant de la mémoire, on obtient l'équation suivante :

$$y = gx + s \tag{3.10}$$

Cet intégrateur est donc une fonction linéaire de son entrée x. g et s sont des variables figées « de l'extérieur » au moment où l'on cherche à calculer la sortie, on peut donc les considérée comme constantes pendant une itération de calcul donnée. Cette forme sera particulièrement importante pour la suite, car elle nous permettra d'utiliser l'arsenal d'algèbre linéaire pour résoudre des assemblages potentiellement très complexes.

3.2 Réalisation

Si on place l'intégrateur trapézoïdal dans notre filtre passe-bas, on obtient la figure 3.4.

Pour rendre les choses plus génériques, nous avons remplacé v_1 et v_2 par x et y, respectivement. Comme on le voit sur le diagramme, la boucle de rétroaction principale n'a pas d'élément de retard z^{-1} sur son chemin. Pourtant, le filtre est bel et bien réalisable tel quel. Pour vérifier cela, transcrivons-le sous forme d'équations. Encore une fois, nous retirons les indices temporels pour simplifier les écritures.

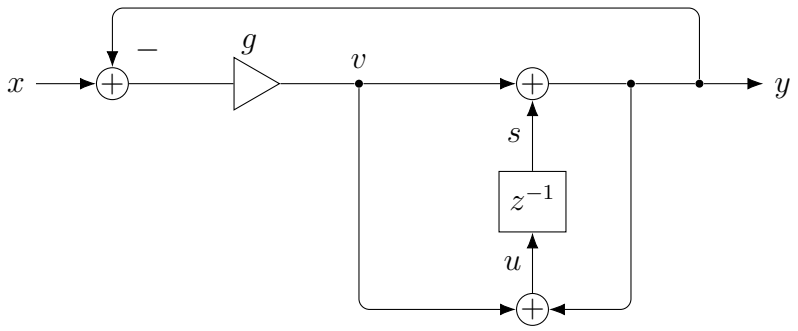


FIGURE 3.4 – *Discrétisation du filtre passe-bas 1 pôle*

$$y = s + v \tag{3.11}$$
$$v = g(x - y) \tag{3.12}$$
$$u = y + v \tag{3.13}$$

La dernière équation (3.13) sert à mettre à jour l'état de la mémoire de l'intégrateur, et u prendra la place de s à l'itération suivante. (3.13) ne nous servira donc pas directement à trouver y. En combinant les deux premières équations (3.11) et (3.12), on obtient :

$$y = \frac{gx + s}{g + 1} \tag{3.14}$$

Qu'on peut encore écrire :

$$y = x\frac{g}{g + 1} + \frac{s}{g + 1} \tag{3.15}$$

On retrouve de nouveau y exprimé comme une fonction linéaire de x !

Quand on modélise des composants, on va systématiquement chercher à les exprimer sous une forme linéaire $y = ax + b$. De cette manière, quand l'équation est insérée dans une formule plus large et que x contient à son tour y, par exemple sous la forme $y = a(x - y) + b$, il est possible de résoudre très facilement l'équation :

$$y = \frac{ax + b}{a + 1}$$

Celle-ci est à nouveau une fonction linéaire qui peut être utilisée dans un ensemble plus grand.

Cette approche est assez similaire à la notion de polarisation et de petits signaux quand on étudie les montages électroniques à transistors, tubes et autres éléments non-linéaires. La différence, c'est qu'ici, c'est l'état instantané du système qui est considéré comme polarisation, les petits signaux servent essentiellement à calculer l'état suivant.

On réinjecte le y qu'on a trouvé dans l'équation donnant v, on simplifie et on écrit l'itération qui réalise ce filtre sous forme de pseudo-code :

```
v  := (x - s) * gv
y  := s + v
s  := y + v
```

Avec :

```
w0 := 2 * fs * tan (pi * f0 / fs)
g  := w0 / (2 * fs)
gv := g / (g + 1)
```

Avant tout traitement, n'oublions pas d'initialiser l'état du filtre :

```
s  := 0
```

La première ligne de paramétrage, donnant ω_0 en fonction de la fréquence de coupure f_0 et la fréquence d'échantillonnage F_S, utilise un *prewarping* standard. On peut bien sûr en utiliser d'autres en fonction des besoins.

3.3 Considérations

Notons la similitude entre le morceau de code que nous venons d'obtenir et celui auquel nous avons abouti dans la première partie. En passant par une méthode en apparence très différente, nous avons retrouvé un résultat quasiment identique. Ce n'est pas un hasard. Le filtre analogique dont nous somme parti est le même, la méthode d'intégration est la même, etc.

Il est également intéressant de noter qu'on peut retranscrire la réalisation finale

de ce filtre sous forme de diagramme. Celui-ci, sans surprise, ne comprendra aucune boucle de rétroaction sans élément de retard. Soit dit en passant, la traduction du filtre sous forme de diagramme n'est pas nécessaire. On peut discrétiser le filtre uniquement avec les équations. Mais la visualisation en diagramme permet de clarifier certaines choses.

On peut également noter que la sortie `y` est située à mi-chemin entre deux états consécutifs `s` (ou `iceq`). C'est normal : c'est l'opération de moyenne de notre intégrateur qui se retrouve formulée dans ces termes.

Parmi les différentes formes de résolution des équations, nous en avons choisie une particulière. Elle consiste à trouver v (dans la méthode précédente) qui est le différentiel entre l'état précédent, y et l'état suivant. Cette forme a pour intérêt d'être relativement peu vulnérable au bruit de calcul.

Nous avons à présent terminé la modélisation de notre filtre passe-bas d'ordre 1. Pour résumer, à partir d'un filtre donné par son montage électronique ou même par une équation de Laplace, on peut obtenir une réalisation stable en passant par l'opérateur d'intégration trapézoïdal. De telles réalisations sont bien plus résistantes au bruit de calcul (imprécisions liés à la quantification des valeurs) que leurs équivalents BLT [6]. De plus, elles se comportent de manière plus fluides lors des changements de paramètres comme la fréquence de coupure. Mentionnons aussi que ces filtres, dans leur version linéaire, ont un surcoût négligeable par rapport à leur équivalent BLT. Ils apportent une amélioration nette, bien que leur conception soit un peu plus complexe.

Nous n'avons pas encore vu comment insérer de la distorsion au cœur du filtre. Ce sera l'objectif de la partie suivante.

4 Ajout de composants non-linéaires

Une manière redoutablement efficace d'introduire une non-linéarité dans le montage du filtre à 1 pôle de la figure 2.3 est de rajouter deux diodes tête-bêche entre la sortie et la masse. Cela nous donne le schéma de la figure 4.1. Assez intuitivement, on voit que ce montage écrête la sortie à des niveaux avoisinant la tension passante des diodes, en négatif comme en positif.

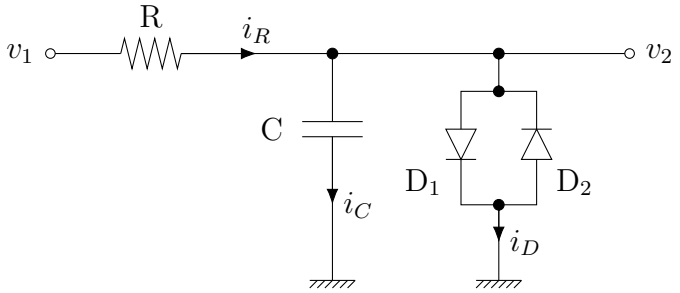


FIGURE 4.1 – *Un filtre passe-bas avec des diodes de saturation*

Nous nous contenterons ici d'un modèle de diode simple et statique, sans capacité parasite. Sa caractéristique est approximée par l'équation de Shockley :

$$i_D = I_S\left(e^{\frac{v_D}{nV_T}} - 1\right) \tag{4.1}$$

Où i_D est l'intensité du courant qui parcourt la diode et v_D la tension à ses bornes. Les autres paramètres sont particuliers au type de la diode : I_S est le courant de saturation en inverse, V_T la tension thermique qui vaut autour de 26 mV à température ambiante et n est son facteur de qualité.

On peut considérer les deux diodes tête bêche comme un seul dipôle, le courant le traversant étant la somme algébrique des courants dans les deux diodes. Appelons S la fonction caractéristique d'un tel dipôle :

$$\begin{aligned} S(v_D) &= I_{S1}\left(e^{\frac{+v_D}{n_1 V_T}} - 1\right) - I_{S2}\left(e^{\frac{-v_D}{n_2 V_T}} - 1\right) \\ &= i_D \end{aligned} \tag{4.2}$$

Cette fonction est un « antisaturateur » : elle a la tête d'une courbe de saturation, mais en miroir par rapport à l'axe $x = y$. Sa formule exacte n'est pas très importante, on peut utiliser une approximation à la place. Nous verrons plus tard que nous devrons l'évaluer plusieurs fois par échantillon de sortie, ainsi que sa dérivée. Les exponentielles étant coûteuses à calculer, il est judicieux d'essayer d'en réduire la quantité voire de les éliminer.

4.1 Technique ZDF avec diagrammes

Pour l'instant on applique la loi des nœuds de Kirchhoff :

$$i_R = i_C + i_D \tag{4.3}$$

On obtient alors :

$$v_1 - v_2 = R \cdot \left(C\frac{dv_2}{dt} + S(v_2) \right) \tag{4.4}$$

Ce qui ajoute un nouveau membre à notre équation initiale (3.4) :

$$v_2(t) = v_2(0) + \int_{\tau=0}^{t} \left(\frac{1}{RC}\Big(v_1(\tau) - v_2(\tau)\Big) - \frac{S\Big(v_2(\tau)\Big)}{C} \right) d\tau \tag{4.5}$$

4.1.1 Conception du diagramme

On peut transposer (4.5) en un diagramme visible sur la figure 4.2. Si on déplace le gain ω_0 juste devant l'intégrateur, qu'on adapte le gain de l'antisaturateur et qu'on regroupe les deux contre-réactions, on obtient le diagramme de la figure 4.3.

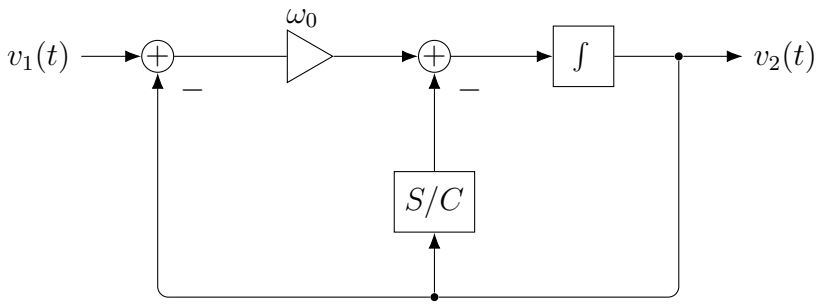


FIGURE 4.2 – *Filtre passe-bas avec saturation à diodes sous forme de diagramme*

En remplaçant le bloc intégrateur continu par l'intégration trapézoïdale, on obtient le filtre discrétisé de la figure 4.4. On peut maintenant le passer sous forme d'équations à résoudre.

$$y = s + v \tag{4.6}$$
$$p = y + R \cdot S(y) \tag{4.7}$$
$$v = g(x - p) \tag{4.8}$$
$$u = y + v \tag{4.9}$$

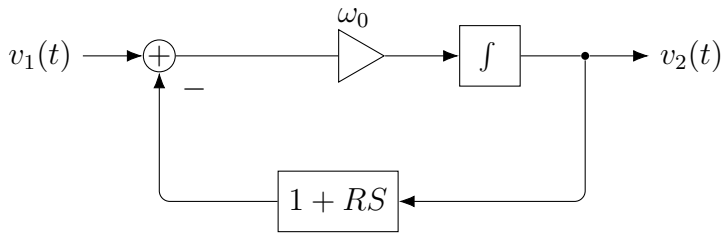


FIGURE 4.3 – *Filtre passe-bas avec saturation à diodes sous forme de diagramme, version factorisée*

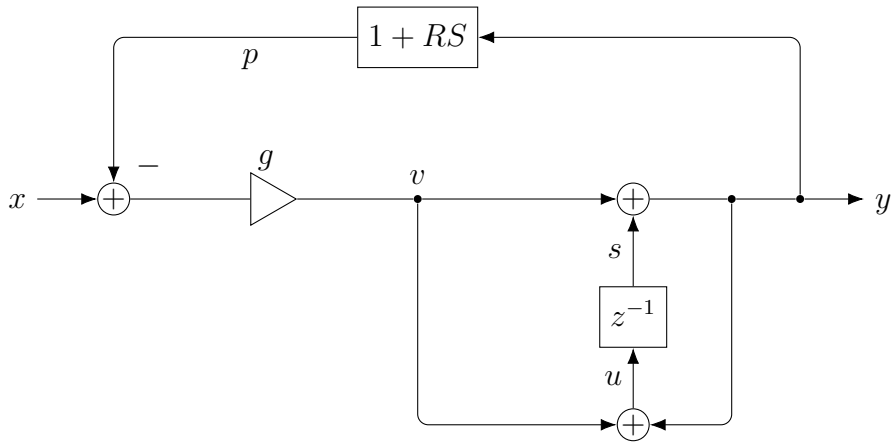


FIGURE 4.4 – *Discrétisation du filtre passe-bas 1 pôle avec saturation à diodes*

4.1.2 Résolution de l'équation non-linéaire

On cherche de nouveau à obtenir y. En combinant (4.6), (4.7) et (4.8), on trouve :

$$y = \frac{s + gx - gR \cdot S(y)}{g + 1} \tag{4.10}$$

Petit problème, y apparaît des deux côtés du signe d'égalité, sans qu'il ne semble possible de le sortir de S. Précisons tout de suite qu'il est a priori inutile de développer S dans l'espoir de trouver une simplification : on va retomber sur des formes où y reste un paramètre d'une fonction non-linéaire spécifique, sans pouvoir trouver de solution analytique pratique à cette équation.

Mais tout n'est pas perdu (heureusement !), car nous pouvons trouver une approximation de la solution à cette équation par l'analyse numérique. En fait, toutes les variables situées dans la boucle de y dans le diagramme sont mutuellement déductibles. Il nous faut chercher la variable qui nous donnera y et u de la manière

la plus directe possible. En l'occurence, y fait déjà très bien l'affaire. Pour trouver sa valeur, nous résoudrons numériquement $f(y) = 0$, fonction donnée par (4.11), déduite de (4.10).

$$f(y) = S(y) + \frac{1+g}{gR}y - \frac{s+gx}{gR} \tag{4.11}$$

La résolution de cette équation devra se faire lors de chaque itération temporelle. À ce moment, toutes les variables de l'équation sont figées, à part y qui reste notre inconnue. Une fois y trouvé, nous pourrons déduire v de (4.6) et mettre à jour l'état de l'intégrateur.

Note : nous présumons ici que $f(y) = 0$ a une et une seule solution. Ce résultat n'est pas trivial et dépend bien sûr de S, mais nous n'essayerons pas de le montrer ici.

Pour résoudre l'équation, diverses méthodes sont à notre disposition : Newton-Raphson, dichotomie, tables interpolées, etc. Nous présentons ici celle de Newton-Raphson (NR), qui reste simple et donne en général de bons résultats. Toutefois, signalons que le choix de la méthode et des condition initiales est d'une grande importance et constitue une partie significative de la modélisation du filtre. Un mauvais choix peut conduire à une absence de solution trouvée en temps raisonnable, ou un résultat trop imprécis se manifestant sous forme de bruit et de craquements. Il peut être parfois nécessaire de combiner plusieurs approches.

La méthode NR est itérative. On part d'une estimation grossière (enfin, pas trop non-plus) pour y, qui est affinée au long des itérations. Il y a différents critères d'arrêt : quand $f(y)$ est suffisamment proche de 0, quand la distance entre deux y consécutifs est suffisamment petite, ou au contraire quand on n'arrive pas à trouver de solution satisfaisante après un nombre d'itérations fixé à l'avance. Ce dernier cas devrait s'avérer rare mais nécessite malgré tout d'être anticipé.

Commençons par indexer $y^{[m]}$. Cet index suit les itérations NR, il n'a rien à voir avec l'indexation temporelle utilisée pour identifier les instants d'échantillonnage. D'autre part, pour que NR puisse marcher correctement, f est censée être C_1 et strictement monotone. C'est le cas pour nos diodes tête-bêche. L'étape itérative de NR est la suivante :

$$d^{[m]} = -\frac{f(y^{[m]})}{f'(y^{[m]})} \tag{4.12}$$

$$y^{[m+1]} = y^{[m]} + d^{[m]} \tag{4.13}$$

Avec ici :

$$f'(y) = \frac{1+g}{gR} + S'(y) \tag{4.14}$$

On commence avec $y^{[0]}$, notre estimation initiale. Mais quelle valeur lui donner ? Il existe différentes possibilités. On peut récupérer la dernière valeur trouvée pour $y[n-1]$ à l'instant précédent. C'est une solution très courante et nous l'adopterons, mais elle ne donne pas forcément les meilleurs résultats. Il y a d'autres façons de faire, citons par exemple l'interpolation de tables ou l'approximation linéaire de S. On peut aussi extrapoler linéairement la valeur de l'intégrateur avec $y^{[0]} = 2y[n-1] - s$. Il n'y a pas de solution universelle, il faut les essayer, les adapter et être imaginative. L'algorithme NR est très sensible à l'estimation initiale, c'est donc une étape à concevoir avec soin.

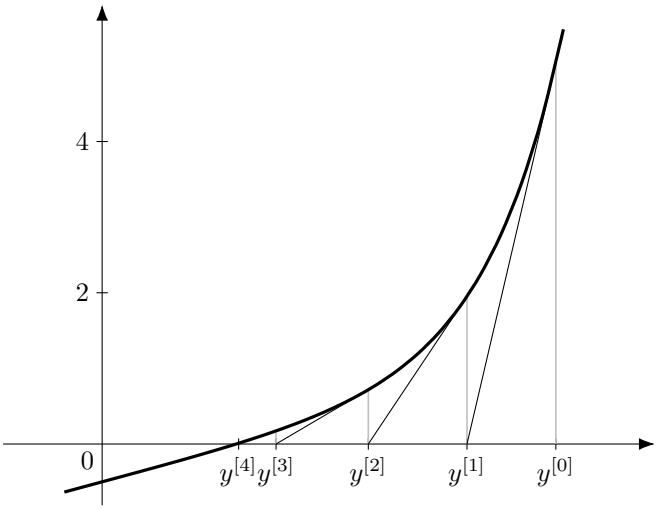


FIGURE 4.5 – *Convergence de l'algorithme de Newton-Raphson en quelques itérations. L'estimation initiale $y^{[0]}$ est éloignée de la solution mais la forme de la courbe sur l'intervalle parcouru reste favorable à une convergence rapide.*

Dans le même ordre d'idée, notons que NR peut avoir des difficultés à converger si l'estimation initiale est faite du mauvais côté d'un coude à petit rayon de courbure de la courbe de f. La première itération peut catapulter la solution de l'autre côté, parfois à une distance importante, et celle-ci peut mettre du temps à revenir sur un intervalle plus favorable. Pour pallier ce genre de problème, il est souvent utile de limiter $d^{[m]}$ en valeur absolue, de façon à éviter les grands sauts. Cette limite peut éventuellement être adaptée en fonction de $y^{[m]}$ si on connaît à l'avance la tête de la courbe. On peut aussi passer à une itération en courant plutôt qu'en tension, comme le font les simulateurs de circuits dans ce genre de cas.

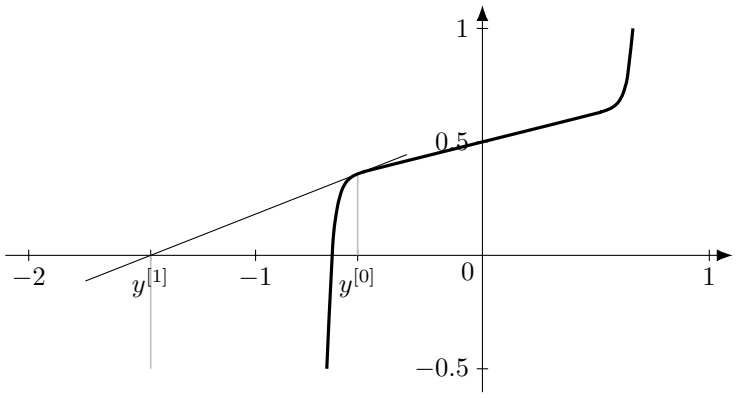


FIGURE 4.6 – *L'algorithme de Newton-Raphson mis en échec par une mauvaise estimation initiale $y^{[0]}$. La première itération pousse f vers des valeurs stratosphériques, et il faudra de nombreuses étapes avant de retrouver un intervalle qui fasse converger l'algorithme de manière attendue, c'est-à-dire quadratique. On voit que $y^{[0]} = 0$ est une très bonne estimation quand la solution se trouve dans la zone quasiment linéaire, mais une des plus mauvaises pour les zones saturées.*

Signalons au passage que le type des opérandes (virgule flottante en simple ou double précision) peut avoir un impact significatif sur les performances générales du système. Le calcul en `double` est généralement deux fois plus lent que celui en `float`, cependant l'imprécision de ces derniers peut nuire sérieusement à la convergence quand il y a plusieurs non-linéarités ou un nombre conséquent de nœuds. Le bruit de calcul peut même devenir une importante source de parasites. Pour un filtre donné, il est ainsi recommandé de tester les deux types avant de faire un choix définitif. Les résultats sont parfois contre-intuitifs.

De même, dans les itérations NR, le calcul de la dérivée doit être précisément ajusté à la fonction principale, faute de quoi la convergence est beaucoup moins rapide. Ainsi, une approximation de la fonctions exponentielle doit être suffisamment précise pour que l'approximation de sa dérivée corresponde. Là aussi, on peut facilement se faire piéger, une économie faite sur l'approximation peut facilement devenir contre-productive.

4.1.3 Codage

Nous pouvons maintenant tirer quelques lignes de code de toutes ces équations. On initialise le filtre ainsi :

```
s  := 0
yz := 0
```

Avec comme paramètres :

```
g    := tan (pi * f0 / fs)
gri  :=  1 / (g * R)
g1gr := (1 + g) * gri
```

Et pour chaque échantillon `x` en entrée, `y` en sortie :

```
a  := (s + g * x) * gri
y  := yz          // Estimation initiale
while (...)       // Newton-Raphson
{
   f  := S (y)  + g1gr * y - a
   fd := Sd (y) + g1gr
   d  := f / fd
   y  := y - d
}
v  := y - s
s  := y + v
yz := y
```

La condition de sortie du NR n'est pas précisée dans ce morceau de code. En général, il faut 2 ou 3 itérations en moyenne pour obtenir le résultat. Si le signal arrive très fort (par exemple plus de 40 dB au-dessus du niveau crête), le nombre moyen d'itérations peut doubler, avec des pointes à plus de 30 itérations.

Les fonctions `S()` et `Sd()` implémentent l'antisaturation S et sa dérivée. `yz` représente la valeur de y à la fin de l'itération temporelle précédente. Nous n'aborderons pas ici l'implémentation et l'optimisation ces fonctions, le présent article étant déjà chargé.

Si on fait le bilan de cet algorithme en charge de calcul, on obtient pour la boucle principale, estimation initiale comprise :

- 2 mul
- 3 add
- 2 mémoires

Et en plus, par itération NR, hors condition d'arrêt :

- 1 évaluation de S
- 1 évaluation de S'
- 1 div
- 1 mul
- 4 add

Il est possible d'absorber `g1gr` dans le paramétrage et dans `S` et `Sd` et ainsi de faire sauter une multiplication dans le calcul de `f`. Nous laissons cet exercice à nos lectrices.

Notons qu'ici, l'estimation initiale ne pèse rien face à une seule itération NR. On a donc tout intérêt à perfectionner cette estimation initiale si cela peut réduire la quantité d'itérations. Lors de la conception du filtre, ne pas hésiter à mettre en place un système de statistiques, de façon à mesurer les temps de convergence en fonction des caractéristiques du signal d'entrée et des paramètres du filtre. Étudier précisément les cas les plus défavorables. Nous en resterons là à propos de l'optimisation de la convergence, qui mériterait un ouvrage entier à elle seule.

On remarque quand même que la valeur de R n'a qu'une influence très limitée sur la distorsion. Combinée à C, elle définit la fréquence de coupure du filtre. Sur les diodes, elle va fixer l'échelle de la tension de sortie, qui physiquement ne peut pas dépasser les limites thermiques de ces semi-conducteurs. On pourrait penser que cette échelle a une influence sur la forme, mais ce n'est pas vraiment le cas à cause de la nature exponentielle de la caractéristique des diodes. Il est donc judicieux de sortir R et C des équations et de ne travailler qu'en termes de fréquence de coupure. Ainsi cette dernière n'aura pas d'influence sur l'amplitude de la distorsion — le découplage des paramètres étant souvent préférable en matière d'utilisation. Si on veut changer l'échelle de la distorsion, il vaut mieux changer les niveaux d'entrée en amont et de sortie en aval, ou encore modifier le paramétrage de la diode avec n et I_S. Ce genre de chose ne serait pas possible avec un circuit réel !

4.2 Méthode par sous-circuits équivalents

Une autre façon d'aborder le problème est de transformer la non-linéarité directement dans le circuit, exactement de la façon dont nous avons transformé le condensateur en son équivalent Norton instantané. Nous conservons le modèle où les diodes tête-bêche sont groupées en un seul dipôle équivalent, de caractéristique S.

4.2.1 Modèles équivalents

Nous allons définir ainsi un modèle linéaire pour notre antisaturateur, modèle qui dépend d'un point de fonctionnement (v_{D0}, i_{D0}). Comme dans la méthode précédente, le point de fonctionnement sera estimé grossièrement au début de chaque étape temporelle, puis sa position sera affinée par itérations. Le schéma de la figure 4.7 montre notre filtre passe-bas avec ses générateurs équivalents.

L'équivalent du condensateur est donné par les formules (2.9), (2.10) et (2.11). Celui des diodes prend une forme très semblable, à une différence près : l'indexation est relative à l'itération interne et non au temps. Nous notons cette indexation de

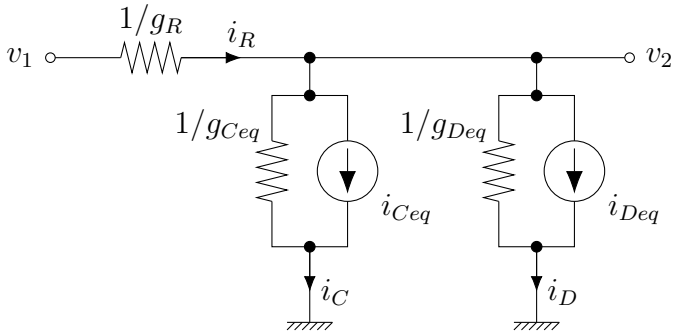


FIGURE 4.7 – *Un filtre passe-bas avec des diodes de saturation, composants sous forme d'équivalents Norton*

la manière suivante : $v_D^{[m]}$. La modélisation est donnée par l'équation ci-dessous :

$$i_D^{[m]} = g_{Deq}^{[m-1]} \cdot v_D^{[m]} + i_{Deq}^{[m-1]} \tag{4.15}$$

avec :

$$g_{Deq}^{[m-1]} = \frac{di_D}{dv_D} \qquad = S'\left(v_D^{[m-1]}\right) \tag{4.16}$$

$$i_{Deq}^{[m-1]} = i_D^{[m]} - g_{Deq}^{[m-1]} \cdot v_D^{[m]} = S\left(v_D^{[m-1]}\right) - S'\left(v_D^{[m-1]}\right) \cdot v_D^{[m-1]} \tag{4.17}$$

Cette modélisation est une approximation linéaire de la courbe de S calquée sur sa tangente au point $v_D^{[m-1]}$. Ça ne commence pas à rappeler quelque chose ?

4.2.2 Mise en équations

Pour mettre le circuit en équations, on repart de la loi des nœuds de Kirchhoff (4.3). On y rajoute les autres relations tension-courant. Par souci d'homogénéité, nous associons à R sa conductance $g_R = 1/R$.

$$\begin{cases} i_R = i_C + i_D^{[m]} \\ i_R = g_R(v_1 - v_2^{[m]}) \\ i_C + i_D^{[m]} = (g_{Ceq} + g_{Deq}^{[m-1]}) \cdot v_2^{[m]} + i_{Ceq} + i_{Deq}^{[m-1]} \end{cases} \tag{4.18}$$

Nous en tirons :

$$v_2^{[m]} = \frac{g_R v_1 - i_{Ceq} - i_{Deq}^{[m-1]}}{g_R + g_{Ceq} + g_{Deq}^{[m-1]}} \tag{4.19}$$

On introduit (4.17) dans (4.19) :

$$v_2^{[m]} = \frac{g_R v_1 - i_{Ceq} - S\left(v_2^{[m-1]}\right) + S'\left(v_2^{[m-1]}\right) \cdot v_2^{[m-1]}}{g_R + g_{Ceq} + S'\left(v_2^{[m-1]}\right)} \tag{4.20}$$

On obtient ainsi une itération permettant de résoudre l'équation non-linéaire. Il s'agit même précisément d'une itération Newton-Raphson. Pas convaincue ? Reformulons (4.20) relativement à $v_2^{[m-1]}$:

$$v_2^{[m]} = v_2^{[m-1]} + \frac{g_R v_1 - i_{Ceq} - S\left(v_2^{[m-1]}\right) - v_2^{[m-1]} \cdot (g_R + g_{Ceq})}{g_R + g_{Ceq} + S'\left(v_2^{[m-1]}\right)} \tag{4.21}$$

$$= v_2^{[m-1]} - \frac{f(v_2^{[m-1]})}{f'(v_2^{[m-1]})} \tag{4.22}$$

avec :

$$f(v) = S(v) + (g_R + g_{Ceq}) \cdot v - g_R v_1 + i_{Ceq} \tag{4.23}$$

$$f'(v) = S'(v) + (g_R + g_{Ceq}) \tag{4.24}$$

Il est intéressant de noter que les variables liées au modèle équivalent des diodes ont disparu de l'équation.

v_2 connu, on peut passer au reste de l'itération temporelle. Il s'agit essentiellement de mettre à jour le courant du condensateur. Et le tour est joué.

4.2.3 Codage

Le pseudo-code suivant est obtenu en rassemblant et ordonnant les équations (4.20), (2.9), (2.10) et (2.11). Pour l'initialisation :

```
iceq := 0
v2z  := 0
```

Le paramétrage :

```
gceq := 2 * C * fs
zceq := 1 / gceq
```

Et l'itération temporelle :

```
a    := x * gr - iceq
b    :=     gr + gceq
v2   := v2z   // Estimation initiale
while (...)
```

```
{
    s0  := S (v2)
    s1  := Sd (v2)
    num := a + s1 * v2 - s0
    den := b + s1
    v2  := num / den
}
ic   :=  gceq * v2 + iceq
iceq := -gceq * v2 - ic
v2z  := v2
```

Nous laissons l'optimisation de ce code en exercice. En l'état, il est très semblable à celui que nous avons trouvé avec la technique ZDF. Encore une fois, ce n'est pas une coïncidence.

Note 1 : pour les antisaturateurs exponentiels, comme celui de nos diodes, il existe des solutions quasiment analytiques qui peuvent être approximées directement et économiquement [9]. Cependant l'objectif de cet article est de montrer une méthode de conception et de résolution plutôt qu'une solution particulière.

Note 2 : si les filtres ainsi formés sont stables et reproduisent assez fidèlement leurs équivalents analogiques, ils ne sont pas débarrassés du fléau qui accompagne les non-linéarités : l'*aliasing* (repliement spectral). Pour l'atténuer, il sera nécessaire de sur-échantillonner le filtre ou d'utiliser des stratagèmes équivalents [8]. Au-delà de générer des tonalités parasites typiques, le repliement peut véritablement pourrir le comportement d'un filtre en créant des « verrouillages » de résonance autour de certaines fréquences.

5 Conclusion

Nous avons vu deux méthodes pour simuler des circuits électroniques simples et obtenir des filtres numériques relativement fidèles à leurs homologues analogiques. Ces approches sont techniquement équivalentes mais ouvrent des perspectives différentes en matière de conception.

Chaque montage étant unique, il est difficile d'utiliser des briques de base pour construire des filtres plus élaborés, comme avec la BLT. En revanche, cette particularité permet d'optimiser efficacement le code et de résoudre des difficultés spécifiques, par exemple liées à la convergence.

Toutefois ces méthodes manuelles sont limitées à des circuits de faible complexité. Si on veut passer à la vitesse supérieure, il est nécessaire de recourir à des algorithmes spécifiques, comme l'analyse nodale modifiée (MNA, *modified nodal analysis*) [4] ou la *nodal DK method* [7]. L'analyse par modèles linéaires équiva-

lents est centrale pour ces méthodes automatisées, et sa bonne compréhension est indispensable avant d'attaquer de tels algorithmes dans toute leur complexité.

6 Annexes

6.1 Bibliographie

[1] Lawrence R. Rabiner, Bernard Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, 1975, ISBN 0139141014

[2] Vesa Välimäki, Timo I. Laakso *Suppression of transients in time-varying recursive filters for audio signals*, Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'98), vol. 6, pp. 3569–3572, Seattle, Washington, May 12–15, 1998

[3] Robert Bristow-Johnson, *Cookbook formulae for audio EQ biquad filter coefficients*, 1998,
`https://music.columbia.edu/pipermail/music-dsp/1998-October/054185.html`

[4] Stefan Jahn, Michael Margraf, Vincent Habchi, Raimund Jacob, *Qucs technical papers*, Transient Analysis, 2005,
`http://qucs.sourceforge.net/tech/node23.html`

[5] Victor Kalinichenko, *Smooth and safe parameter interpolation of biquadratic filters in audio applications*, Proceedings of the 9th International Conference on Digital Audio Effects (DAFx–06), Montreal, Canada, September 18–20, 2006

[6] Andrew Simper, *Linear Trapezoidal integrated SVF vs Direct Form 1 and other variations with 32-bit floating point numbers*, Cytomic, 2011,
`https://cytomic.com/index.php?q=technical-papers`

[7] Piero Rivera Benois, *Simulation Framework for Analog Audio Circuits based on Nodal DK Method*, master thesis, Helmut Schmidt University, 2013

[8] Julian D. Parker, Vadim Zavalishin, Efflam Le Bivic, *Reducing the Aliasing of Nonlinear Waveshaping Using Continuous-Time Convolution*, Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic, September 5–9, 2016

[9] Stefano D'Angelo, Leonardo Gabrielli, Luca Turchet, *Fast Approximation of the Lambert W Function for Virtual Analog Modelling*, Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), Birmingham, UK, September 2–6, 2019

[10] Vadim Zavalishin, *The Art of VA Filter Design*, v2.1.2, 2020,
`https://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_2.1.2.pdf`

6.2 Glossaire

BLT *Bilinear transform*, transformée bilinéaire

MNA *Modified nodal analysis*, analyse nodale modifiée

TPT *Topology preserving transform*, transformation préservant la topologie

ZDF *Zero delay feedback*, contre-réaction instantanée

6.3 Historique des révisions

v1.0	2020-04-23	Première version publiée
v1.1	2020-07-15	Ajouts dans les considérations sur le filtre passe-bas, modification du résumé, remarques sur la qualité de la convergence, quelques reformulations