Informatique - Corrigé DM1

Q18. La précision demandée est 0,02 mm Hg : on choisit donc de représenter cette valeur par une unité de la valeur numérique : 1 bit <-> 0,02 mm Hg.

On veut échantillonner des valeurs jusqu'à 1350 hPa ce qui donne 1350/1013*750 = 999,5 mmHg.

Cela représente une valeur après numérisation de 999,5/0,02 = $\boxed{49975}$

10 bits permettent de représenter des entiers de 0 à 2**10 = 1024, 12 bits donnent une plage 0 à 4096 et 16 bits de 0 à 65536.

On choisit $\boxed{\text{des CAN avec une résolution de 16 bits}}$.

Et, on règle les tensions pour avoir 1 bit $\longleftrightarrow$ 5/65536 = 7,6.10^5 V $\longleftrightarrow$ 999,5/65536 = 0,016 mm Hg.

Q19. La durée d'échantillonnage multipliée par la fréquence d'échantillonnage donne 60*1000 = 60000 valeurs numériques à stocker.

Une valeur numérique est une valeur entière sur 16 bits entre 0 : tension de 0V et 65536 : tension de 5V. Alors on choisit de stocker les valeurs sous forme $\boxed{\text{d'entiers non signés sur 16 bits}}$. C'est le stockage optimal.

Un entier sur 16 bits est stocké dans 2 octets, alors le stockage nécessite 60000 × 2 = $\boxed{\text{120000 octets}}$.

Q20. Il s'agit d'un filtre analogique du deuxième ordre.

La forme de l'équation différentielle $\dfrac{1}{\omega_0^2}\ddot{U} + \dfrac{1}{\omega_0 Q}\dot{U} + U = Ue$ nous indique qu'il s'agit d'un filtre passe-bas de pulsation de coupure $\omega_0 = 20$ rad/s et de facteur de qualité $Q = 1/(2z) = 0,71$.

La fréquence de coupure est $\boxed{f_0 = \omega_0/(2*\pi) = 3,2 \text{ Hz}}$

et la fréquence des battements est $\boxed{f = 190 \text{ min}^{-1} = 3,2 \text{ Hz}}$.

$\boxed{\text{Le choix de la pulsation } \omega \text{ supérieure à la pulsation maximale des battements du coeur permet de ne conserver que la fréquence principale des pulsations cardiaques.}}$

$\boxed{\text{Le facteur de qualité inférieur à } \sqrt{2}/2 \text{ permet d'éviter tout phénomène de résonance}}$.

Q20,5. Le schéma numérique de la méthode d'Euler est

$$\dot{U}_{f,i+1} = \dot{U}_{f,i} + T_e \ddot{U}_{f,i}$$
$$U_{f,i+1} = U_{f,i} + T_e \dot{U}_{f,i}$$

Q21. En notant U pour U_f et U_e pour U_e, on écrit l'équation différentielle au temps i+1 ce qui donne

$$\ddot{U}_{i+1} = \omega^2 U_{e,i+1} - 2z\omega\dot{U}_{i+1} - \omega^2 U_{i+1}$$
$$\ddot{U}_{i+1} = \omega^2 U_{e,i+1} - 2z\omega\left(\dot{U}_i + (1-\gamma)T_e\ddot{U}_i + \gamma T_e \ddot{U}_{i+1}\right) - \omega^2\left(U_i + T_e\dot{U}_i + T_e^2(1/2-\beta)\ddot{U}_i + T_e^2\beta\ddot{U}_{i+1}\right)$$
$$\ddot{U}_{i+1} = \omega^2 U_{e,i+1} - 2z\omega\dot{U}_i - 2z\omega(1-\gamma)T_e\ddot{U}_i - 2z\omega\gamma T_e\ddot{U}_{i+1} - \omega^2 U_i - \omega^2 T_e\dot{U}_i - \omega^2 T_e^2(1/2-\beta)\ddot{U}_i - \omega^2 T_e^2\beta\ddot{U}_{i+1}$$
$$(2z\omega\gamma T_e + \omega^2 T_e^2\beta + 1)\ddot{U}_{i+1} = \omega^2 U_{e,i+1} + (-2z\omega(1-\gamma)T_e - \omega^2 T_e^2(1/2-\beta))\ddot{U}_i - (2z\omega + \omega^2 T_e)\dot{U}_i - \omega^2 U_i$$
$$\ddot{U}_{i+1} = \frac{\omega^2}{2z\omega\gamma T_e + \omega^2 T_e^2\beta + 1}\left(U_{e,i+1} + \left(-2\tfrac{z}{\omega}(1-\gamma)T_e - T_e^2(1/2-\beta)\right)\ddot{U}_i - \left(2\tfrac{z}{\omega} + T_e\right)\dot{U}_i - U_i\right)$$

Il s'ensuit que

$$\boxed{B_1 = \frac{\omega^2}{1 + 2z\omega\gamma T_e + \omega^2 Te^2\beta}, \quad B_2 = -\left(Te^2(1/2-\beta) + \tfrac{2z}{\omega}(1-\gamma)T_e\right), \quad B_3 = -\left(T_e + \tfrac{2z}{\omega}\right), \quad B_4 = -1}$$

Q22. La figure 7 représente le résultat de la simulation de la réponse du filtre à un échelon de tension de 1 V.

L'équation différentielle régissant ce filtre est de la forme $\ddot{s} + 2z\omega_0\dot{(s)} + \omega_0^2 s = \omega_0^2 e$

Le discriminant de l'équation caractéristique est $\Delta = 4z^2\omega_0^2 - 4\omega_0^2 = 4\omega_0^2(z^2-1)$.

Ici, $z = 0,7$ donc $z^2 - 1 < 0$ donc le régime est pseudo-périodique.

On sait que le coefficient d'amortissement α (noté z dans l'énoncé) vérifie $\alpha < 1$ alors les solutions de l'équation différentielle sont pseudo-périodiques de la forme $e^{at}(A\cos(bt) + B\sin(bt))$ et tendent rapidement vers une limite finie car $b < 0$ et cette limite est la valeur de U_e.

On constate que pour $T_e > 0,175$ les simulations numériques ne tendent pas vers une limite finie.

La simulation n'est donc pas correcte pour $T_e > 0,175$.

Pour $T_e < 0,13$, la simulation donne un résultat qui semble cohérent avec la solution attendue.

Q23. L'erreur η est proportionnelle au pas de temps $\eta \simeq 5T_e$. On dit que la convergence est linéaire ou d'ordre 1.

Q24. Avec la fréquence d'échantillonnage de 1000 Hz, on a $T_e = 0,001$ alors on obtient une bonne stabilité car $0,001 < 0,175$ et une erreur raisonnable car l'erreur maximale est $\eta \simeq 0,005$ (en Volts).

Q25. L'algorithme attendu par le jury du concours utilise des listes :

```python
def newmark(gamma, beta, omega, z, e, Te):
    N = len(e)
    U = [0]
    Up = [0]    # U'
    Upp = [0]   # U''
    for n in range(N-1):
        Upp.append(B1*(e[n+1] + B2*Upp[n] + B3*Up[n] + B4*U[n]))
        Up.append(Up[n] + (1-gamma)*Te*Upp[n] + gamma*Te*Upp[n+1])
        U.append(U[n] + Te*Up[n] + Te**2*(1/2 - beta)*Upp[n] + Te**2*beta*Upp[n+1])
    return U # toutes les valeurs de U
```

Une version utilisant **numpy**

```python
def newmark(gamma, beta, omega, z, e, Te):
    N = e.shape[0]
    U, Up, Upp = np.zeros(N) , np.zeros(N) , np.zeros(N)
    for n in range(N-1):
        Upp[n+1] =  B1*(e[n+1] + B2*Upp[n] + B3*Up[n] + B4*U[n])
        Up[n+1] = Up[n] + (1-gamma)*Te*Upp[n] + gamma*Te*Upp[n+1]
        U[n+1] = U[n] + Te*Up[n] + Te**2*(1/2 - beta)*Upp[n] + Te**2*beta*Upp[n+1]
    return list(U) # toutes les lignes et la colonne 2
```

Q26. On ne conserve que les valeurs de U, U', U'' pour n (valeur courante) et $n+1$ (valeur suivante)

```python
def newmark(gamma, beta, omega, z, e, Te):
    N = len(e)
    U = [0]
    upp_courant, up_courant, u_courant  = 0,0,0     # notations upp, up, u pour U'', U', U
    upp_suivant, up_suivant, u_suivant = 0, 0, 0    # valeurs au rang suivant
    for n in range(N-1):
        upp_suivant = B1(*e[n+1] + B2*upp_courant + B3*up_courant + B4*u_courant)
        up_suivant = up_courant + (1-gamma)*Te*upp_courant + gamma*Te*upp_suivant
        u_suivant = u_courant + Te*up_courant + Te**2*(1/2 - beta)*upp_courant + Te**2*beta*upp_suivant
        U.append(u_suivant)
        upp_courant,up_courant,u_courant = upp_suivant,up_suivant,u_suivant
    return U
```