Électronique numérique

J.-M Friedt

FEMTO-ST/département temps-fréquence

jmfriedt@femto-st.fr

transparents à `jmfriedt.free.fr`

12 janvier 2017

Plan des interventions :

7 cours/TP (séances de 4 h) :

1. Exploitation d'un signal échantillonné en temps discret
   fréquence d'échantillonnage et filtrage
2. Quantification des coefficients, risques de dépassement de capacité.
   Conception et mise en œuvre d'un filtre FIR
3. Environnements exécutifs : FreeRTOS, RTEMS, Nuttx
4. FreeRTOS, RTEMS, Nuttx
   Multitâche et méthodes associées pour garantir l'intégrité des données partagées
5. Système embarqué sous GNU/Linux – système d'exploitation compatible POSIX
   Architecture d'un système d'exploitation, rôle du noyau
   Connectivité internet et configuration réseau
6. Accès aux ressources matérielles depuis l'espace utilisateur (MMU) - - `/dev/mem`
7. Accès aux ressources matérielles depuis un serveur

Pilote noyau de type caractère (*char device*) ?

Cas des interruptions

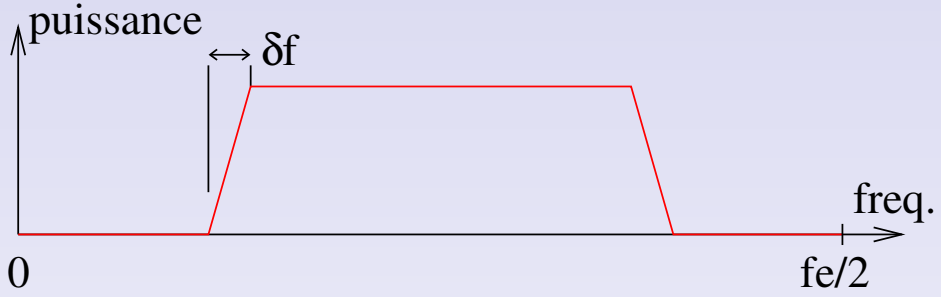
1. Une interruption est un méthode pour interrompre l'exécution séquentielle d'un programme lorsqu'un évènement qui ne peut pas attendre survient
2. Plusieurs sources d'interruptions disponibles sur microcontrôleur (GPIO, timer, communication) ...
3. ... selon les architectures, un gestionnaire d'interruption teste la source, ou plusieurs gestionnaires.
4. Un gestionnaire d'interruption doit être le plus bref possible : définir un drapeau et quitter
5. Passage de paramètres : variable globale

Variables globales

```
volatile int global_index , tim0 , heure , minute , seconde ;

void IRQ_Handler (void)          // unique gestionnaire pour ADuC7026
{ if (IRQSIG & UART_BIT)
    {                            // UART Interrupt
      global_tab[global_index] = my_getchar ();
      if (global_index < (NB_CHARS - 1))
        global_index++;
    }
  if (IRQSIG & GP_TIMER_BIT)
    {                            // Timer1 Interrupt
      GLOBAL_TIMER1++;
      T1CLRI = 1;                // Clear Timer 1 interrupt
    }
  if (IRQSIG & RTOS_TIMER_BIT)
    { tim0++;                    // Timer0 Interrupt
      seconde++;
      if (seconde > 600)
        {seconde = 0;minute++;} // seconde en 1/10 seconde
      if (minute > 60) {minute = 0;heure++;}
      if (heure > 24)  {heure = 0;}
      T0CLRI = 0;
    }
}
```

Pourquoi faut-il échantillonner périodiquement ?

- Tout traitement numérique du signal échantillonné en temps discret est basé sur l'hypothèse d'un échantillonnage périodique



- exemple d'un filtre :

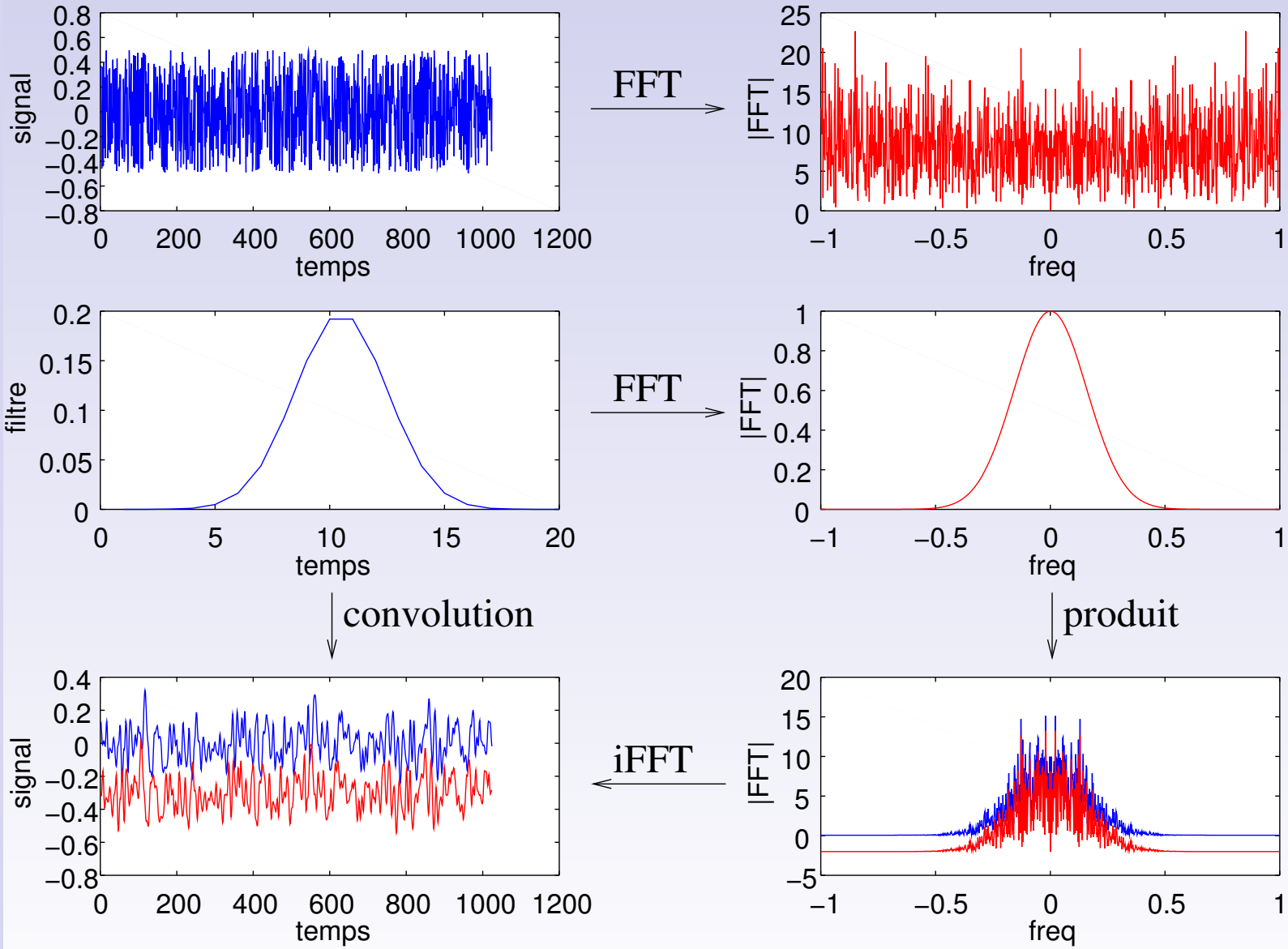
un FIR est défini par une convolution (combinaison linéaire des entrées) :

$$y_n = \sum_{k=1..m} b_k \cdot x_{n-k}$$

$\Rightarrow$ il *faut* attendre m données pour enclencher le filtre

- attendre = déphasage
- déphasage = point critique d'un asservissement qui peut rendre le système instable (oscillation)

FIR



Approche temporelle v.s fréquentielle [1]

1. http://www.trondeau.com/examples/2010/9/12/basic-filtering.html

FIR

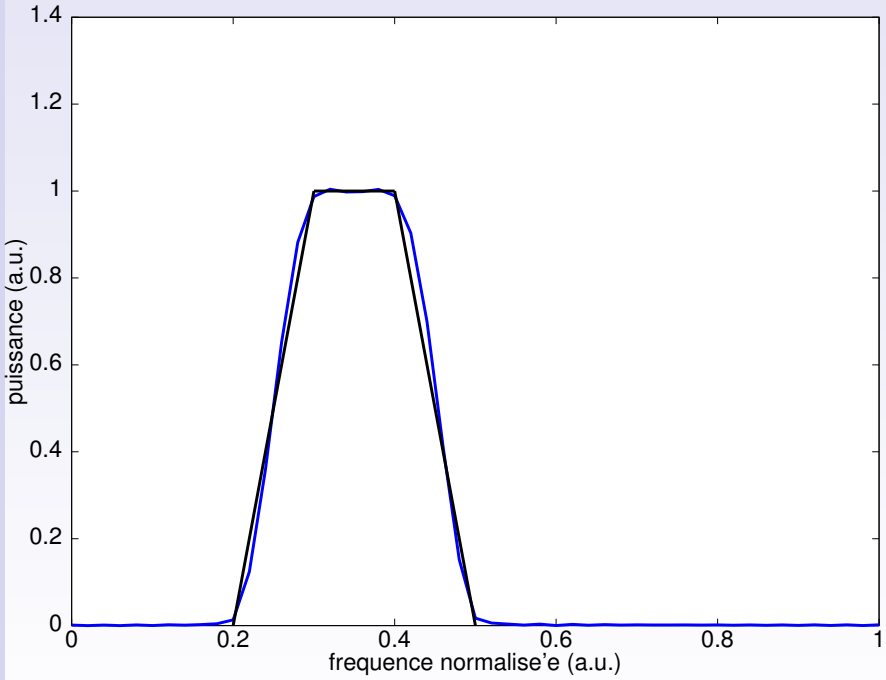
```
x=rand(1024,1);x=x-mean(x);     % signal : valeur moyenne=0
subplot(321); plot(x);xlabel('temps');ylabel('signal')
t=linspace(-10,10,20);
y=exp(-t.^2/9);y=y/sum(y);      % filtre
subplot(323);plot(y);xlabel('temps');ylabel('filtre')
f=linspace(-1,1,1024);

yc=conv(x,y);yc=yc(10:end-10); % signal filtre' (conv)
subplot(325);plot(yc);xlabel('temps');ylabel('signal')

subplot(322);
plot(f,abs(fftshift(fft(x))),'r')  % spectre du signal
subplot(324);
plot(f,abs(fftshift(fft(y,1024))),'r') % spectre du filtre
subplot(326);
plot(f,abs(fftshift(fft(yc))))     % spectre signal filtre'
hold on
yyc=fft(x,1024).*fft(y,1024)';     % produit des spectres
plot(f,abs(fftshift(yyc))-2,'r')

subplot(325);hold on;
plot(real(ifft(yyc))-0.3,'r')      % retour dans le temps
```

Exercice : refaire avec une fenêtre rectangulaire

FIR v.s IIR

1. FIR : $y_n = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées, pas de risque de divregence, mais long à converger
2. IIR : $\sum a_k y_{n-k} = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées *et des sorties passées*, risque de divregence, mais convergence rapide
3. x_k est mesuré par le matériel : résolution imposée
4. combien de bits sur les cofficients b_k pour garantir le résultat de la sortie ?

Exemple de coefficients représentés en nombres à virgule flottante



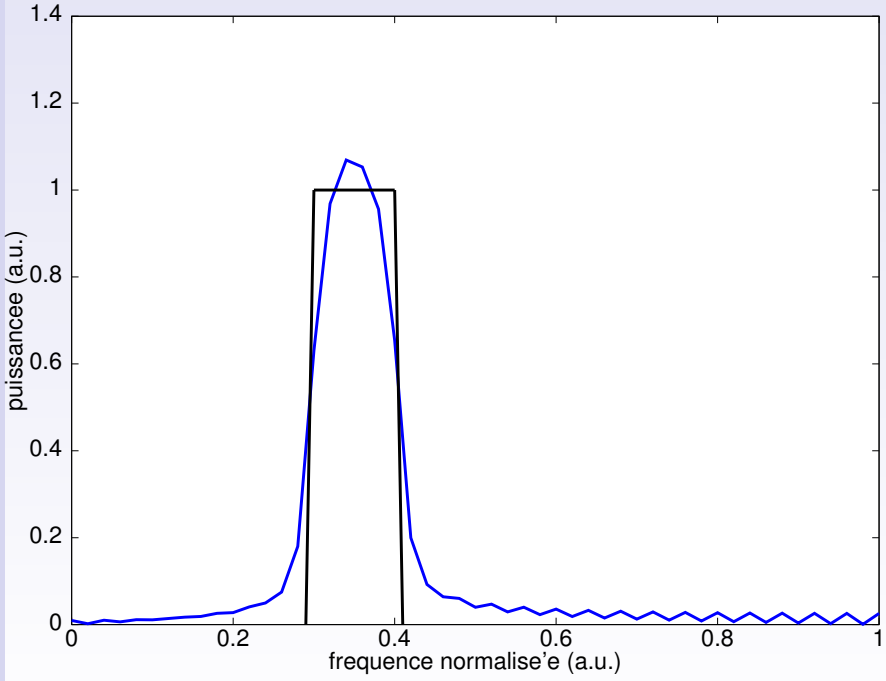
1. Fonction `firls` de GNU/Octave (ou Matlab) pour fournir les fréquences normalisées, les coefficients de pondération, et le nombre de coefficients `a_k`
2. $y_n = \sum_1 \frac{a_k}{a_0} y_{n-k} - \sum_0 \frac{b_k}{a_0} x_{n-k}$ ou en transformée en Z :

$$H(z) = \frac{\sum_0 b_{k+1}/a_0 \cdot z^{-k}}{1 + \sum_1 a_k/a_0 \cdot z^{-k}}$$

FIR v.s IIR

1. FIR : $y_n = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées, pas de risque de divregence, mais long à converger
2. IIR : $\sum a_k y_{n-k} = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées *et des sorties passées*, risque de divregence, mais convergence rapide
3. x_k est mesuré par le matériel : résolution imposée
4. combien de bits sur les cofficients b_k pour garantir le résultat de la sortie ?

50 coefficients représentés en nombres à virgule fixes



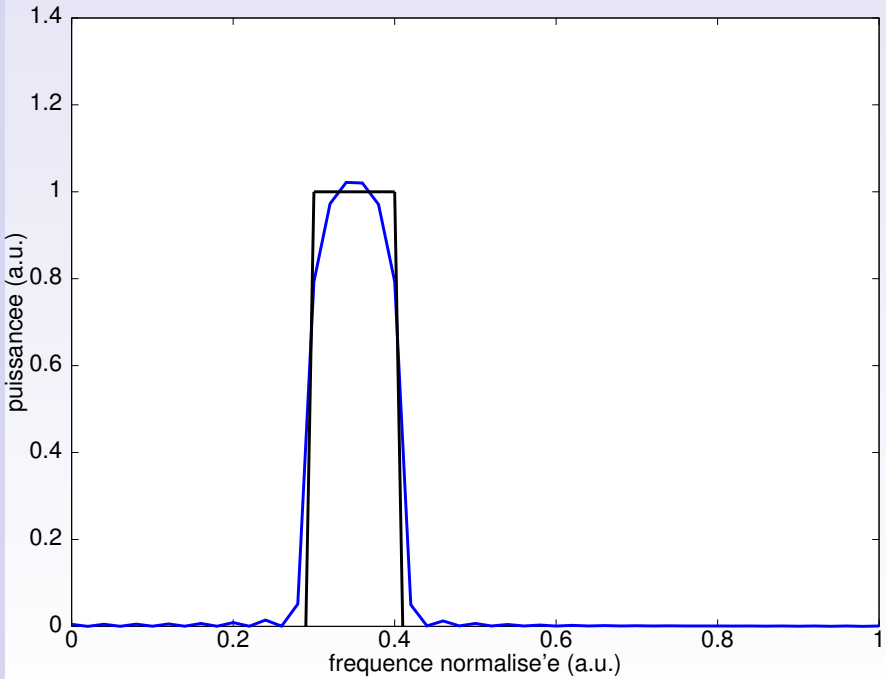
1. Fonction `firls` de GNU/Octave (ou Matlab) pour fournir les fréquences normalisées, les coefficients de pondération, et le nombre de coefficients `a_k`
2. $y_n = \sum_1 \frac{a_k}{a_0} y_{n-k} - \sum_0 \frac{b_k}{a_0} x_{n-k}$ ou en transformée en Z :

$$H(z) = \frac{\sum_0 b_{k+1}/a_0 \cdot z^{-k}}{1 + \sum_1 a_k/a_0 \cdot z^{-k}}$$

FIR v.s IIR

1. FIR : $y_n = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées, pas de risque de divregence, mais long à converger
2. IIR : $\sum a_k y_{n-k} = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées *et des sorties passées*, risque de divregence, mais convergence rapide
3. x_k est mesuré par le matériel : résolution imposée
4. combien de bits sur les cofficients b_k pour garantir le résultat de la sortie ?

150 coefficients représentés en nombres à virgule fixe

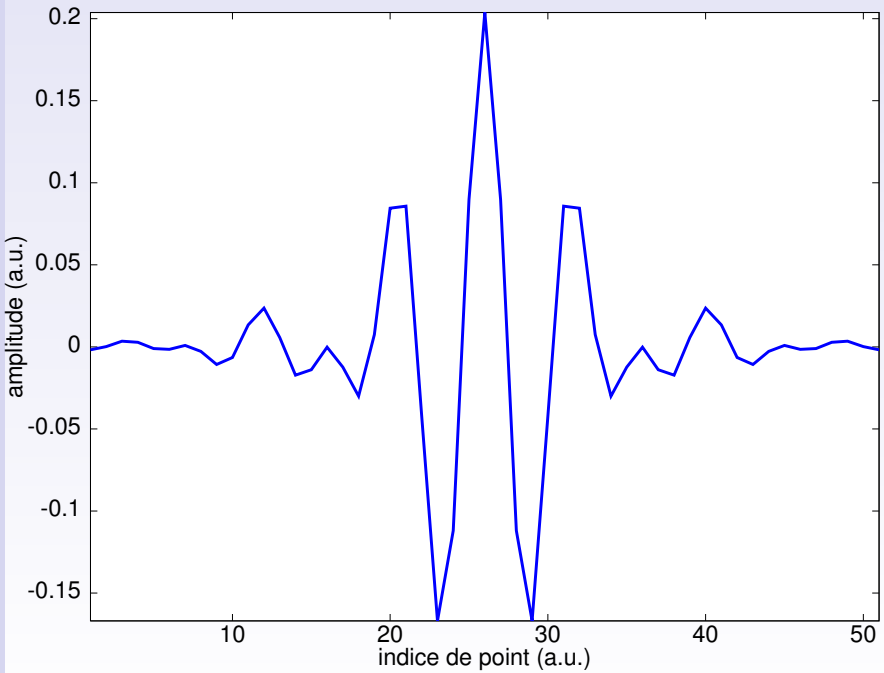


1. Fonction `firls` de GNU/Octave (ou Matlab) pour fournir les fréquences normalisées, les coefficients de pondération, et le nombre de coefficients `a_k`
2. $y_n = \sum_1 \frac{a_k}{a_0} y_{n-k} - \sum_0 \frac{b_k}{a_0} x_{n-k}$ ou en transformée en Z :

$$H(z) = \frac{\sum_0 b_{k+1}/a_0 \cdot z^{-k}}{1 + \sum_1 a_k/a_0 \cdot z^{-k}}$$

FIR v.s IIR

1. FIR : $y_n = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées, pas de risque de divregence, mais long à converger
2. IIR : $\sum a_k y_{n-k} = \sum b_k \cdot x_{n-k}$: somme pondérée des entrées *et des sorties passées*, risque de divregence, mais convergence rapide
3. x_k est mesuré par le matériel : résolution imposée
4. combien de bits sur les cofficients b_k pour garantir le résultat de la sortie ?



1. Fonction `firls` de GNU/Octave (ou Matlab) pour fournir les fréquences normalisées, les coefficients de pondération, et le nombre de coefficients `a_k`
2. Les coefficients du FIR sont la réponse impulsionnelle du filtre

Mise en pratique

- Échantillonnage périodique d'un signal sur *timer* (interruption pour lire ADC)
- Filtrage (conception d'un FIR)
- Traduire un algorithme exprimé en virgule flottante vers des entiers
- Estimer le nombre de décimales (bits) nécessaires pour garantir un l'exactitude d'un résultat