

Modélisation de l'Hydroptère

Basile Graf

4 mai 2006

Intoduction

Buts du projet

Modèle

Géométrie du modèle

Génération du modèle

Simulation

L'Hydroptère, un voilier à hydrofoils



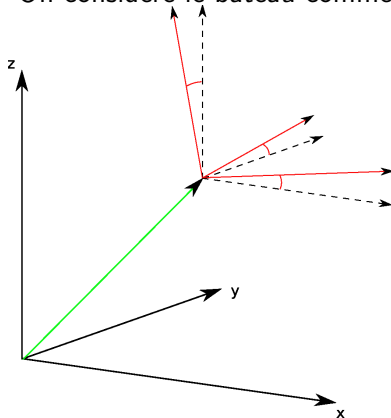
- ▶ Longueur 18m
- ▶ Envergure 24.5m
- ▶ Hauteur du mât 27m
- ▶ Masse 6.5t
- ▶ Grand Voile 165m²
- ▶ Vitesse de pointe 45 noeuds (83km/h)

Buts

- ▶ Continuer la modélisation de l'hydroptère
 - ▶ Comportement des surfaces portantes (polaires)
 - ▶ Comportement des voilures (polaires)
 - ▶ Gouvernes
 - ▶ Inclure les paramètres réels de la machine (ou réalistes si pas disponibles)
- ▶ Traduction du modèle de MATLAB à C++
- ▶ (Implémentation en temps réel)

Géométrie du modèle (1)

On considère le bateau comme un solide



Pour le situer dans l'espace, il faut

- ▶ 3 coordonnées pour sa

$$\text{position } \vec{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- ▶ 3 coordonnées pour son

$$\text{attitude } \vec{a} = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}$$

Géométrie du modèle (2)

Pour décrire la dynamique, il faut les coordonnées $\begin{pmatrix} \vec{r} \\ \vec{a} \end{pmatrix}$ et leur dérivées temporelles $\begin{pmatrix} \dot{\vec{r}} \\ \dot{\vec{a}} \end{pmatrix}$

L'état du système est donc décrit par $\vec{q} = \begin{pmatrix} \vec{r} \\ \vec{a} \\ \dot{\vec{r}} \\ \dot{\vec{a}} \end{pmatrix}$, $\dim(\vec{q}) = 12$

Évolution temporelle

L'évolution de l'état est décrite par le système d'équations différentielles

$$\dot{\vec{q}} = f(\vec{q}, \vec{Q}, \vec{p})$$

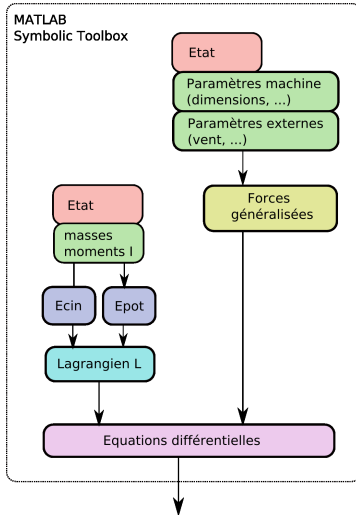
\vec{q} : état

\vec{Q} : forces appliquées

\vec{p} : paramètres (positions des gouvernes, vent, etc...)

Solution par intégration numérique

Génération du modèle sous MATLAB



Le programme génère des expressions qui calculent :

- ▶ Les paramètres géométriques
- ▶ Les forces
- ▶ Les moments de force
- ▶ Les vitesses
- ▶ Les accélérations

Simulation sous MATLAB

Le système $\dot{\vec{q}} = f(\vec{q}, \vec{Q}, \vec{p})$ obtenu est intégré sous MATLAB avec le "solver" `ode45(...)`

Problèmes :

- ▶ Les expressions à évaluer sont très longues (jusqu'à 20000 caractères)
- ▶ Sur mon Pentium IV, temps relatif de simulation :
 $\frac{t_{simulation}}{t_{reel}} = 400\%$
- ▶ On est loin du temps réel...

MATLAB \rightarrow C++

Écriture d'un programme MATLAB de traduction MATLAB \rightarrow C++ automatique

- ▶ Recherche de sous-expressions communes (Symbolic Toolbox)
- ▶ Recherche de fonctions trigonométriques fréquentes
- ▶ Variables intermédiaires et substitutions
- ▶ Fonction de traduction d'une expression MATLAB \rightarrow C++
- ▶ Génération de fichiers-fonctions .cpp

Note : Le compilateur MATLAB n'est pas adapté (interpréteur)

Résultats intermédiaires

- ▶ Pour l'instant, seules les fonctions d'évaluation $\vec{f}(\dots)$ sont implémentées en C++
- ▶ L'intégration se fait toujours depuis MATLAB par ode45()
- ▶ Sur la même machine : $\frac{t_{simulation}}{t_{reel}} = 3\% \text{ à } 16\%$
 - environ 100 fois plus rapide
 - temps réel envisageable

Aussi testé :

- ▶ Lecture de la position d'un joystick

Étapes suivantes (1)

Intégration en C++ (au lieu de `ode45()`) avec la librairie *ODE++*
ODE++ offre :

- ▶ Runge-Kutta d'ordres 4, 5 et 8 et d'autres méthodes
- ▶ Spécification d'erreur maximale et ou relative
- ▶ Intégration sur $t \in [a, b]$ ou pas à pas
- ▶ Pas variables automatiques
- ▶ ...

Étapes suivantes (2)

- ▶ Modélisation des surfaces portantes et des voiles par tables de polaires (jusqu'ici : modèle polynomial pour la version C++)
- ▶ Utiliser les données de la machine réelle (si on les obtient)
- ▶ Affichage *OpenGL* temps réel (ce serait rigolo)
- ▶ Pilotage par joystick (encore plus rigolo)
- ▶ Propositions ?