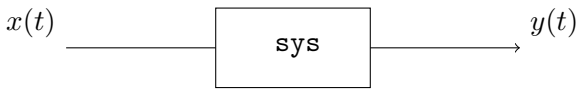
III – Fête de Laplace

Nous allons dans cette séance explorer différentes notions liées à la transformée de Laplace et aux systèmes linéaires. Dans un objectif pédagogique, nous allons faire beaucoup de choses « à la main », mais notez que MATLAB sait calculer et manipuler symboliquement des transformées.

Assurez-vous que la boîte à outils *Control* est présente et chargée dans votre copie de MATLAB (c'est le cas par défaut dans la version en ligne).

A) Retrouver la réponse à un échelon

On considère le système



dont la sortie $y(t)$ est liée à l'entrée $x(t)$ par l'équation différentielle avec conditions initiales

$$y''(t) + 0{,}6\,y'(t) + y(t) = x(t), \qquad y(0) = y'(0) = 0.$$

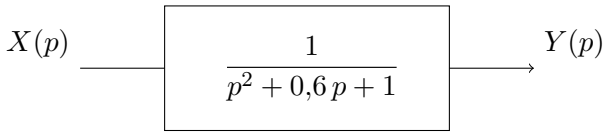
En passant dans le domaine opérationnel, cette équation s'écrit (n'est-ce pas ?)

$$(p^2 + 0{,}6\,p + 1)\,Y(p) = X(p),$$

de sorte que la transformée de la sortie est obtenue à partir de celle de l'entrée par simple division :

$$Y(p) = \frac{X(p)}{p^2 + 0{,}6\,p + 1}.$$

Du point de vue opérationnel, le système est donc caractérisé par sa *fonction de transfert* :



Pour définir ce système dans MATLAB, on exécute les commandes

```
p = tf("p");              % this defines p as a transfer function (tf) and
                          % will be used henceforth as the Laplace variable
sys = 1/(p^2+0.6*p+1)
```

Dans cette première partie, on doit retrouver la sortie du système lorsque l'entrée est un échelon unitaire.

Définissons le tableau contenant les instants d'observation par les commandes

```
Tend = 20;                % end of experiment (in seconds)
N = 1000;                 % number of sample points
t = linspace(0,Tend,N);   % time vector with N uniformly spaced points between 0 and Tend
```

Remarquez que, MATLAB étant capable de résoudre numériquement les équations différentielles, il peut approximer la réponse recherchée sans utiliser Laplace ni aucune théorie : c'est ce que fait la commande `step` (NB : échelon = *step function* en anglais).

```
step(sys,t);
```

À nous de jouer avec Laplace ! L'entrée est la transformée de l'échelon H :

```
input = 1/p;
```

la sortie, dans le domaine opérationnel, s'obtient comme le produit :

```
output = input * sys
```

Avant de faire la décomposition en éléments simples, il faut récupérer numérateur et dénominateur comme ceci :

```
[num, den] = tfdata(output, "v");
[R, P, K] = residue(num, den);
```

Cette dernière commande décompose la fraction `output` en éléments simples dans $\mathbb{C}$:

$$\texttt{output} = \sum_{m=1}^{M} \frac{r_m}{p - p_m} + \sum_{i=1}^{N} k_i \, p^{N-i}$$

`R` contient les numérateurs r_m (des nombres complexes), `P` les pôles (supposés *simples*) et `K` la partie entière (qui est nulle pour un système stable).

Or

$$\frac{r}{p - p_0} = \mathscr{L}\left[r e^{p_0 t}\right].$$

On commence par construire un vecteur `y` ayant la bonne taille et des zéros partout, puis l'on additionne toutes les exponentielles complexes :

```
y = zeros(size(t));
for i=1:length(R)
  y = y + R(i) * exp(t*P(i));
end
```

Ne reste qu'à tracer et vérifier l'adéquation des deux méthodes :

```
hold on;              % allows new graphic objects to be added to the old ones
plot(t,real(y),"r");
```

Notez qu'on peut obtenir directement les pôles de la transformée de Laplace de la réponse avec `pole(output)` et visualiser ceux-ci grâce à la commande `pzplot`.

En raisonnant sur la décomposition en éléments simples, on sait que chaque pôle simple $\lambda \in \mathbb{C}$ de la transformée fera apparaître dans l'expression temporelle de la réponse un terme en $e^{\lambda t}$. Par conséquent, on sait que l'on aura dans la sortie :

- un terme constant ($\lambda = 0$) représentant la tendance à long terme (régime établi) ;

- des oscillations amorties correspondant aux deux pôles complexes conjugués $\lambda \approx -0{,}3 \pm 0{,}9539\,\mathrm{i}$ faisant apparaître des termes en

$$e^{-0{,}3t}(A\cos(0{,}9539t) + B\sin(0{,}9539t)) = Ce^{-0{,}3t}\sin(0{,}9539t + \phi)\,:$$

oscillation de pulsation 0,9539 (donc fréquence 0,1518 ; période 6,5868) amorties avec un temps de demi-vie $\ln 2/0{,}3 \approx 2{,}31$, ce qui est cohérent avec ce qu'on observe sur le graphe.

1. Tracer de même la réponse à un échelon du système dont la fonction de transfert est

$$G(p) := \frac{10(5p+1)}{p^3 + p^2 + 42p + 4}$$

Repérez les pôles de ce système et discutez des constantes de temps présentes dans la réponse à l'échelon (fréquence d'oscillation, vitesse d'atténuation, ...)

B) Réponse à une sinusoïde pure

Reprenons le système initial : $F(p) := \dfrac{1}{p^2 + 0{,}6\,p + 1}$ et imposons-lui une entrée sinusoïdale $x(t) := \sin(\omega t)$ de pulsation ω.

Pour laisser passer les phénomènes transitoires, nous allons simuler tout cela sur un intervalle suffisamment long, disons $[0, 60s]$:

```
Tend = 60;
NbPoints = 1000;
t = linspace(0,Tend,NbPoints);
```

MATLAB propose la commande `lsim` qui permet de calculer numériquement la réponse à une entrée quelconque dans le domaine temporel (ici dans le cas où $\omega = 2$) :

```
x = sin(2*t);
y = lsim(sys, x, t);
clf; hold on;
plot(t,x); plot(t,y)
```

On observe qu'en régime établi, la sortie est une sinusoïde de même période que l'entrée (légèrement déphasée) et d'amplitude :

```
max(y(NbPoints/2:NbPoints))   % on ignore la première moitié des valeurs
```

2. Refaire les calculs pour d'autres valeurs de ω afin d'observer si les observations ci-dessus se confirment.

Pour vous aider dans vos explorations, vous pouvez utiliser une boucle comme celle dont le gabarit est donné ci-dessous :

```
for omega = 0:.1:2
  % your code here
end
```

3. Pour quelle valeur de ω obtient-on la sortie dont l'amplitude en régime établi est la plus grande ? (Vous pourriez par exemple tracer le graphe de l'amplitude en fonction de ω pour l'observer).

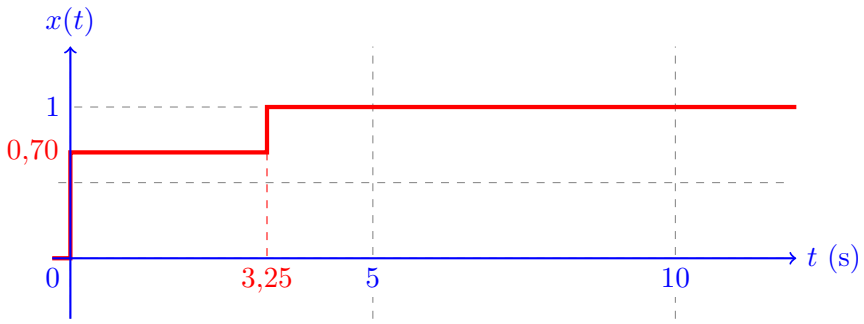
   Souvenirs d'un cours d'asservissement : comment retrouver cette pulsation directement ?

C) Réponse à une entrée originale

Le dépassement que présente le système est un élément potentiellement gênant (surchauffe d'un four, collision sur une butée, etc.) pour éviter cela, il faut réguler le système : très souvent, on met un capteur (onéreux) qui permet un retour et on adapte la commande en fonction de l'écart mesuré entre consigne et observation.

Voici une autre méthode qui, certes a des inconvénients, mais qui ne nécessite pas de capteur (commande en boucle ouverte). Le but est donc d'atteindre la consigne (constante d'amplitude 1) sans (trop de) dépassement et le plus rapidement possible.

4. Simuler la sortie du système avec fonction de transfert $F(p)$ si l'entrée est



5. (bonus) D'où sortent ces valeurs (0,70 et 3,25) ? Peut-on faire mieux ?