306

Scilab/Scicos による組込みシステムのモデルベース開発に関する研究

苫小牧工業高等専門学校 ○ 杉本 大志, 吉村 斎, 阿部 司, 北海道立工業試験場 大村 功

本研究では, Scilab/Scicos による組込みシステムのモデルベース開発を提案する. 適用例として, 二輪型倒立振子の安定化制御を試みた. 姿勢制御で重要な役割を持つジャイロセンサを改良し, 角度の計算方法等を変える事により, より安定した姿勢制御法を検証する.

1. はじめに

組込みシステム開発においては, C 言語やアセンブラ言語等の非オブジェクト指向言語による開発が行われているが, 近年は, UML を適用したオブジェクト指向開発も行われており, ハードウェア設計の場面でも UML が用いられている.

UML は, ソフトウェアに関する分析・設計を行うためのモデリング言語であり, 分析・設計を行っている内容が, ソフトウェアで提供される機能か, ハードウェアで提供される機能かについて十分注意する必要がある. もしもハードウェア自体も含めて検討を行う場合には, 他の言語を用いる必要がある [1]. そして, UML は組み込みシステム開発のいくつかの局面では使えないという事が指摘されている [1]. 例えば, 並行性・タイミングの表現や, 連続系のフィードバックループ等, 機能の制御系そのものを設計する事が難しい. この問題については, RT ミドルウェアによる設計手法が提案されている [1]. この手法では, コンポーネントとして Scilab/Scicos を取り込む事により, Scicos のブロック図から間接的にタスクを生成している.

本研究では, 特に組み込みシステムにおける制御系の設計とそのシミュレーションに着目し, 設計からシミュレーション, そして Scicos のブロック図からタスクを直接生成出来るシステムの構築を目的とする. 制御系設計やシミュレーションが行える Scilab/Scicos に注目し, UML で不足している機能を, Scilab/Scicos で補完する事により, 移植性や効率性を保った組込みシステム開発方法を提案する.

我々は, LEGO MINDSTORMS NXT を用いた二輪型倒立振子 NXTway-GS(図 1) を実機として採用し, Scilab/Scicos を用いてシミュレーションを行い, ブロック図から直接 NXTway-GS 用の姿勢制御タスクを自動的に生成する事を目的とした. この為に, 実機のモデリングと制御系の設計を行い, シミュレーションを行った. そして, シミュレーション結果から得たゲインと姿勢制御タスクを実装し, 実機を動作させた.

山本による先行研究 [3] では, 数値計算システム Matlab/Simulink によってシミュレーションを行っている. Matlab は多くの使用実績があり信頼性が高いシステムであるが, その反面非常に高価であり, 拡張パッケージの入手に関しても有償である. 更に, 『数の定義』や演算子のオーバロード等が出来ず, 『拡張性』に乏しいという欠点を持っている. 一方, 本研究で用いる Scilab/Scicos は, フランスの INRIA と ENPC で開発されているオープンソースの数値計算システムである. 拡張パッケージも無償で配布されており, ユーザが独自に『数の定義』や, 演算子のオーバロード等が出来, 『拡張性』に優れたシステムと言える.

オープンソースである点や『優れた拡張性』という利点を持つ Scilab/Scicos が, モデルベース開発に適した環境であると考え, 本研究ではこれを採用した.

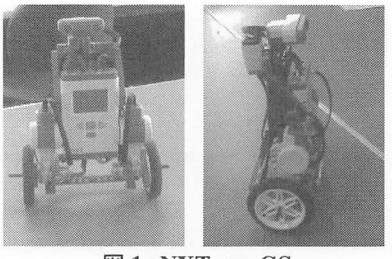

図 1 NXTway-GS

2. システム構成と処理の流れ

本研究で開発したシステムの機能は,

(1) シミュレータ機能
(2) 自動コード生成機能
(3) ビルド・アップロード機能

の 3 つである. 図 2 に, システム構成と処理の流れを示す.

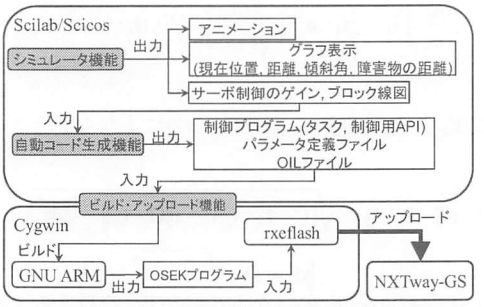

図 2 システム構成と処理の流れ

2.1 シミュレータ機能

プラントである NXTway-GS を, 二輪型倒立振子としてモデリングする. モデルの主要変数を表 1 に示す. ラグランジュ方程式をもとに, 鉛直近傍で線形化した運動方程式を立てると, 次式を得る.

$$\left[(2m+M)R^2+2J_w+2n^2J_m\right]\ddot{\theta}+\left(MLR-2n^2J_m\right)\ddot{\psi}=F_\theta \quad (1)$$

$$\left(MLR-2n^2J_m\right)\ddot{\theta}+\left(ML^2+J_\psi+2n^2J_m\right)\ddot{\psi}-MgL\psi=F_\psi \quad (2)$$

$$\left[\frac{1}{2}mW^2+J_\phi+\frac{W^2}{2R^2}\left(J_w+n^2J_m\right)\right]\ddot{\phi}=F_\phi \quad (3)$$

以上の運動方程式を $\mathbf{x}_1 = [\theta\ \psi\ \dot{\theta}\ \dot{\psi}]^T, \mathbf{x}_2 = [\phi\ \dot{\phi}]^T, \mathbf{u} = [v_l\ v_r]^T, y = \theta$ とする事で

$$\dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_1\mathbf{u} \quad (4)$$

$$\dot{\mathbf{x}}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_2\mathbf{u} \quad (5)$$

$$y = \mathbf{C}\mathbf{x}_1 \quad (6)$$

を得る.

この状態方程式を用いて, 状態フィードバックにより制御器を構成した. フィードバックゲインは最適レギュレータ法により設定した. 構成したサーボ制御系を図 3 に示す.

表 1 変数・定数の定義

定数	意味	定数	意味
g	重力加速度	m	駆動輪 1 本の質量
R	駆動輪半径	J_w	駆動輪慣性モーメント
M	車体質量	W	車体幅
L	車輪中心から車体重心までの距離	J_ψ	車体慣性モーメント(ピッチ)
J_ϕ	車体慣性モーメント(ヨー)	J_m	DC モータ慣性モーメント
		n	ギア比
θ	駆動輪の平均回転角度	ψ	ピッチ角度
ϕ	ヨー角度	F_θ	θ に対応する一般化力
F_ψ	ψ に対応する一般化力	F_ϕ	ϕ に対応する一般化力

2.2 自動コード生成機能

自動コード生成機能は,

(1) C ファイル
- 制御パラメータを定義したファイル
- バランス制御プログラム

(2) OIL(OSEK アプリケーション設定) ファイル
(3) make ファイル

を生成する. 制御パラメータを定義したファイルには,

- シミュレータで用いた各種ゲイン
- シミュレーション結果より得たサーボ制御のフィードバックゲイン

を記述する. バランス制御プログラムには, Scicos で構成した制御則 (図 3) を記述する.

2.3 ビルド・アップロード機能

make コマンドによって, OSEK プログラムを GNU ARM を用いてビルドする. そして, rxeflash コマンドによって, OSEK プログラムを NXTway-GS にアップロードする.

3. 検証

3.1 最適レギュレータ法を用いた姿勢制御

構成したサーボ制御系を図 3 に示す. 図 3 の $\mathbf{k}_f, k_i$ は, 最適レギュレータによって得たフィードバックゲインである.

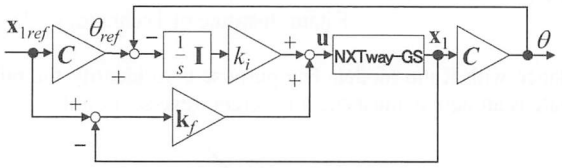

図 3 NXTway-GS 用サーボ制御器のブロック線図

ジャイロセンサでは, 電源印加後からジャイロセンサの出力電圧が変動するドリフト現象がある. 本研究で行っている様にジャイロセンサの出力を数値積分して傾斜角度を算出する際, ドリフト現象による誤差が蓄積される事により, 長時間に渡る姿勢維持動作に悪影響を与える事が予想される. そこで, 本研究では NXTway-GS が標準で装着しているジャイロセンサ (NXT Gyro Sensor NGY1044) を分解し, ホットボンドでケースとセンサを密着させる事で伝熱性を高める改良を行った.

シミュレーションを $0 \leq t \leq 50$ [sec.] で実施し, シミュレーションで得たゲインを実機に実装して, その時の応答を観測した. ジャイロセンサ改良前のシミュレーションと実機それぞれの応答を図 4 に, ジャイロセンサ改良後のシミュレーションと実機それぞれの応答を図 5 に示す.

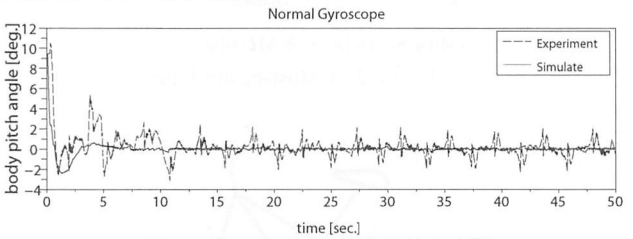

図 4 ジャイロセンサ改良前の応答

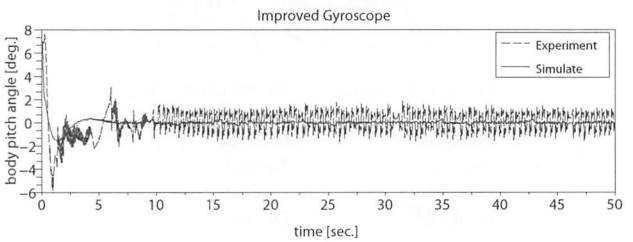

図 5 ジャイロセンサ改良後の応答

図 4 では初期車体傾斜角度を $\psi_0 = 9.318$[deg.], 図 5 では $\psi_0 = 6.905$[deg.] とした.

初期状態から目標値をゼロにする状態の応答では, シミュレーションでは車体傾斜角度がゼロの近傍で僅かに変化している. 実機の応答については, 図 4 では, ±2[deg.] 程度の比較的大きな振れ幅が定期的に現れているが, 図 5 では, ±1.5[deg.] 程度の振れ幅となっており, 改良前に比べ小さな触れ角が連続的に観測されている事が判る. これより, 改良によってドリフトの影響が抑えられているものと考えられる.

3.2 ドリフトの観測とプログラムによる補正

改良したジャイロセンサを装着した NXT を常温環境に静置して, 観測した角速度から数値積分によって角度を算出し, 角度の時間変化を測定した. 図 6 に示す様に, 300[sec.] 付近から単調減少を始めている.

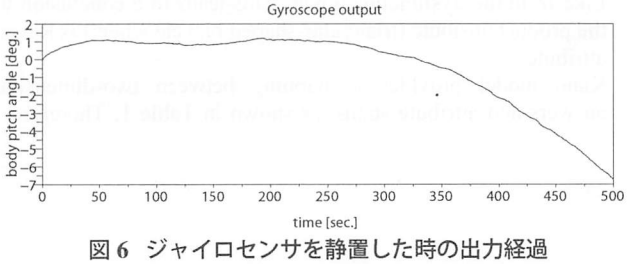

図 6 ジャイロセンサを静置した時の出力経過

図 6 を開始から 50[sec.] 毎に区切り二次曲線で近似し, この曲線を姿勢制御に反映した時の, 実機の応答を図 7 に示す.

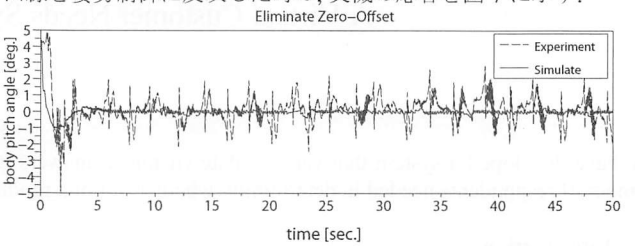

図 7 ドリフトを補正した時の実機の応答

目標値をゼロにする状態の応答では, 実機では, ±2[deg.] 程度の比較的大きな振れ幅となっているが, 図 4, 5 に比べ目標値への追従が早くなっている.

3.3 同一次元オブザーバのシミュレーション結果

ここでは, 同一次元オブザーバを用いて, NXTway-GS の傾斜角度と, ジャイロセンサの出力から得た傾斜角度との出力偏差をモデルにフィードバックするシステムを考える. オブザーバへの入力は, $\mathbf{x}_1$ とその状態推定値 $\hat{\mathbf{x}}_1$ の差 $\hat{\mathbf{x}}_{1e}$, $\mathbf{u}$ である. 図 8 は, オブザーバを含んだコントローラである.

図 9 に, 初期車体傾斜角度を $\psi_0 = 0.5$[deg.] とした時の, 最適レギュレータのみで制御系を構成した場合と, 最適レギュレータに同一次元オブザーバを組み合わせて制御系を構成した場合それぞれの応答を示す.

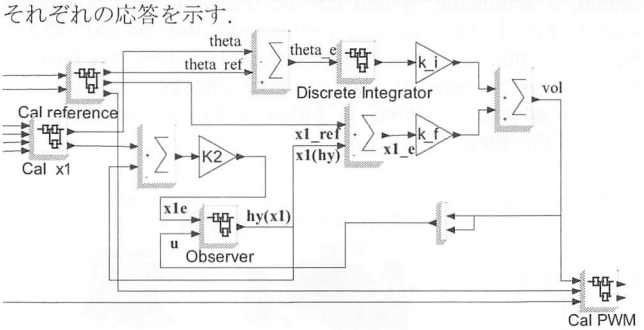

図 8 同一次元オブザーバを含んだコントローラ

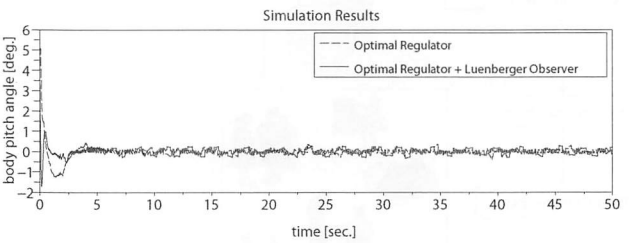

図 9 オブザーバを含んだ時のシミュレーション結果

上図の様に, オブザーバを組み合わせる事によって, 最適レギュレータ法のみで制御系を構成した場合に比べて目標値への追従が早くなる事が判る. 今後は, この制御則を実機に実装し, その応答を観測する事である.

4. おわりに

提案した Scilab/Scicos によるモデルベース開発手法を, 二輪型倒立振子に適用する事で, 制御系の設計から実機への実装を容易にし, 更に実機を安定して制御出来る結果を得た.

また, 改善した制御則をブロック線図で表現しシミュレーションを行い自動的にコード化して, その結果を観測した事により, 開発したコード生成機能の汎用性を確認した.

参考文献

[1] 水川真, 坂本武志, 大原賢一, “UML と RT ミドルウェアによるモデルベースロボットシステム開発”, オーム社, 2009 年.

[2] 藤倉俊幸, “UML を基礎から理解する ——UML でできること, できないこと”, Embedded and Electronics Engineers Network, 2003 年.

[3] Y.Yamamoto, “NXTway-GS Model-Based Design -Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT-”, CYBERNET SYSTEMS CO.,LTD. , 2008.