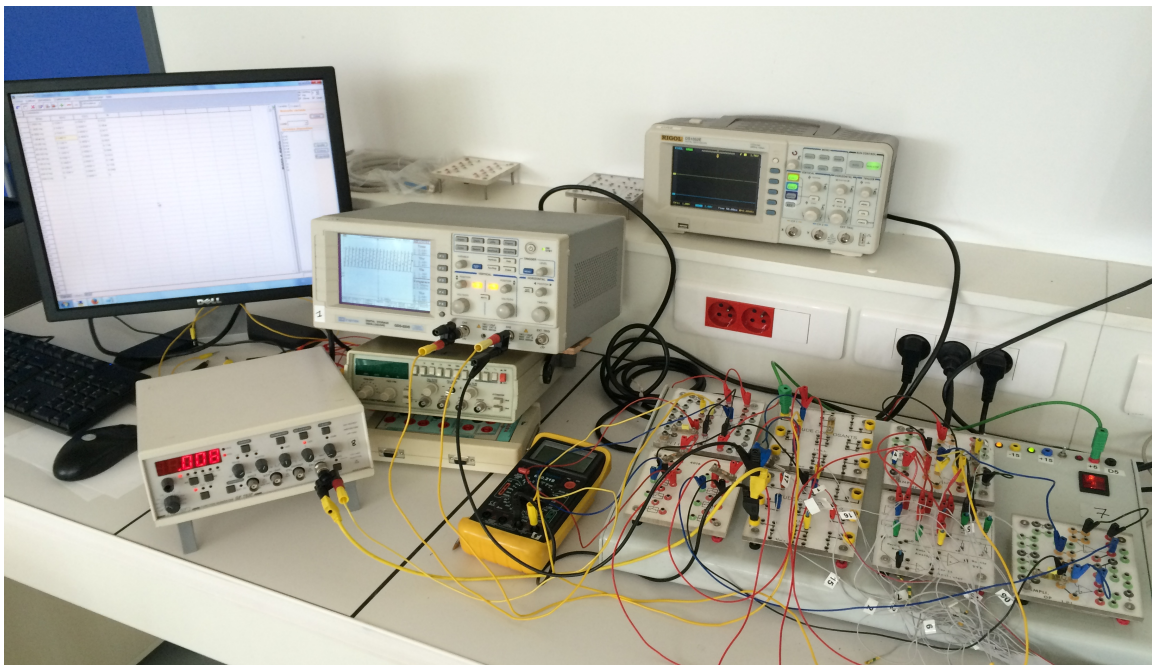
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6

STPI/P6/2017 – 030

FILTRAGE NUMERIQUE EN TEMPS REEL



Etudiants :

Matthieu BELLUCCI **Vivien DALIGAUX**

Rémi HEMERY **Alexis PETIOT**

Walid ZOUHRY

Enseignant-responsable du projet :

François GUILLOTIN

Date de remise du rapport : ***19/06/2017***

Référence du projet : ***STPI/P6/2017 – 030***

Intitulé du projet : ***Filtrage numérique en temps réel***

Type de projet : ***Théorique et expérimental***

Objectifs du projet : Réaliser le montage d'un filtre numérique modulable afin de filtrer des signaux en temps réel. Approche théorique avec calculs des valeurs mathématiques. Application aux expériences pour vérifier le fonctionnement du filtre.

Mots-clefs du projet *(4 maxi)* : ***filtrage numérique, intégrateur, Simpson***

TABLE DES MATIERES

NOTATIONS, ACRONYMES

CAN : Convertisseur Analogique-Numérique

CNA : Convertisseur Numérique-Analogique

TFD : Transformée de Fourrier Discrète

1. INTRODUCTION

Le projet P6 prend place au cours du 4ème semestre de cycle STPI. Il permet aux élèves ingénieurs de fournir un travail et une réflexion de groupe avancée sur un sujet précis qui intéresse les élèves. En effet, le sujet est libre parmi une liste riche en thèmes.

Ce projet a pour objectif de familiariser les étudiants à une démarche de réflexion plus proche de celle d'un ingénieur, avec une approche moins scolaire.

C'est dans ce cadre que nous avons choisi de traiter le sujet « Filtrage numérique en temps réel ». Nous avons, sur une période de plusieurs mois, étudié le fonctionnement d'un filtre numérique modulable et réalisé des expériences afin d'en comprendre le fonctionnement. Pour cela, nous avons du effectuer des calculs mathématiques théoriques mais aussi manipuler des circuits électriques complexes.

2. MÉTHODOLOGIE / ORGANISATION DU TRAVAIL

La première partie du projet a en grande partie consisté en un travail de recherche et de documentation afin d'apprendre à connaître ce sujet que nous ne connaissions que peu encore. Une fois le montage effectué par notre tuteur, nous avons pu réaliser plusieurs expériences. Pour les mener à bien, le calcul littéral de coefficients était nécessaire comme nous pourrons le voir dans la suite du rapport. Nous nous sommes organisés de la manière suivante : deux élèves effectuaient les calculs mathématiques théoriques alors que les autres réalisaient les mesures en utilisant les valeurs trouvées.

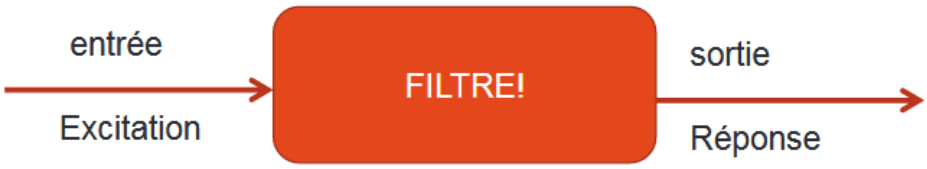
De manière générale, nous avons assez peu travaillé en dehors du créneau horaire dédié au projet mis à part pour la rédaction de ce rapport. En effet, la nécessité d'avoir le montage pour effectuer les recherches limitait le travail « personnel » à un travail de documentation. Enfin, nous nous sommes répartis les paragraphes pour la rédaction du rapport puis nous avons mis en commun afin d'apporter des idées nouvelles ou éclaircissements sur les parties de chacun.

3. TRAVAIL RÉALISÉ ET RÉSULTATS

3.1. Introduction générale sur les filtres numériques

Dans un monde de plus en plus numérisé, le filtrage numérique prend une place de plus en plus importante. Ainsi nous retrouvons le filtrage numérique dans plusieurs appareils de notre quotidien : les radios, les téléphones, les télévisions… De plus, le filtrage numérique intervient dans le traitement du signal, mais nous le retrouvons aussi dans différents types de logiciels de traitement audio ou image.

Techniquement, un filtre reçoit un signal d'entrée auquel il opère certaines modifications pour émettre un signal de sortie. Techniquement, il effectue ces opérations en appliquant des formules mathématiques sur l'échantillon sélectionné.



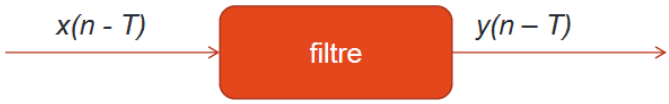
Face à l'étendue des fréquences auxquelles un appareil doit répondre, il existe différents types de filtres et différentes techniques de filtrage : filtrage non-récursif, filtrage récursif et filtrage par produit TFD.

D'autre part, les filtres ont plusieurs caractéristiques qui leurs sont communes. En effet les filtres peuvent être linéaires ou non-linéaires, variant ou invariant dans le temps, adaptatifs ou non-adaptatifs et enfin récursifs ou non-récursifs.

Chaque caractéristique modifie la réponse du filtre, par conséquent, pour connaître l'effet d'un filtre, il faut représenter son action dans un domaine temporel (quels sont les changement opérés dans le temps) mais aussi dans un domaine fréquentiel (l'effet du filtre sur les composantes fréquentielles).

Dans notre projet , on s'intéressera à l'étude d'un filtre passe-bas et d'un filtre passe-bande qui sont deux filtres invariants dans le temps mais aussi linéaires.

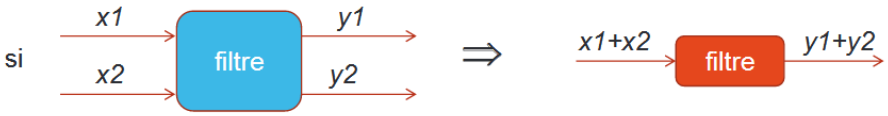
- **Invariance dans le Temps**



un retard de *T* échantillons

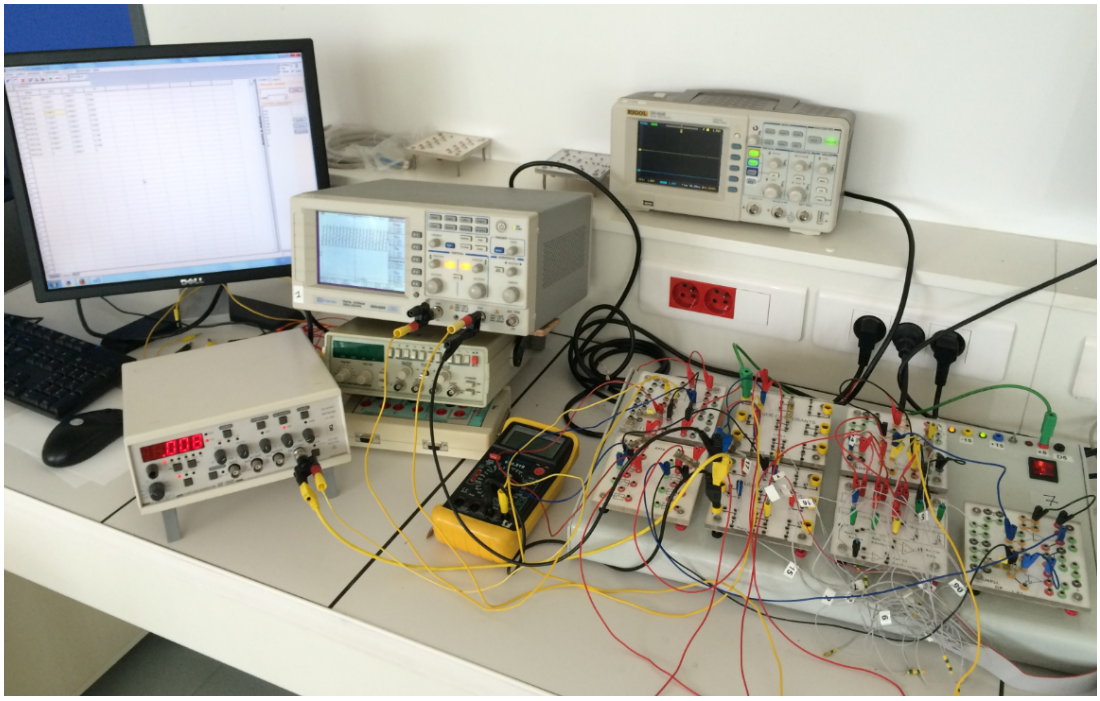
- **Linéarité**
  - Proportionnalité:

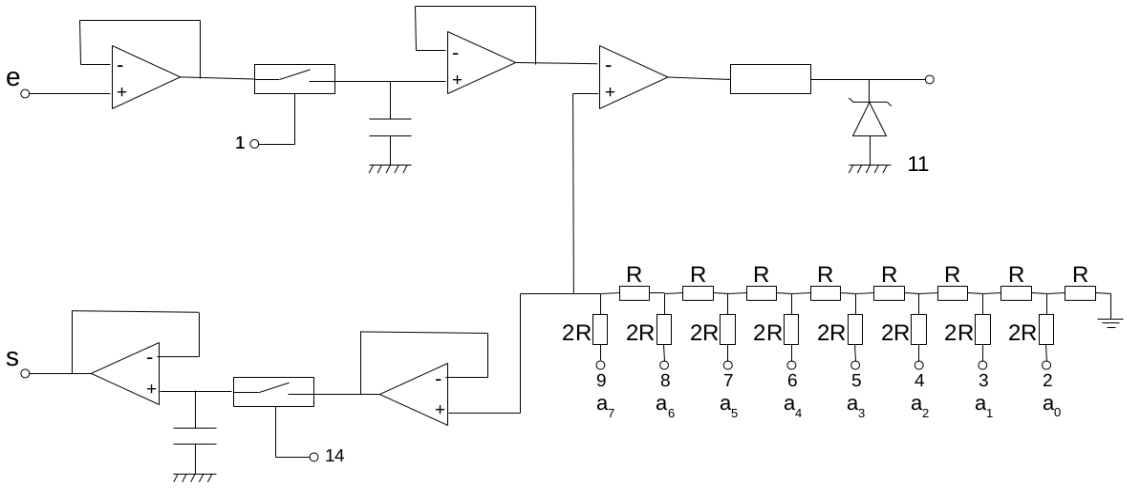
  - Superposition:



Linéaire ➔ des composants spectrales ne sont pas ajoutées.

3.2. Montage électrique





Le montage est composé de 5 amplificateurs, 2 interrupteurs, 2 condensateurs et 8 résistances.

Le montage ci-dessous se divise en plusieurs parties dont chacune est spécialisée dans une fonction précise. Le premier amplificateur a pour fonction d'amplifier le signal d'entrée avant d'être stocké dans le condensateur, lui-même lié à un interrupteur contrôlé par l'ordinateur. L'interrupteur est fermé quand le programme ordonne l'échantillonnage.

Le deuxième amplificateur, quant à lui , sert à éviter le déchargement du condensateur et donc de maintenir l'information sur le signal d'entrée. Le signal passe ensuite par une résistance avant d'être traité par l'ordinateur une fois que le signal analogique est transformé en signal numérique.

Le signal, une fois traité par l'ordinateur, passe par un montage R-2R. Les entrées du R-2R sont reliées aux ports de transmission de données du câble DB25. Cette partie a pour objectif de reconvertir les valeurs numériques en valeurs analogique. Il remplit donc le rôle de CNA. La tension est ensuite transmise au travers d'un réseau d'amplificateurs identique à celui du début du montage.

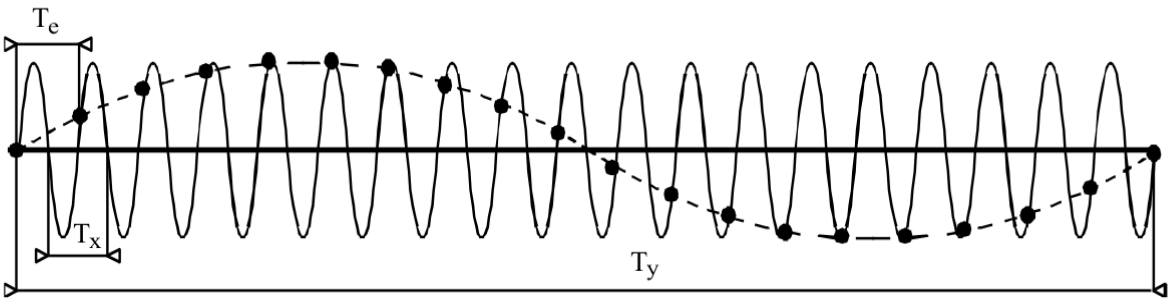
De l'entrée à la sortie, le montage convertit le signal d'entrée en numérique (CAN) puis après traitement par l'ordinateur convertit la valeur numérique en analogique (CNA).

3.3. Choix de la fréquence d'échantillonnage

Comme expliqué dans l'introduction, un filtre a pour but de laisser passer ou alors d'atténuer le signal en entrée selon une fréquence donnée et d'y associer un signal de sortie. Ce signal de sortie est calculé selon des formules mathématiques et il est donc nécessaire d'avoir des valeurs numériques afin d'effectuer ces opérations. Or, le signal d'entrée est analogique. La conversion du signal analogique vers un signal numérique est effectuée par un CAN (Convertisseur Analogique-Numérique) qui relève des la valeur de tension à intervalle de temps régulier. Cette intervalle est appelé période d'échantillonnage, à laquelle est liée la fréquence d'échantillonnage.

La problématique de cette partie est de définir la fréquence d'échantillonnage afin que le signal numérique donné par le CAN soit une approximation acceptable du signal analogique en entrée. En effet, si celle-ci n'est pas suffisamment grande un phénomène appelé « repliement » peut apparaître dans certains cas.

Dans cet exemple, nous utiliserons un filtre passe-tout qui n'atténue le signal d'entrée pour aucune fréquence.

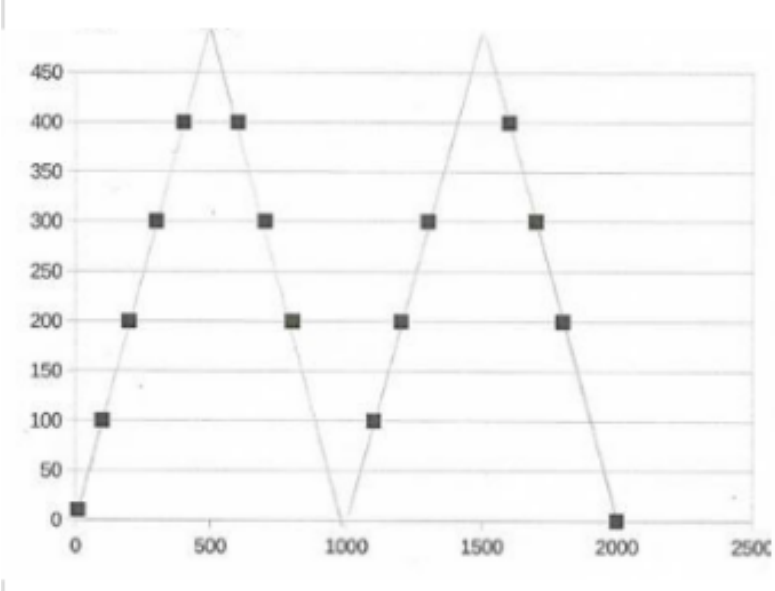


On observe qu'avec une période d'échantillonnage T_e trop grande, le signal après le CAN (ici celui en pointillé) ne correspond pas du tout à celui en entrée. En effet, le signal de sortie possède une fréquence plus petite : c'est ce qu'on appelle le phénomène de repliement du spectre.

Comment éviter ce phénomène et avoir une bonne approximation du signal analogique ? Il paraît évident que la réponse est de diminuer la période d'échantillonnage, donc d'augmenter la fréquence d'échantillonnage. Le théorème de Shannon énonce que pour un signal d'entrée sinusoïdal, la fréquence d'échantillonnage doit au moins être deux fois supérieure à la fréquence du signal d'entrée. Autrement dit, il faut minimum deux échantillons par période de signal d'entrée.

Bien sûr, augmenter plus f_e permet d'avoir une meilleure finesse et donc une meilleure approximation. C'est pourquoi dans la pratique, on choisit au moins 10 points d'échantillonnage par période pour une reconstruction par marche d'escalier.

Afin d'observer ce phénomène, nous avons réalisé une expérience. Nous avons observé au travers d'un filtre passe-tout la fréquence de sortie pour des signaux de fréquence d'entrée divers avec une fréquence d'échantillonnage de 1000 Hz.



fx (entrée)	fy (sortie)
10	10
100	1,00E+02
200	2,00E+02
300	300
400	400
1100	100
1200	200
1300	300
1600	400
1700	300
1800	200
2000	0
600	400
700	300
800	200

On constate bien que la fréquence de sortie n'est pas identique à celle du signal d'entrée lorsque la fréquence de celui-ci est trop élevée.

3.4. Obtention du signal de sortie par le calcul

3.4.1. Approximation d'une intégrale

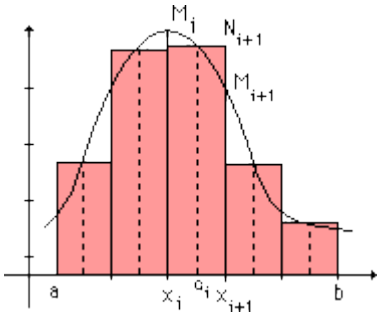
Notre objectif est d'obtenir une relation de récurrence de la forme :

$$y_k = (\alpha_0 x_k + \alpha_1 x_{k-1} + ... + \alpha_n x_{k-n}) - (\beta_1 y_{k-1} + \beta_2 y_{k-2} + ... + \beta_n y_{k-n})$$

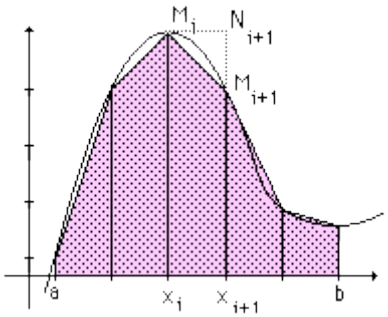
Afin d'obtenir une relation de récurrence définissant le signal, nous passerons par l'approximation de l'intégrale du signal.

3.4.1.1. Méthode des trapèzes

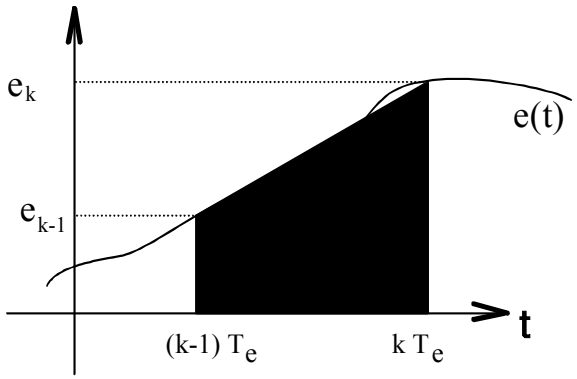
La méthode d'intégration approchée, dite des trapèzes, décrite ci-après, introduite par Newton & Cotes est plus précise que la méthode élémentaire, dite des rectangles, correspondant aux sommes de Riemann consistant à remplacer la fonction initiale par une approximation en escalier. Graphiquement, sur l'intervalle $[x_i, x_{i+1}]$, on remplace l'arc de courbe par le segment $[M_iN_{i+1}]$, donc l'aire sous la courbe, par le « rectangle » $x_iM_iN_{i+1}x_{i+1}$



La méthode des *trapèzes*, étudiée ici, remplace tout arc de courbe correspondant à $[x_i,x_{i+1}]$ par le segment $[M_iM_{i+1}]$, donc l'aire sous la courbe, par « trapèze » $x_iM_iM_{i+1}x_{i+1}$au lieu du « rectangle » $x_iM_iN_{i+1}x_{i+1}$:



Soit un filtre intégrateur tel que $s(t_1) = \int_0^{t_1} e(u)du$, $s(t)$ étant la sortie et $e(t)$ l'entrée de ce filtre.



Soit s_{k-1} la sortie à l'instant $(k-1)T_e$, à l'instant suivant kT_e on prélève la valeur d'entrée e_k.

La valeur de la sortie s_k s'obtient en ajoutant la surface hachurée à s_{k-1}. On obtient alors $s_k = s_{k-1} + \frac{e_k + e_{k-1}}{2} T_e$.

A l'instant kT_e, la sortie s_k constitue alors la somme des surfaces hachurées jusqu'à cet instant, ce qui est bien l'approximation de l'intégrale de l'entrée **en fonction du temps**.

Pour appliquer cette technique au filtrage numérique, **on simplifie le travail** en passant par la notion de **fonction de transfert.**

Pour le filtre analogique on utilise la fonction de transfert $H_a(p) = \frac{S(p)}{E(p)}$ par la transformation de LAPLACE.

Pour le filtre numérique on utilise la fonction de transfert $H_n(Z) = \frac{S(Z)}{E(Z)}$ par la transformation en Z.

3.4.1.2. Méthode de Simpson

Soit f une fonction quelconque. On souhaite calculer l'intégrale de cette fonction entre deux bornes a et b. Cependant, il est impossible de demander à un ordinateur de calculer automatiquement l'intégrale de n'importe quelle fonction. C'est pourquoi on utilise une approximation de ce calcul. La méthode de Simpson propose de calculer l'intégrale du polynôme quadratique le plus proche.

$$\int_a^b f(x)\,dx \approx \int_a^b P(x)\,dx$$

Avec P(x) = mx² + cx + d . Afin d'obtenir le polynôme de degré 2 le plus proche, on prend 3 points sur la courbe de f, à savoir f(a), f(b) et f((a+b)/2) afin d'avoir un intervalle constant entre les abscisses. Il existe plusieurs moyens d'obtenir la formule suivante :

$$\int_a^b P(x)\,dx = \frac{b-a}{6} + [f(a) + 4f(\frac{a+b}{2}) + f(b)]$$

On obtient cette formule en utilisant l'interpolation lagrangienne ou la formule de Newton-Cotes, qui permet de trouver P(x), puis de calculer son intégrale. Ou on calcule l'intégrale de P(x) entre a et h, avec h = (b-a)/2 à laquelle on additionne l'intégrale entre b-h et b. Après simplification et identification des termes, on obtient le même résultat. C'est cette dernière méthode que nous avons utilisé afin de démontrer cette formule.

Enfin, pour diminuer l'erreur due à l'approximation, on peut prendre des points plus proches. Ainsi, pour calculer l'intégrale d'une fonction quelconque entre deux bornes, on utilise la relation de Chasles, afin de calculer la somme des intégrales entre deux points proches, puis on utilise la méthode de Simpson pour approximer chacune de ces intégrales.

Cette approximation est plus précise que la méthode des trapèzes car elle prend 3 points de la fonction à approximer, plutôt que 2. Cependant, cela la rend plus longue à calculer pour l'ordinateur, ralentissant ainsi la vitesse de traitement du signal.

En appliquant la formule précédente à notre situation, sachant que f(b)= e_k , f(a)= e_{k-2} , f((a+b)/2)= e_{k-1} et (b-a)/2 est la période et est donc égale à T_e, on obtient alors:

$$s_k = \sum_{i=2}^{k} \frac{T_e}{3}(e_{i-2} + 4e_{i-1} + e_i)$$

$$s_k = s_{k-1} + \frac{T_e}{3}(e_{k-2} + 4e_{k-1} + e_k)$$

3.4.2. Transformée en Z

La transformation en Z est une application qui transforme une suite *s* (définie sur les entiers) en une fonction *S* d'une variable complexe nommée *z*, telle que **:**

$$S(Z) = Z\{s(n)\} = \sum_{n=-\infty}^{+\infty} s(n)z^{-n}, z \in \left\{z \in C \mid \sum_{n=-\infty}^{+\infty} s(n)z^{-n} \text{ converge}\right\}$$

La variable *n* représente en général le *temps* discrétisé, la variable complexe *z* n'est qu'un être mathématique. Lorsqu'on travaille sur *s*(*n*) on dit que l'on est dans le *domaine temporel*, lorsqu'on travaille sur *S*(*z*) le domaine est appelé *fréquentiel* par analogie avec la transformée de Fourier.

Si $\forall n < 0, s(n) = 0$, on parle de signal causal. Les signaux étudiés durant ce projet sont tous causaux.

La transformation en Z s'introduit de plusieurs façons, ici c'est la relation de récurrence qui nous intéresse.

Soit la relation de récurrence entre la suite des valeurs de sortie {sk} et la suite des valeurs d'entrée {ek}

$$b_0 s_k + b_1 s_{k-1} + b_2 s_{k-2} + \ldots + b_m s_{k-m} = a_0 e_k + a_1 e_{k-1} + \ldots + a_q e_{k-q}$$

On a alors l'équation :

$$s_k = \frac{1}{b_0}\left[a_0 e_k + a_1 e_{k-1} + \ldots + a_q e_{k-q} - (b_1 s_{k-1} + b_2 s_{k-2} + \ldots + b_m s_{k-m})\right]$$

dite équation aux différences finies.

On fait la correspondance suivante :

La suite {wk} se transforme en W(Z) , où W(Z) est une fonction du nombre complexe Z.

La suite {wk-n} se transforme en W(Z) Z-n

On utilise la quantité Z = epTe qui joue le rôle d'un opérateur de décalage. L'expression exacte de la fonction

W(Z) n'intervient pas pour le T.P. . On pourra consulter l'annexe.

La relation de récurrence :

$$b_0 s_k + b_1 s_{k-1} + b_2 s_{k-2} + \ldots + b_m s_{k-m} = a_0 e_k + a_1 e_{k-1} + \ldots + a_q e_{k-q}$$

donne une relation entre les suites {sk} {sk-1} ...{sk-m} et {ek} {ek-1} ...{ek-q}

ce qui donne finalement :

$$b_0 S(Z) + b_1 S(Z) Z^{-1} + b_2 S(Z) Z^{-2} + \ldots + b_m S(Z) Z^{-m} = a_0 E(Z) + a_1 E(Z) Z^{-1} + \ldots + a_q E(Z) Z^{-q}$$

La fonction de transfert en Z est alors :

$$H_n(Z) = \frac{S(Z)}{E(Z)} = \frac{a_0 E(Z) + a_1 E(Z) Z^{-1} + a_2 E(Z) Z^{-2} + \ldots + a_q E(Z) Z^{-q}}{b_0 S(Z) + b_1 S(Z) Z^{-1} + b_2 S(Z) Z^{-2} + \ldots + b_m S(Z) Z^{-m}}$$

3.4.3. Transformée de Laplace, pic de Dirac et produit de convolution

La transformée de Fourier est un outil fondamental du traitement du signal qui permet d'associer une représentation fréquentielle du signal à la représentation temporelle classique. Cette image fréquentielle permet notamment de définir et étudier simplement les filtres linéaires et d'analyser des signaux périodiques.

Seulement, il est courant de rencontrer des systèmes soumis à des contraintes non périodique comme des impulsions de Dirac ou des fonctions échelons, la transformée de Fourier n'est alors pas utilisable dans ce genre de cas. Sous certaines conditions on peut alors utiliser la Transformée de Laplace pour analyser ces signaux quelconques.

On appelle transformée de Laplace de la fonction f(t) la fonction F(p) p définie par :

$$F(p)=TL[f(t)]=\int_{0^{-}}^{\infty} f(t)e^{-pt}dt$$

Dérivation et Intégration dans le formalisme de Laplace :

$$TL[f'(t)]=P\times F(P) \quad \text{et} \quad TL[\int f(t)dt]=\frac{F(P)}{P}$$

Application à la physique

Soit une équation différentielle avec e(t) et s(t) deux fonctions dérivables n fois:

$$a_n\frac{d^n s}{dt^n}+a_{n-1}\frac{d^{n-1}s}{dt^{n-1}}+...+a_0 s=b_m\frac{d^m e}{dt^m}+b_{m-1}\frac{d^{m-1}e}{dt^{m-1}}+...+b_0 e$$

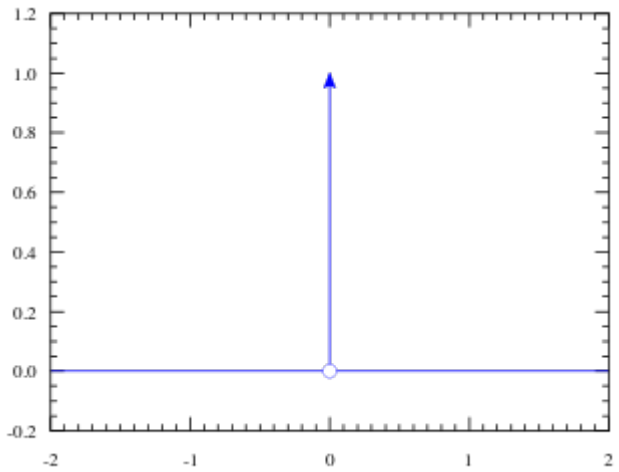
On applique la TL à l'équation et on regroupe les termes calculés en 0^- dans le polynôme $C(0^-)$. Ici les conditions initiales rendent ce polynôme nul. On a alors :

$$S(p)=\frac{[b_m p^m+b_{m-1}p^{m-1}+...+b_0]E(p)}{a_n p^n+a_{n-1}p^{n-1}+...+a_0}$$ avec S(p) et E(p) les TL de s(t) et e(t).

Pic de Dirac

On appelle impulsion de Dirac la fonction $\delta(t)$ telle que :

$$\delta(t)=\begin{pmatrix} 0 & si\ t\neq 0 \\ +\infty & pour\ t=0 \end{pmatrix}$$



et $$\int_{-\infty}^{+\infty}\delta(t)dt=1$$

L'impulsion de Dirac est alors une « impulsion infiniment fine, d'amplitude infinie et d'aire unité ». On la dessine sous la forme d'une flèche car $\delta(0) = +\infty$.
L'impulsion de Dirac est l'élément neutre du produit de convolution.

Convolution

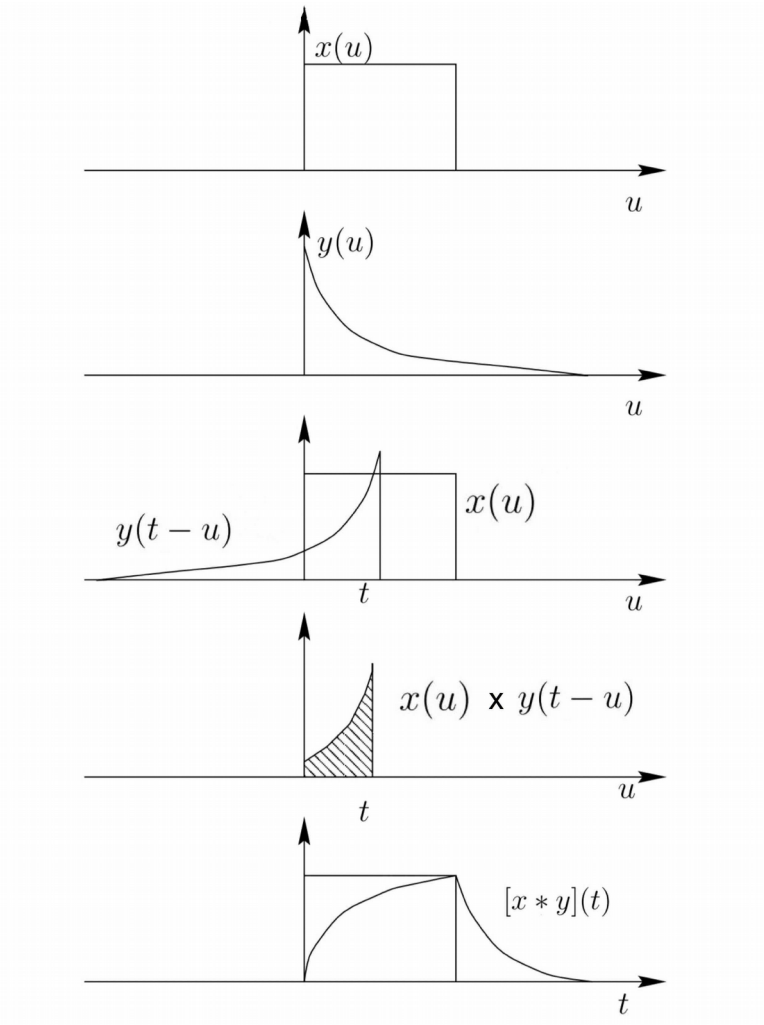
Soit y(t) la réponse globale à une entrée x(t) et h(t) la *réponse impulsionnelle (RI)* d'un système à une impulsion de Dirac :

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau = [x*h](t)$$

L'opération [x*h](t) est appelé convolution. Elle permet d'associer à une entrée x(t), une sortie y(t) du système caractérisé par sa RI h(t).

Interprétation graphique

Le produit de convolution revient à calculer la surface x(u)*y(t-u) à t donné, puis faire « glisser » y pour obtenir la convolution pour tout t. y(t-u) correspond au signal initial y(u) retourné dans le temps et translaté de t.



Réponse en fréquence

Soit un système de réponse impulsionnelle h(t) et d'entrée $X_0 e^{jw_0 t}$.

$$y(t) = \int_{-\infty}^{+\infty} X_0 e^{jw_0(t-\tau)} h(t) d\tau = X_0 e^{jw_0 t} \int_{-\infty}^{+\infty} h(t) X_0 e^{-jw_0 \tau} d\tau$$

On peut identifier la fonction de transfert ou gain complexe :

$$H(f_0) = \int_{-\infty}^{+\infty} h(t) X_0 e^{-jw_0 \tau} d\tau$$

La transformée de Fourier de la réponse impulsionnelle correspond à la fonction de transfert du filtre.

La sortie s'écrit alors simplement : $y(t) = X_0 e^{jw_0 t} \quad H(f_0)$

Théorème de Shannon

Lorsqu'un signal x(t) a un spectre à support borné ($X(f) = 0 \quad pour \quad (f) > f max$), il est possible d'échantillonner ce signal sans perdre d'information : avec une fréquence d'échantillonnage $f_e < 2f_{max}$ on pourra reconstruire x(t) à partir des $x(nT_e)$.

3.5. Partie algorithmique

Démarrage du filtre

Le programme a été mis sur une clé USB de Boot en FreeDOS, grâce au logiciel RUFUS. En plus du programme et du système d'exploitation, il y a un driver pour le support de la souris et le compilateur Turbo Pascal 7, ce qui permet d'éditer, compiler et lancer le programme directement depuis la clé USB.

Afin de démarrer le programme, il faut donc démarrer l'ordinateur sur la clé USB. On lance ensuite le fichier « tp.bat », un script qui démarre le driver de la souris et lance Turbo Pascal 7. On peut donc entrer les coefficients du filtre, puis lancer le programme et enfin choisir l'option 5 dans le menu pour démarrer le filtre. Pour l'arrêter, il faut appuyer sur une touche quelconque du clavier.

Le fonctionnement du filtre

L'ordinateur est relié à un circuit analogique grâce au port imprimante parallèle db25. Le port parallèle permet d'envoyer et recevoir de l'information. L'ordinateur commande donc les interrupteurs du circuit et reçoit les données sur 8 bits du convertisseur analogique numérique.

Tout d'abord, l'ordinateur doit recevoir un signal d'entrée. Pour cela, il bloque le signal d'entrée afin que celui-ci soit constant, grâce à un échantillonneur-bloqueur. Ensuite, l'ordinateur cherche la tension d'entrée en générant une tension partant de 0V, cette tension

est envoyée à un R2/R puis comparée avec la tension d'entrée, grâce à un comparateur. Le programme utilise la méthode d'approximations successives, en utilisant le comparateur de tension qui compare la sortie du R2/R au signal en entrée.

Une fois que l'ordinateur connaît le signal d'entrée, il le centre autour de 0V puis il calcule la tension de sortie en fonction des sorties et entrées précédentes, multipliées par certains coefficients qui dépendent du filtre voulu. Cette tension est à son tour centrée autour de 2,5V puis envoyée au R2/R pour être convertie en analogique. L'ordinateur commande à l'échantillonneur-bloqueur placé à la sortie de se bloquer sur cette nouvelle tension. Puis le processus recommence au début.

Dans le code source, on peut modifier les coefficients que l'on aura calculé préalablement pour obtenir le filtre recherché ainsi que la fréquence d'échantillonnage.

Les problèmes rencontrés

Lors des mesures de la sortie du filtre, on a remarqué des parasites, sous la forme de pics, qui apparaissent de manière aléatoire. Cela est du au fait que le programme gère tout seul les interruptions de l'horloge interne, il faut donc une connaissance du matériel sur lequel fonctionne le programme. Or ce programme n'a pas été écrit pour les chipsets et bus PCI actuels, cela crée donc ces parasites sur le signal de sortie.

3.6. Manipulations du filtre

Le programme est basé sur un filtre récursif, c'est-à-dire que la sortie s_k dépend des entrées et sorties précédentes, comme suit :

$$b_0 s_k + b_1 s_{k-1} + \ldots + b_m s_{k-m} = a_0 e_k + a_1 e_{k-1} + a_2 e_{k-2} + \ldots + a_n e_{k-n}$$

$$s_k = \frac{1}{b_0}(a_0 e_k + a_1 e_{k-1} + a_2 e_{k-2} + \ldots + a_n e_{k-n} - b_1 s_{k-1} - \ldots - b_m s_{k-m})$$

Cette équation est dite équation aux différences finies. On applique la transformée en Z tel que :

$$w_k = W(Z), w_{k-n} = W(Z)Z^{-n}$$

On a alors :

$$b_0 S(Z) + b_1 S(Z)Z^{-1} + \ldots + b_m S(Z)Z^{-m} = a_0 E(Z) + a_1 E(Z)Z^{-1} + a_2 E(Z)Z^{-2} + \ldots + a_n E(Z)Z^{-n}$$

$$\frac{S(Z)}{E(Z)} = \frac{a_0 + a_1 Z^{-1} + a_2 Z^{-2} + \ldots + a_n Z^{-n}}{b_0 + b_1 Z^{-1} + b_2 Z^{-2} + \ldots + b_m Z^{-m}} = H_n(Z)$$

On connaît l'écriture des filtres avec leur modèle analogique. On a donc $H_a(p)$. Pour connaître $H_n(Z)$, il faut donc connaître la variable p en fonction de Z. Or pour un circuit intégrateur, on a :

$$S(p) = \frac{E(p)}{p}$$
$$\frac{S(p)}{E(p)} = \frac{1}{p} = H_a(p)$$

Or on a :

$$H_a(p) = H_n(Z)$$

Il faut donc trouver $H_n(Z)$ pour trouver le rapport entre p et Z. Or, en fonction de la méthode d'intégration, on a une formule récursive, qui nous permettra d'obtenir $H_n(Z)$ et donc p en fonction de Z.

3.6.1. Méthode des trapèzes

Pour un intégrateur numérique au sens des trapèzes, on a :

$$s_k = s_{k-1} + \frac{e_k + e_{k-1}}{2} T_e \Rightarrow S(Z) = Z^{-1}S(Z) + \frac{E(Z) + Z^{-1}E(Z)}{2} T_e$$
$$\Rightarrow \frac{S(Z)}{E(Z)} = \frac{T_e}{2} \frac{1+Z^{-1}}{1-Z^{-1}}$$

On peut donc calculer p en fonction de Z :

$$\frac{S(Z)}{E(Z)} = \frac{T_e}{2} \frac{1+Z^{-1}}{1-Z^{-1}} = H_n(Z) = H_a(p) = \frac{1}{p}$$
$$\Rightarrow p = \frac{2}{T_e} \frac{1-Z^{-1}}{1+Z^{-1}}$$

Cherchons à créer un filtre quelconque avec le filtre numérique. Il nous faut donc connaître les coefficients α_k et β_k tels que :

$$s_k = \alpha_0 e_k + \alpha_1 e_{k-1} + \alpha_2 e_{k-2} + \ldots + \alpha_n e_{k-n} + \beta_1 s_{k-1} + \ldots + \beta_m s_{k-m}$$
$$\Rightarrow S(Z) = \alpha_0 E(Z) + \alpha_1 E(Z) Z^{-1} + \alpha_2 E(Z) Z^{-2} + \ldots + \alpha_n E(Z) Z^{-n} + \beta_1 S(Z) Z^{-1} + \ldots + \beta_m S(Z) Z^{-m}$$

Filtre passe-bas

La fonction de transfert d'un filtre passe-bas est :

$$H_a(p) = \frac{1}{1+\tau p}$$

On remplace p par Z :

$$H_a(p) = \frac{1}{1+\tau p}$$

$$\Rightarrow H_n(Z) = \frac{1}{1+\tau\frac{2(1-Z^{-1})}{T_e(1+Z^{-1})}} = \frac{1}{\frac{T_e(1+Z^{-1})+2\tau(1-Z^{-1})}{T_e(1+Z^{-1})}}$$

$$\Rightarrow \frac{T_e+T_eZ^{-1}}{T_e+2\tau+Z^{-1}(T_e-2\tau)}$$

Puis on cherche les coefficients :

$$H_n(Z) = \frac{S(Z)}{E(Z)} = \frac{T_e+T_eZ^{-1}}{T_e+2\tau+Z^{-1}(T_e-2\tau)}$$

$$\Rightarrow S(Z)(T_e+2\tau+Z^{-1}(T_e-2\tau)) = E(Z)(T_e+T_eZ^{-1})$$

$$\Rightarrow (T_e+2\tau)S(Z) = T_eE(Z)+T_eE(Z)Z^{-1}-(T_e-2\tau)S(Z)Z^{-1}$$

$$\Rightarrow (T_e+2\tau)s_k = T_ee_k+T_ee_{k-1}-(T_e-2\tau)s_{k-1}$$

Posons :

$$(T_e+2\tau)s_k = T_ee_k+T_ee_{k-1}-(T_e-2\tau)s_{k-1}$$
$$b_0 = T_e+2\tau$$
$$b_1 = -(T_e-2\tau)$$
$$a_0 = T_e = a_1$$

Ainsi, on a :

$$b_0s_k = a_0e_k+a_1e_{k-1}+b_1s_{k-1}$$

$$\Rightarrow s_k = \frac{a_0}{b_0}e_k+\frac{a_1}{b_0}e_{k-1}+\frac{b_1}{b_0}s_{k-1}$$

$$\Rightarrow s_k = \alpha_0e_k+\alpha_1e_{k-1}+\beta_1s_{k-1}$$

On peut donc calculer les valeurs numériques et entrer les coefficients α_k et β_k dans le programme.

$$\alpha_0 = \frac{a_0}{b_0}$$

$$\alpha_1 = \frac{a_1}{b_0}$$

$$\beta_1 = \frac{b_1}{b_0}$$

Filtre passe-bande

On applique les mêmes étapes que pour le filtre passe-bas. Nous ne détaillerons donc pas les calculs.

On cherche $H_n(Z)$:

$$H_a(p) = \frac{2\alpha_a \omega_0 p}{\omega_0^2 + 2\alpha_a \omega_0 p + p^2}$$

$$p = \frac{2}{T_e}\frac{1 - Z^{-1}}{1 + Z^{-1}}$$

$$H_n(Z) = \frac{a_0 + a_2 Z^{-2}}{b_0 + b_1 Z^{-1} + b_2 Z^{-2}}$$

α_a est le coefficient d'amortissement. On a les coefficients suivants :

$$a_0 = 4\alpha_a \omega_0 T_e$$
$$a_1 = 0$$
$$a_2 = -a_0$$
$$b_0 = \omega_0^2 T_e^2 + 4\alpha_a \omega_0 T_e + 4$$
$$b_1 = 2\omega_0^2 T_e^2 - 8$$
$$b_2 = \omega_0^2 T_e^2 - 4\alpha_a \omega_0 T_e + 4$$

On obtient alors les coefficients α_k et β_k suivants :

$$\alpha_0 = \frac{a_0}{b_0}$$
$$\alpha_1 = 0$$
$$\alpha_2 = -\alpha_0$$
$$\beta_1 = \frac{-b_1}{b_0}$$
$$\beta_2 = \frac{-b_2}{b_0}$$

3.6.2. Méthode de Simpson

Pour un intégrateur numérique au sens de Simpson, on a

$$s_k = s_{k-1} + \frac{T_e}{3}(e_{k-2} + 4e_{k-1} + e_k) \Rightarrow S(Z) = Z^{-1}S(Z) + \frac{T_e}{3}(E(Z)Z^{-2} + 4E(Z)Z^{-1} + E(Z))$$

$$\Rightarrow \frac{S(Z)}{E(Z)} = \frac{T_e}{3(1-Z^{-1})}(Z^{-2} + 4Z^{-1} + 1) = \frac{1}{p}$$

$$\Rightarrow p = \frac{3}{T_e}\frac{1-Z^{-1}}{Z^{-2} + 4Z^{-1} + 1}$$

Filtre passe-bas

La fonction de transfert d'un filtre passe-bas est :

$$H_a(p) = \frac{1}{1+\tau p}$$

On remplace p par Z :

$$H_a(p) = \frac{1}{1+\tau p}$$

$$\Rightarrow H_n(Z) = \frac{1}{1+\tau\frac{3(1-Z^{-1})}{T_e(1+4Z^{-1}+Z^{-2})}} = \frac{1}{\frac{T_eZ^{-2} + 4T_eZ^{-1} + T_e + 3\tau - 3\tau Z^{-1}}{TeZ^{-2} + 4T_eZ^{-1} + T_e}}$$

$$\Rightarrow \frac{T_e + Z^{-1}(4T_e) + Z^{-2}(T_e)}{(T_e + 3\tau) + Z^{-1}(4T_e - 3\tau) + Z^{-2}(T_e)} = \frac{S(Z)}{E(Z)}$$

Posons :

$$a_0 = T_e$$
$$a_1 = 4T_e = 4a_0$$
$$a_2 = T_e = a_0$$
$$b_0 = T_e + 3\tau$$
$$b_1 = 4T_e - 3\tau$$
$$b_2 = T_e$$

On a ainsi :

$$\frac{a_0 + Z^{-1}a_1 + Z^{-2}a_2}{b_0 + Z^{-1}b_1 + Z^{-2}b_2} = \frac{S(Z)}{E(Z)}$$
$$\Rightarrow S(Z)b_0 + S(Z)Z^{-1}b_1 + S(Z)Z^{-2}b_2 = a_0E(Z) + E(Z)Z^{-1}a_1 + E(Z)Z^{-2}a_2$$
$$\Rightarrow s_k b_0 + s_{k-1}b_1 + s_{k-2}b_2 = a_0e_k + e_{k-1}a_1 + e_{k-2}a_2$$
$$\Rightarrow s_k = \frac{a_0}{b_0}e_k + \frac{a_1}{b_0}e_{k-1} + \frac{a_2}{b_0}e_{k-2} - \frac{b_1}{b_0}s_{k-1} - \frac{b_2}{b_0}s_{k-2}$$

On pose :

$$\alpha_0 = \frac{a_0}{b_0}$$
$$\alpha_1 = 4\alpha_0$$
$$\alpha_2 = \alpha_0$$
$$\beta_1 = \frac{-b_1}{b_0}$$
$$\beta_2 = \frac{-b_2}{b_0}$$

Il suffit de calculer les valeurs numériques de ces coefficients et les entrer dans le code source afin d'avoir un filtre passe-bas utilisant Simpson.

Filtre passe-bande

La fonction de transfert d'un filtre passe-bande est :

$$H_a(p) = \frac{2\alpha_a\omega_0 p}{\omega_0{}^2 + 2\alpha_a\omega_0 p + p^2}$$

avec α_a le coefficient d'amortissement. Pour faciliter les calculs, on pose :

$$\lambda = 2\alpha_a\omega_0$$

On a alors :

$$H_a(p) = \frac{\lambda}{\frac{1}{p}\omega_0^2 + \lambda + p}$$

$$\Rightarrow H_n(Z) = \frac{\lambda}{\frac{T_e(Z^{-2}+4Z^{-1}+1)}{3(1-Z^{-1})}\omega_0^2 + \lambda + \frac{3(1-Z^{-1})}{T_e(Z^{-2}+4Z^{-1}+1)}}$$

$$= \frac{\lambda}{\frac{T_e^2(Z^{-2}+4Z^{-1}+1)^2\omega_0^2 + \lambda[3(1-Z^{-1})T_e(Z^{-2}+4Z^{-1}+1)] + 3^2(1-Z^{-1})^2}{3(1-Z^{-1})T_e(Z^{-2}+4Z^{-1}+1)}}$$

$$= \frac{3\lambda T_e(Z^{-2}+4Z^{-1}+1-Z^{-3}-4Z^{-2}-Z^{-1})}{T_e^2\omega_0^2(Z^{-2}+4Z^{-1}+1)^2 + 3\lambda T_e(-3Z^{-2}-Z^{-3}+3Z^{-1}+1) + 9(1-2Z^{-1}+Z^{-2})}$$

$$= \frac{3\lambda T_e + Z^{-1}(9\lambda T_e) + Z^{-2}(-9\lambda T_e) + Z^{-3}(-3\lambda T_e)}{(T_e^2\omega_0^2 + 3\lambda T_e + 9) + Z^{-1}(8T_e^2\omega_0^2 + 9\lambda T_e - 18) + Z^{-2}(18T_e^2\omega_0^2 - 9\lambda T_e + 9) + Z^{-3}(8T_e^2\omega_0^2 - 3\lambda T_e) + Z^{-4}(T_e^2\omega_0^2)}$$

et :

$$\frac{a_0 + Z^{-1}a_1 + Z^{-2}a_2 + Z^{-3}a_3}{b_0 + Z^{-1}b_1 + Z^{-2}b_2 + Z^{-3}b_3 + Z^{-4}b_4} = \frac{S(Z)}{E(Z)}$$

Donc on en déduit :

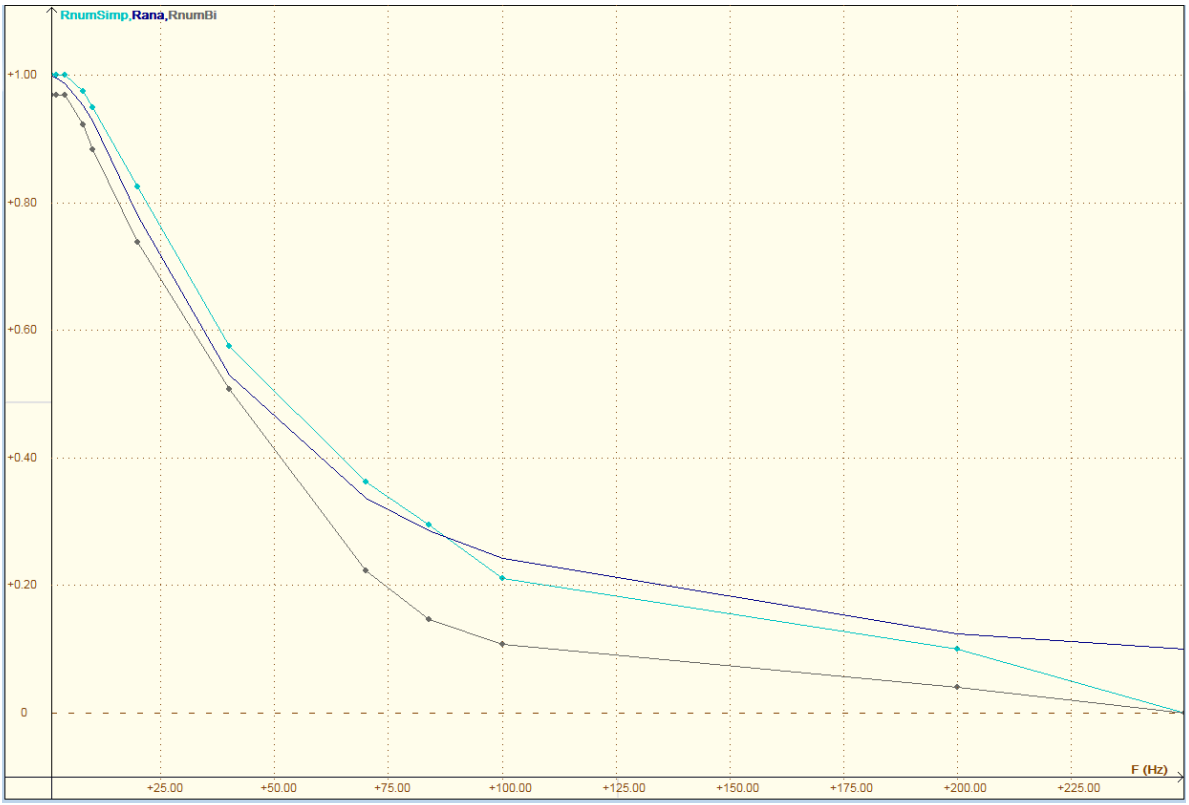
$$a_0 = 3\lambda T_e$$
$$a_1 = 9\lambda T_e = 3a_0$$
$$a_2 = -9\lambda T = -3a_0$$
$$a_3 = -3\lambda T_e = a_0$$
$$b_0 = T_e^2\omega_0^2 + 3\lambda T_e + 9$$
$$b_1 = 8T_e^2\omega_0^2 + 9\lambda T_e - 18$$
$$b_2 = 18T_e^2\omega_0^2 - 9\lambda T_e + 9$$
$$b_3 = 8T_e^2\omega_0^2 - 3\lambda T_e$$
$$b_4 = T_e^2\omega_0^2$$

Et de la même manière que pour les filtres précédents, on obtient les coefficients α_k et β_k :

$$\alpha_0 = \frac{a_0}{b_0}$$
$$\alpha_1 = 3\alpha_0$$
$$\alpha_2 = -3\alpha_0$$
$$\alpha_3 = -\alpha_0$$
$$\beta_1 = \frac{-b_1}{b_0}$$
$$\beta_2 = \frac{-b_2}{b_0}$$
$$\beta_3 = \frac{-b_3}{b_0}$$
$$\beta_4 = \frac{-b_4}{b_0}$$

On remarque qu'avec la méthode de Simpson, on va jusqu'à la 4è sortie précédente pour calculer la sortie actuelle.

Courbes expérimentales pour le filtre passe-bas :



Filtre passe-bas théorique (parfait)

Filtre passe-bas avec méthode des trapèzes

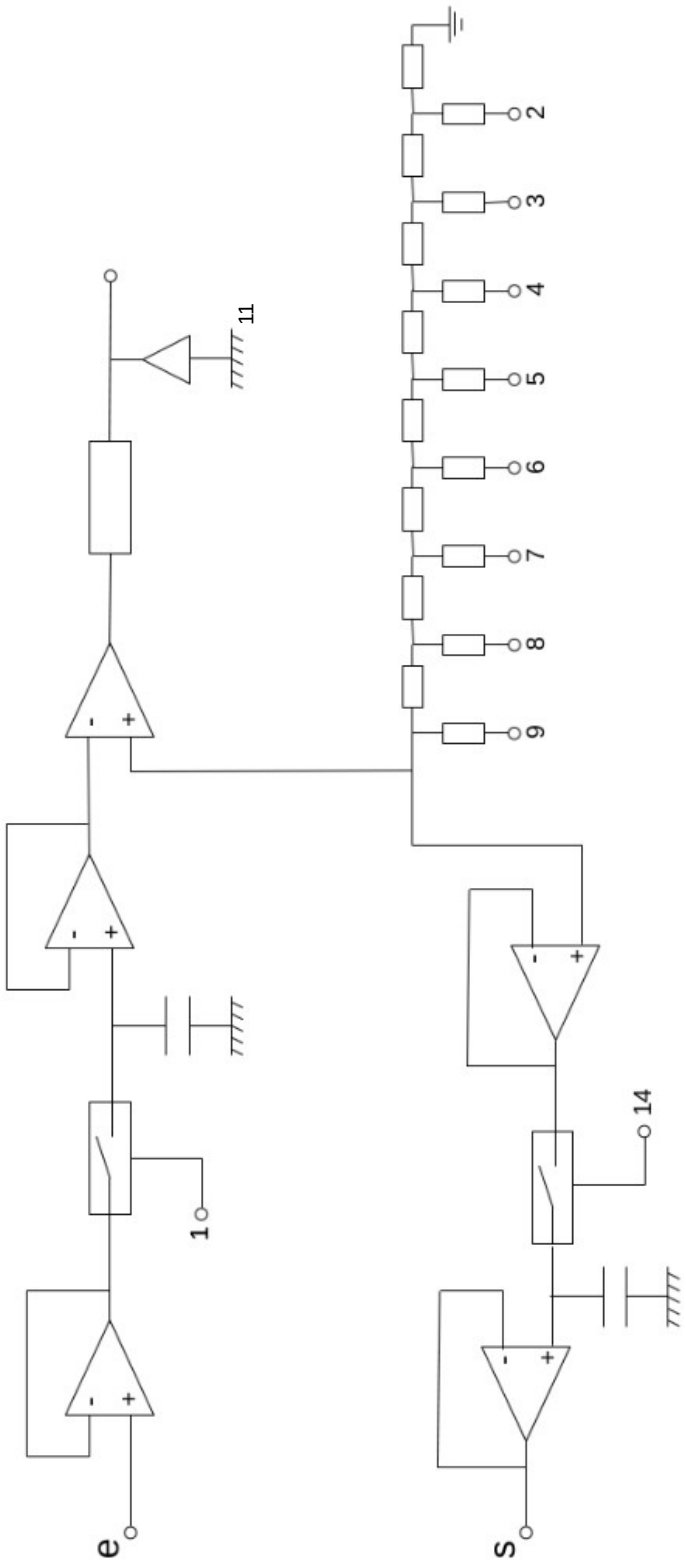
Filtre passe-bas avec méthode de Simpson

4. CONCLUSIONS ET PERSPECTIVES

Les courbes expérimentales et les formules théoriques montrent que la méthode de Simpson donne une meilleure approximation que celle des trapèzes. Le résultat reste cependant légèrement différent du filtre analogique, le filtre numérique étant limité par la fréquence d'échantillonnage et la chute à 0 à $f_e/2$. En améliorant la méthode d'approximation, on peut arriver un résultat très proche du filtre théorique. De plus, le filtrage numérique permet d'utiliser tous les filtres possibles (passe-bas, passe-bandes, etc..) sans modifier le montage. Cela permet de démocratiser l'utilisation du filtre et permet à un public non familier au filtrage de l'utiliser simplement.

5. ANNEXES

5.1. Schéma électrique



5.2. Programme implémenté dans l'ordinateur

```
{
Projet P6 mai 2017 / filtre numérique /  fichier filtre2.pas
Francois Guillotin / 1er cycle stpi
INSA de Rouen

Configuration matérielle: ordinateur ASUS pentium e2160 ; port imprimante parallèle db25
réseau R2R (branchement direct sur le DB25); un comparateur simple; deux fonctions échantillonneur-bloqueur

le programme ci-dessous contient:

des procédures d'entrée, de sortie et de test
le cadencement par interruption avec l'horloge interne PIT 8353
une procédure, pour un filtre iir récursif, appelée par interruption

lancement de l'ordinateur par une clé USB de Boot en FreeDOS
(création de la clé (2Go) par le logiciel RUFUS)
sur la clé il y a le compilateur Turbo Pascal 7 (avec un fichier TPL fixant le bug "200")
les coefficients du filtre sont entrés directement dans le listing du programme
avec une exécution lancée à partir de l'environnement IDE du Turbo Pascal

remarque: ce programme est basique et incomplet
il est  basé sur 8 lignes d'IRQ
il faudrait revoir la gestion des interruptions (masquage notamment)
pour tenir compte des chipsets et des bus PCI actuels.

en effet , il apparaît de façon hiératique et fugitive des pics parasitant le signal en sortie
mais cela ne semble pas brouiller les calculs du filtre,
et les valeurs en sorties sont correctes hors les pics.

au lancement de l'ordinateur, on appuie sur la touche F8
pour avoir le menu de BOOT
(le disque dur ne sert à rien il pourrait être enlevé)

un fois le DOS installé, on lance la commande TP
(fichier tp.bat pour la mise en place du driver de la souris et le lancement du turbo pascal)
```

après modification des coefficients dans le listing du programme

on lance celui-ci par la commande RUN (menu de l'IDE)

dans le programme:

on choisit l'option 5 dans le menu (entrer 5 et return)

l'arrêt du filtre se fait par un appui sur une touche quelconque du clavier

option 7 permet l'arrêt du programme et le retour au listing dans l'IDE

remarque:

dans le listing les commentaires sont entre accolades { }

(*ou entre (* *)

```
(************* zone des declarations **********************)
uses dos,crt;

const reg_out=$378; {lignes de sortie de données}
      reg_in=$379; {lignes d'entrée }
      reg_com=$37A;{registre de commande}
      decal:array[0..7] of byte=(1,2,4,8,16,32,64,128);

type tab=array[0..7] of byte;

var val_e,val_s:byte;
    tab_e,tab_s:tab;


procedure conv_bin(valeur:byte;var tabbin:tab);
var dividende,i:integer;
begin
dividende:=valeur;
for i:=0 to 7
do begin
   tabbin[i]:= dividende mod 2;
   dividende:=dividende div 2;
   end;
end;

procedure conv_oct(tabbin:tab;var valeur:byte);
var i:byte;
```

```
begin
valeur:=0;
for i:=0 to 7 do valeur:=valeur+tabbin[i]*(1 shl i);
end;



(***** gestion du TIMER et de l'interruption associee  **********************)

var masque:word;
    vecteuravant:pointer;
    Regs:Registers;

{les valeurs, comme $40, sont des adresses particulières dans la mémoire de l'ordinateur
correspondant au circuit intégré PIT 8353 gérant l'horloge interne
il faudrait revoir cela avec les chipsets actuels}
procedure Timerfreq(Freq:real);
begin
 asm cli end;

 Port[$43] := $36;
 Port[$40] := Lo(round(1193181.7/Freq));
 Port[$40] := Hi(round(1193181.7/Freq));

 asm sti end;
end;



procedure init_irq0(addr_proc:pointer);
begin
{ bloque les interruptions; sauve le vecteur initial;
  place le nouveau vecteur (c'est une adresse dans la mémoire de l'ordinateur)
  valide l'entree IRQ0 (Timer)  dans le 8259 registre IMR
  détourne l'IRQ0 de $08 en $80 (zone libre !)
  autorise les interruptions
  la directive ASM permet de mettre des lignes de code en assembleur (langage machine)
  directement dans le code source turbo pascal
  cli (clear interrupt) et sti (set interrupt)
  sont des codes pour le microprocesseur Intel}
asm cli end;
```

```
getintvec($08,vecteuravant);
setintvec($08,addr_proc);

masque:=port[$21];
port[$21]:=$Fc;{valide les entrees IRQ0 et IRQ1(clavier!) dans le 8259 registre IMR}
{masquages possibles d'interruptions:
port[$21]:=$EE;com1(souris) et timer autorises
port[$21]:=$Fc;valide les entrees IRQ0 et IRQ1(clavier!) dans le 8259 registre IMR}

setintvec($80,vecteuravant);
asm sti end;
end;



procedure restaure_timer;
begin
asm cli end;
 Port[$43] := $36;
 Port[$40] := $FF;
 Port[$40] := $FF;
asm sti end;
end;

procedure restaure_int08;
begin
asm cli end;
port[$21]:=masque;
setintvec($08,vecteuravant);
asm sti end;
end;

{*********************************************************}
procedure test_sorties;
var i,octet:byte;
begin
clrscr;
gotoxy(1,12);write('donnez une valeur a sortir:');
readln(val_s);
port[reg_com]:=$0f;
```

```
port[reg_out]:=val_s;
conv_bin(val_s,tab_s);
gotoxy(1,20);
for i:=7 downto 0 do write(tab_s[i]:3);readln;
end;


procedure test_entrees;
var i,octet:byte;
    fin:boolean;
    touche:string;
begin
clrscr;
if keypressed then touche:=readkey;
fin:=false;

gotoxy(1,12);write('lecture des entrees:');
repeat
val_e:=port[reg_in];
gotoxy(1,18);write('valeur en entree:',val_e:4,' avant conversion');
conv_bin(val_e,tab_e);
gotoxy(1,20);
for i:=7 downto 0 do write(tab_e[i]:3);
if keypressed then begin touche:=readkey;fin:=true end;
until fin;
end;


procedure rampe;
var i:byte;
    fin:boolean;
    touche:string;

begin
clrscr;
gotoxy(1,12);
write('sortie d''une rampe:');
write('  fin par appuie sur une touche');
```

```
if keypressed then touche:=readkey;
fin:=false;
port[reg_com]:=$0f;

i:=0;
repeat
port[reg_out]:=i;
delay(2);
if i=255 then i:=0 else i:=i+1;
if keypressed then begin touche:=readkey;fin:=true end;
until fin;

end;



procedure bloque_e;
var compteur:integer;
    test:real;
begin
{échantillonnage et blocage de la tension d'entrée}
port[$37A]:=$0f;
port[$37A]:=$0e;{impulsion broche 14 /autofeed}
for compteur:=0 to 500 do test:=45.25;{boucle bidon d'attente}
port[$37A]:=$0f;
end;

procedure sortie_bloquee(valeur:byte);
var compteur:integer;
    test:real;
begin
{envoi et blocage de la tension de sortie}
port[$37A]:=$0f;
port[$378]:=valeur;
port[$37A]:=$0d;{impulsion broche 1 /strobe}
for compteur:=0 to 500 do test:=45.25; {boucle bidon d'attente}
port[$37A]:=$0f;
end;

function approx_entree:byte;
```

```
var vale:byte;
   compteur,tempo:word;
   test:real;
begin
tempo:=2500;
{force le registre reg_out en sortie si ECP, et laisse les bloqueurs fermés}
port[$37A]:=$0f;

{conversion analogique numérique par approximations successives}
vale:=0;
port[$378]:=128;
for compteur:=0 to tempo do test:=45.25; {boucle bidon d'attente}
vale:=port[$379] and 128;

port[$378]:=vale+64;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 1);

port[$378]:=vale+32;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 2);

port[$378]:=vale+16;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 3);

port[$378]:=vale+8;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 4);

port[$378]:=vale+4;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 5);

port[$378]:=vale+2;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 6);
```

```
port[$378]:=vale+1;
for compteur:=0 to tempo do test:=45.25;
vale:=vale+((port[$379] and 128) shr 7);
{fin conversion }

approx_entree:=vale;
end;



procedure mesure_entree;
var mesure:real;
   touche:string;
   finapp:boolean;
begin
clrscr;gotoxy(1,12); write('hello');
finapp:=false;

repeat
bloque_e;
val_e:=approx_entree;
mesure:=val_e*0.0131111;{coefficient fonction de la valeur en volt du niveau haut des sorties sur la DB25}
gotoxy(1,14);writeln(mesure:6:3);
if keypressed then begin touche:=readkey;finapp:=true; end;
until finapp;

end;

(** test de l'interruption par l'horloge interne **************************)

var itest:byte;

{$F+} {si il faut faire des appels longs en mémoire, ?}
procedure int_prog_test;
interrupt;
begin
    asm cli end;
    if itest=255 then itest:=0 else itest:=itest+1;
    asm sti end;
    port[$20]:=$20;{indique au circuit 8259 la fin du traitement de l'interruption }
```

```
    intr($80,regs);
end;
{$F-}

var  frequence:real;

procedure test_interruption;
var touche_clav:string;
   finapp:boolean;
begin
itest:=0;
clrscr;
if keypressed then touche_clav:=readkey;
finapp:=false;
frequence:=10;{test à 10Hz}
timerfreq(frequence);
init_irq0(@ int_prog_test);
repeat
        gotoxy(1,14);writeln('   ',itest,'    ');
        if keypressed then begin touche_clav:=readkey;finapp:=true; end;
until finapp;
restaure_int08;
restaure_timer;
end;

(** filtre recursif ***************************)
var e,s:array[-10..10] of real;
   a0,a1,a2,a3,a4,b1,b2,b3:real;
   i:integer;
procedure int_filtre;
interrupt;
begin
    asm cli end;
    bloque_e;
{la tension d'entrée est centrée sur 2.5V car le R2R fonctionne entre 0 et 5v environ
avant les calculs on recentre l'entrée sur 0
après les calculs on écrête le résultat
et on sort la valeur recentrée sur 128 (environ 2.5 volt)}
    e[0]:=approx_entree-128;{centrage au 0}
```

```
    s[0]:=a0*e[0]+a1*e[-1]+a2*e[-2]+a3*e[-3]+a4*e[-4]+b1*s[-1]+b2*s[-2]+b3*s[-3];
    if s[0]<-128 then s[0]:=-128; {écrêtage inférieur du résultat}
    if s[0]>127 then s[0]:=127; {écrêtage supérieur du résultat}
    sortie_bloquee(round(s[0])+128); {sortie de la valeur centrée sur 128}
    s[-3]:=s[-2];s[-2]:=s[-1];s[-1]:=s[0];
    e[-4]:=e[-3];e[-3]:=e[-2];e[-2]:=e[-1];e[-1]:=e[0];
    asm sti end;
    port[$20]:=$20;
    intr($80,regs);
end;

procedure filtre_iir;
var touche_clav:string;
   finapp:boolean;
begin
{ici, on place les coefficients du filtre, exemple avec le passe bande et la transformation bilinéaire }
a0:=0.054094;
a1:=0;
a2:=-0.054094;
a3:=0;
a4:=0;
b1:=1.551928;
b2:=-0.891812;
b3:=0;
clrscr;
if keypressed then touche_clav:=readkey;
finapp:=false;
for i:=-10 to 10 do begin e[i]:=0;s[i]:=0 end;
{on fixe la fréquence de l'interruption IRQ0 de l'horloge interne}
frequence:=500;
timerfreq(frequence);
init_irq0(@ int_filtre); {lancement du filtre}
repeat   {boucle d'attente , on ne fait rien c'est l'horloge interne qui appelle la procédure du filtre}
      if keypressed then begin touche_clav:=readkey;finapp:=true; end;
until finapp;
{remise en place des éléments du système}
restaure_int08;
restaure_timer;
end;
```

```
{*****************programme*************}
var fin:boolean;
   touche:char;
   choix:integer;
begin
if keypressed then touche:=readkey;
fin:=false;
port[reg_com]:=$0f;
port[reg_out]:=0;
clrscr;

repeat
     choix:=0;
     clrscr;
     gotoxy(1,5);write('1:ecriture sur les sorties');
     gotoxy(1,7);write('2:lecture des entrees');
     gotoxy(1,9);write('3:envoi d''une rampe');
     gotoxy(1,11);write('4:mesure par approximations successives');
     gotoxy(1,13);write('5:filtre iir');
     gotoxy(1,15);write('6:test comptage par interruption horloge interne');
     gotoxy(1,17);write('7:fin programme');
     gotoxy(1,21);write('donnez votre choix: ');

     readln(choix);
     case choix of
     1:test_sorties;
     2:test_entrees;
     3:rampe;
     4:mesure_entree;
     5:filtre_iir;
     6:test_interruption;
     7:fin:=true;
     end;
until fin;
end.
```