SIMSYS TD2
Résolution d'équations différentielles

L'objectif de ce TD est de mettre en œuvre des méthodes numériques d'intégration d'équations différentielles. On appliquera ces méthodes au modèle dynamique du bras plan obtenu lors du TD précédent.

1. Définitions

Une variable $y(t)$ est décrite par une équation différentielle ordinaire du premier ordre $\frac{dy(t)}{dt} = f(y(t), t)$. On connait les conditions initiales $y(t = t_0) = y_0$. Comment déterminer $y(t)$ pour un temps quelconque ?

Tout d'abord, on ne pourra pas obtenir $y(t)$ pour un temps « quelconque ». En pratique, on discrétisera le temps pour calculer $y(t)$ sur un nombre fini de points. Le pas de temps à utiliser doit faire l'objet d'un choix judicieux, car si un grand pas de temps va tendre à minimiser le nombre d'opérations nécessaires, il va également engendrer une grande erreur. Un pas plus petit améliorera les résultats mais pour un coût de calcul plus élevé. Enfin, un pas de temps trop petit entrainera une amplification des erreurs de troncature. Il existe donc un pas idéal.

Les deux algorithmes parmi les plus simples pour résoudre numériquement des équations différentielles sont les méthodes d'Euler et les méthodes de Runge-Kutta. Nous allons détailler ces deux méthodes dans les paragraphes qui suivent.

2. Méthodes d'Euler

Ce sont les méthodes les plus simples pour résoudre une équation différentielle du premier ordre. Cependant, ces méthodes sont rarement utilisées en raison de leur manque de précision et de leur instabilité. Nous avons comme données de départ :

$$\begin{cases} \dfrac{dy(t)}{dt} = f(y(t), t) \\ y(t = t_0) = y_0 \end{cases}$$

Discrétisons d'abord le temps en prenant des pas de temps de la même durée $h \rightarrow t_k = t_0 + kh$. On notera pour simplifier : $y(t_k) = y_k$. Le problème s'exprime désormais sous la forme suivante :

$$\begin{cases} \frac{dy_k}{dt} = f(y_k, t_k) \\ y(t = t_0) = y_0 \\ t_k = t_0 + kh \end{cases}$$

Exprimons la dérivée de $y(t)$ par une différence finie. Cela conduit à trois expressions différentes, suivant la différence finie considérée :

$$\begin{cases} y'_{t_k} = \frac{y_{t_{k+1}} - y_{t_k}}{h} + \partial(h) & \text{Différence avant d'ordre 1} \\ y'_{t_k} = \frac{y_{t_k} - y_{t_{k-1}}}{h} + \partial(h) & \text{Différence arrière d'ordre 1} \\ y'_{t_k} = \frac{y_{t_{k+1}} - y_{t_{k-1}}}{2h} + \partial(h^2) & \text{Différence centrée d'ordre 2} \end{cases}$$

Suivant que l'on considère l'une de ces trois différences finies, on va pouvoir déterminer trois méthodes différentes :

- **Méthode d'Euler explicite :** on considère la différence avant d'ordre 1. En remplaçant dans l'équation différentielle discrétisée, on obtient l'équation à résoudre à chaque pas de temps :

  $$y_{k+1} = y_k + hf(y_k, t_k) + \partial(h^2)$$

  Cette méthode est peu utilisée en raison de son manque de précision et de stabilité.

- **Méthode d'Euler implicite :** On considère cette fois-ci la différence arrière. La méthode est plus gourmande en temps de calcul car on a une équation plus complexe en y_k à résoudre :

  $$y_k - hf(y_k, t_k) = y_{k-1} + \partial(h^2)$$

  Cette équation se résout numériquement, ce qui demande une méthode de recherche de racine (comme par exemple la méthode de Newton-Raphson). Elle est par contre inconditionnellement stable.

- **Méthode d'Euler centrée :** on considère cette fois-ci la différence centrée. Elle est néanmoins peu utilisée car il faut pouvoir calculer y_{k+1} , y_k , y_{k-1} sur le même pas.

  $$y_{k+1} = y_{k-1} + 2hf(y_k, t_k) + \partial(h^3)$$

On notera que seule la méthode Euler implicite est inconditionnellement stable. On peut se retrouver avec des résultats divergents pour les deux autres méthodes, c'est pourquoi il convient de bien choisir la méthode en fonction du problème.

3. Méthodes de Runge-Kutta

Ce sont les méthodes de résolution d'équations différentielles les plus courantes, car elles sont simples à implémenter, relativement rapides, stables et précises (rien que ça). Elles nécessitent cependant à chaque pas de calcul le calcul de plusieurs itérations.

Nous allons détailler les deux méthodes les plus courantes, la méthode de Runge-Kutta d'ordre 2 et la méthode d'ordre 4, combinant respectivement 2 et 4 itérations successives.

- **Méthode de Runge-Kutta d'ordre 2 :** Elle combine deux itérations successives de la méthode d'Euler explicite. On évalue une première fois la dérivée y'_k au point courant y_k et on l'utilise pour une première approximation du point y_{k+1}. On note cette valeur α. Une fois calculée cette première approximation, on l'utilise pour recalculer la valeur de la dérivée à mi-parcours :$t_k + \frac{h}{2}$. On note cette valeur β.On utilise alors cette deuxième valeur de la dérivée afin d'approximer le résultat y_{k+1}. La procédure se résume donc à :

$$\begin{cases} \alpha = hf(y_k, t_k) \\ \beta = hf(y_k + \frac{\alpha}{2}, t_k + \frac{h}{2}) \\ y_{k+1} = y_k + \beta + \partial(h^3) \end{cases}$$

  On imagine aisément qu'augmenter le nombre d'itérations va tendre à améliorer la précision sur l'approximation réalisée.

- **Méthode de Runge-Kutta d'ordre 4 :** on augmente donc le nombre d'itérations, ici au nombre de 4. Sa précision est encore meilleure :

$$\begin{cases} \alpha = hf(y_k, t_k) \\ \beta = hf(y_k + \frac{\alpha}{2}, t_k + \frac{h}{2}) \\ \gamma = hf(y_k + \frac{\beta}{2}, t_k + \frac{h}{2}) \\ \delta = hf(y_k + \gamma, t_k + h) \\ y_{k+1} = y_k + \frac{1}{6}(\alpha + 2\beta + 2\gamma + \delta) + \partial(h^5) \end{cases}$$

On notera que les méthodes de Runge-Kutta présentées ici sont des méthodes explicites, qui présentent également des limites pour la résolution d'équations différentielles complexes (notamment équations aux dérivées partielles). Néanmoins, pour les problèmes classiques tels que les problèmes de mécanique des solides rigides, ce type de méthode est très efficace.
Voir également les slides pour plus d'explications dans la résolution des équations associées aux systèmes de solides rigides polyarticulés.

4. Equations différentielles d'ordre supérieur à 1

Les équations différentielles d'ordre supérieur à 1 peuvent aisément être transformées en des systèmes d'équations d'ordre 1, moyennant la définition de variables intermédiaires.

De manière générale, une équation différentielle d'ordre p et ses conditions initiales s'écrivant :

$$\begin{cases} y^{(p)}(t) = f(y(t), y'(t), y''(t), \dots, y^{(p-1)}(t), t) \\ y(t_0) = y_0 \\ y'(t_0) = y_0' \\ \vdots \\ y^{(p-1)}(t_0) = y_0^{(p-1)} \end{cases}$$

Cette équation peut se réécrire :

$$\begin{cases} Y'(t) = f(Y(t), t) \\ Y(t_0) = Y_0 \end{cases}$$

Avec

$$Y(t) = \begin{pmatrix} y(t) \\ y'(t) \\ \vdots \\ y^{(p-1)}(t) \end{pmatrix} \text{ et } Y_0 = \begin{pmatrix} y_0 \\ y_0' \\ \vdots \\ y_0^{(p-1)} \end{pmatrix}$$

Quand on utilisera une méthode numérique, on appliquera donc la méthode à un vecteur et non à un scalaire.

5. Intégrer en pratique

Choix du pas de temps :

On a dit précédemment que le choix du pas de calcul était primordial afin de résoudre une équation différentielle. On peut définir le choix du pas de temps de cette manière : Le pas de temps doit être choisi nettement inférieur au temps caractéristique de la fonction $y(t)$ à calculer. Par exemple, pour une fonction périodique de période T, on choisira $h \ll T$.

Pour programmer Runge-Kutta en Matlab :

Il va être nécessaire que vous passiez en argument de votre fonction d'intégration une autre fonction qui contiendra le système d'équations que vous voulez résoudre. Ça peut paraitre difficile comme ça, mais il suffit tout simplement d'adopter la syntaxe suivante :

```
1-    % exemple de fonction prenant en argument une autre fonction
2-
3-    function [c]=exemple(fun,a)
4-
5-    c=feval(fun,a) ;
6-
```

Cette fonction, en fonction de l'argument fun, va rendre fun(a) pour c. La fonction feval permet d'évaluer la valeur de la fonction « fun » pour l'argument a.

Par exemple, si j'appelle cette fonction de la manière suivante (vous noterez la présence des côtes ' ' nécessaires pour appeler une fonction en argument ici) :

```
>> w=exemple('exp',10)

w =

  2.2026e+004
```

W prend donc pour valeur exp(10), tout simplement. Il est donc important de bien définir la fonction fun que l'on va passer en argument afin de pouvoir l'évaluer dans la fonction cible.

Si je définis une fonction :

```
1-    % exemple de fonction prenant en argument une autre fonction
2-
3-    function [y]=jesuisunargument(x)
4-
5-    y=x*x;
6-
```

On pourra appeler la fonction exemple de la manière suivante :

```
>> w=exemple('jesuisunargument',10)

w =

  100
```

W prend bien la valeur voulue…

6. Application au bras robot 2ddl

Question 1 : à partir du travail que vous avez réalisé au précédent TD, exploitez votre script symbolique afin d'obtenir non plus les couples moteurs en fonction du mouvement, mais l'accélération en fonction de q,dq,Fext et Tau.

Question 2 : reformuler le problème sous la forme d'une équation différentielle vectorielle du premier ordre (système d'équations différentielles du premier ordre). En d'autres termes, exprimer la dérivée de l'état du système en fonction de l'état du système et de la commande.

Question 3 : Réalisez une fonction « Runge-Kutta 4 » permettant de d'obtenir l'estimée à l'instant suivant d'un vecteur de variables régi par une équation différentielle du premier ordre :

```
Function y=RungeKutta_4(func,t0,h,y0,varargin)
```

On notera l'usage de la variable de taille indéfinie 'varargin' permettant de passer des paramètres en nombre indéfini à une fonction. Voir l'aide de Matlab pour plus de détails.

'func' est une chaine de caractères que l'on pourra transformer en appel de fonction grâce à la fonction 'str2func' (voir l'aide de Matlab). Cette chaine de caractères est le nom de la fonction représentant le système d'équations différentielles.

Question 4 : Ecrire le système d'équations différentielles défini à la question 2 sous la forme d'une fonction Matlab exprimant l'état dérivé en fonction de l'état courant et des données de commande du système à l'instant t.

```
Function dQ=FD_robot_plan(t,Q,specs,Fext,Tau,g)
```

Question 5 : Pour tester votre méthode, étendez le script défini au TD précédent afin de réaliser des essais en dynamique inverse pour obtenir des couples moteurs associés à des mouvements, puis réalisez l'appel à la dynamique directe à l'aide de ces couples afin de vérifier le mouvement obtenu.