ブロック線図に基づくシステム制御学習環境におけるリアルタイム制御実験

Real-Time Control Experiment with Learning Environment for System Control based on Block Diagram

九州工業大学　○ 谷口 仁志，古賀 雅伸，矢野 健太郎
Hitoshi Taniguchi and Masanobu Koga, Kentaro Yano
Kyushu Institute of Technology

Abstract This paper describes a learning support system for system control. The system boots from an USB, and it reduces burdens about building, maintaining, and operation of a learning environment. The system makes it possible to learn system control based on block diagram. It also realizes efficient shift from simulation phase to experimental phase by automatically generating code for real-time experiment from block diagram. At the experimental phase the system utilizes real-time (RT) control package which provides facilities for RT processing on Linux without reconfiguring kernel. Therefore, it improves the efficiency of learning of system control based on block diagram.

1 はじめに

システム開発の考え方の一つに Model Based Design という考え方がある。UML やブロック線図などのモデルに基づいて開発プロセスを行う方法であり、開発プロセスの上流から下流までを一貫して実行することができる。この手法にとる制御系開発を支援するツールの 1 つとして MATLAB[1] が挙げられる。MATLAB では、Simulink によってブロック線図に基づいた制御系の解析、設計、シミュレーションを行うことができ、Real Time Workshop を用いることで Simulink 上のブロック線図から制御実験用のコードを出力することができる。これにより制御系開発プロセスを一貫して実行できる。

しかし、MATLAB/Simulink·Real Time Workshop[2] を制御工学教育分野へ導入するには、いくつかの問題がある。その一つが実験環境の構築に関わる問題である。一般の PC を実験に利用する場合、リアルタイム制御実験を行うには、リアルタイム OS を別途インストールする必要がある。リアルタイム OS の代表として、Linux をリアルタイム環境に拡張した RT-Linux[3] が挙げられるが、インストールにはカーネルの再構築が必要であり、専門的な知識が要求される。[1]

本研究では、システム制御の学習のためのリアルタイム制御実験環境の構築・管理・運用を容易に行うことができるシステムを開発した。本システムでは、制御系設計支援ツール Jamox[4] 上で作成したブロック線図から、リアルタイム制御実験のためのソースコードを生成することができるので、制御系のモデリング・解析・設計・シミュレーションだけでなく、制御実験まで含めた制御系開発プロセスを効率的に学習することができる。

[1] また、RT-Linux は現在開発が停止しているため、最新の PC 環境に適用できないという問題もある。

2 システム制御学習支援システム

2.1 システム制御学習支援システムの特徴

本研究で開発したシステム制御学習支援システムの特徴を以下に示す。

- システムの起動が容易かつ安全
  システム制御学習支援システムは、HDD にインストールして使用する他にも、CD ブートや USB ブートによって起動することができる。したがって、PC 内の既存の環境に影響を与えずに安全にシステムを起動することが可能である。
- リアルタイム実験環境の構築
  リアルタイム制御パッケージ [5] を利用し、容易にリアルタイム制御実験を行うことができる環境を提供する。リアルタイム制御パッケージの詳細については後述する。
- ブロック線図を用いた学習を支援
  制御系設計支援ツール Jamox と連携し、ブロック線図に基づいた学習環境を提供する。

2.2 システム制御学習支援システムの構成

システム制御学習支援システムの構成を図 1 に示す。CD ブートが可能な Linux である KNOPPIX[6] がこのシステムのベースとなっている。

2.2.1 KNOPPIX

KNOPPIX[6] は Debian GNU/Linux をベースにした CD ブート型の Linux である。KNOPPIX の特徴を以下に示す。

- HDD へインストールしなくても、CD ブートを用いて容易に Linux 環境を構築することができる。

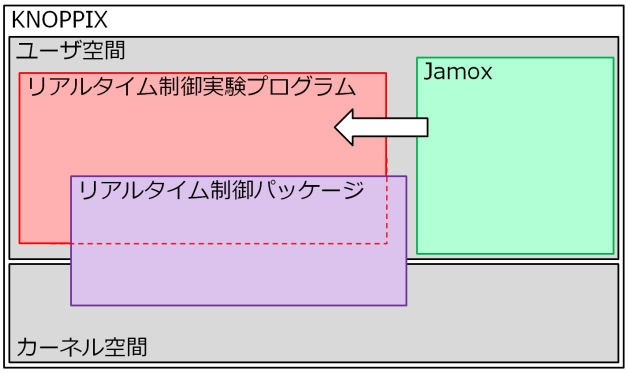


図 1: システム制御学習支援システムの構成

- システムトラブルが起きてもシステムを再起動することで環境を復元できる。
- オープンソースで開発されているため無償で入手でき、カスタマイズも行うことができる。

本研究では、これらの特徴に着目し、KNOPPIX を制御実験用にカスタマイズすることにより、CD/USB ブートするだけで容易に実験環境を構築できるシステムを開発した。

2.2.2 リアルタイム制御パッケージ [5]

制御実験環境には、制御対象のリアルタイム制御を可能にするために、時間制約の保障が必要である。しかし、Linux は、非リアルタイム OS であるため、時間制約を保障することができない。つまり、KNOPPIX 上でリアルタイム制御実験を実施することは困難である。

そこで、本研究では、この問題に対し、我々が研究開発しているリアルタイム制御パッケージを KNOPPIX に組み込むことにより、KNOPPIX 上での制御実験を可能にした。

リアルタイム制御パッケージは、非リアルタイム OS である Linux 上にリアルタイム環境を構築するミドルウェアである。リアルタイム制御パッケージは以下に示す 2 つのモジュールから構成される。

- リアルタイムタスク制御モジュール
  リアルタイムタスク制御モジュールは、カーネルモジュールであり、リアルタイムタスクの制御と管理を行う。そして、Linux カーネル 2.6 上でリアルタイム性を阻害する問題を解決し、Linux 上にリアルタイム環境を構築する。
- リアルタイムタスク操作モジュール
  リアルタイムタスク操作モジュールは、リアルタイム処理に必要な表 1 に示す API を提供する。

以下にリアルタイム制御パッケージの特徴を示す。

- インストールが容易

表 1: リアルタイムタスク操作モジュール

関数名	説明
pthread_create_np	RT タスクを生成
pthread_suspend_np	RT タスクの実行を中断
pthread_wakeup_np	RT タスクの実行を再開
pthread_make_periodic_np	周期処理の設定
pthread_stop_periodic_np	周期処理の終了
pthread_wait_np	次の周期まで待つ

リアルタイム制御パッケージのインストールは、リアルタイムタスク制御モジュールを Linux カーネルにロードする (insmod コマンドで組み込む) だけで容易に行うことができる。RT-Linux のようにカーネルの再構築を行う必要は無い。

- リアルタイムタスクがユーザ空間で動作
  リアルタイムタスクがユーザ空間のタスクとして動作する。そのため、タスク上で Linux 既存のソフトウェア資源を利用することができ、かつメモリ保護が適用される。

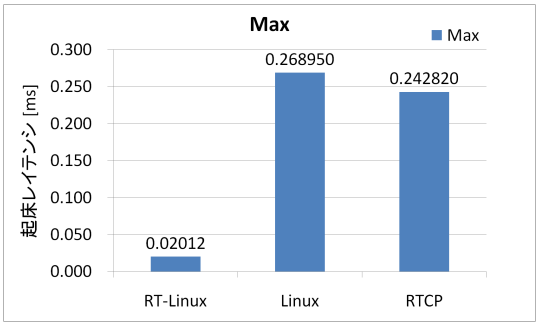


図 2: 起床レイテンシの最大値

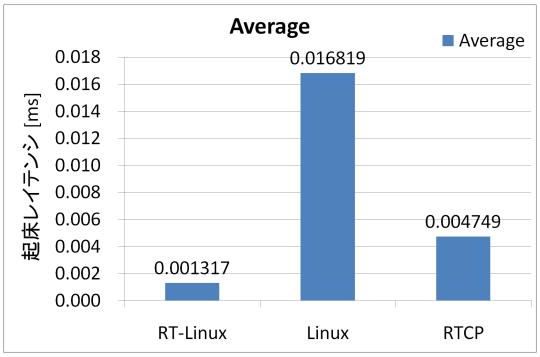


図 3: 起床レイテンシの平均値

RT-Linux、通常の Linux、リアルタイム制御パッケージでそれぞれ起床レイテンシの測定を行った結果を図 2、

図 3 に示す。起床レイテンシとは、RT タスクに対して起床要求が発生してから実際に RT タスクが起床するまでにかかる時間であり、これが小さいほどリアルタイム性能が高いといえる。

測定は、ディスクアクセスを繰り返す外部負荷を与えた状況下で、1[ms] 周期のので動作する RT タスクを 10000 回実行して測定した。このときの PC のスペックは CPU が Celeron M(1.3[GHz]) であり、メモリが 512[MB] である。

リアルタイム制御パッケージ (RTCP) を利用した場合の起床レイテンシの平均値は、RT-Linux の起床レイテンシの平均値と比べて約 3 倍になっているが、通常の Linux と比較した場合約 4 分の 1 になっており、起床レイテンシが改善されていることが分かる。

2.2.3 Jamox

システム制御学習支援システムでは、制御系設計支援ツール Jamox(Java Agile MOdeling Tool for Control System) を利用することで、ブロック線図を用いた制御工学学習を行うことができる。Jamox の外観を図 4 に示す。Jamox の特徴を以下に示す。

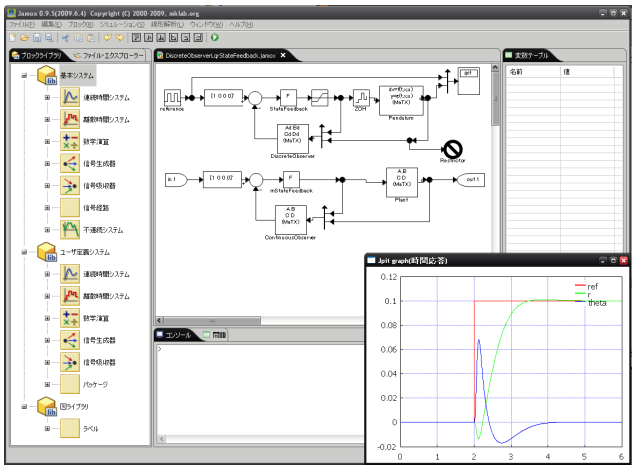


図 4: Jamox

- GUI による操作
  GUI によってブロック線図を簡単に作成することができる。ブロック線図の作成は、ブロックライブラリから作成したいブロックをエディタ上にドラッグ＆ドロップするだけで行うことができる。ブロックパラメータの変更もダイアログを呼び出すことで容易に行える。

  また、MATX エンジン [7] を利用したコンソールを利用でき、MATX の関数を利用した解析等も簡単に行うことができる。

- ユーザ定義ブロック
  Jamox では、Java や数値計算言語である MATX[8] を用いたユーザ定義ブロックを作成することができ、ブロックライブラリに無いシステムをユーザが自由に作成できる。また、MATX を用いて作成されたブロックから、Java や c 言語のコードを生成することができ、シミュレーションや実験で利用することができる。

  ユーザ定義ブロックの一覧を以下に示す。

  – 連続時間動的システム
  – 離散時間動的システム
  – 連続時間静的システム
  – 離散時間静的システム
  – 連続時間状態空間表現
  – 離散時間状態空間表現

- 各種シミュレーション
  時間応答や周波数応答のシミュレーションを GUI から容易に実行することができ、Gnuplot や Jpit[9] によってグラフに出力することができる。

- Java 言語で開発
  Jamox は Java 言語で開発されているため、Java が実行可能である多くのプラットフォームで利用することができる。

2.3 ブロック線図からのコード生成

システム制御学習支援システムでは、Jamox 上で作成したブロック線図からリアルタイム制御実験用のコードを生成でき、スムーズに実験フェーズに移行することができる。

2.3.1 リアルタイム制御実験用コードの生成

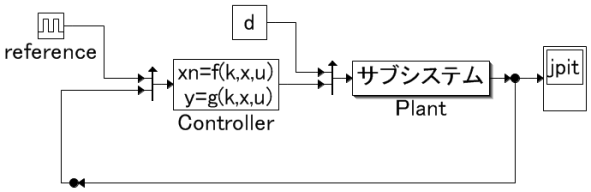


図 5: 実験用コード出力のためのブロック線図

図 5 のように、制御対象の出力と、目標値等の外部信号を入力とし、制御対象への入力を出力とする離散時間システムの Controller ブロックがリアルタイム制御実験用コード生成の対象ブロックとなる。

ユーザは、対象ブロックを MATX ユーザ定義ブロックとして作成する。MATX ユーザ定義ブロックでは、数値計算言語 MATX を用いてシステムの入出力関係を記述できる。

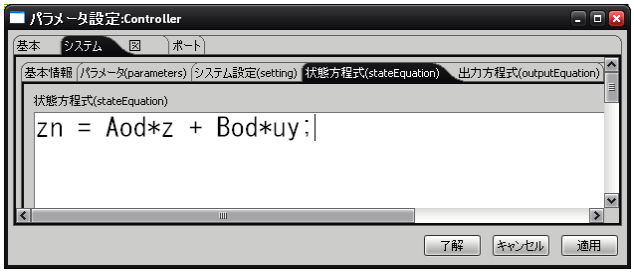


図 6: 状態方程式タブの設計例

図 7: 出力方程式タブの設計例

ユーザ定義 MATX ブロックの設計例を図 6、図 7 に示す。

図 8 に示すように、対象となるブロック上で右クリックメニューを利用して「制御コード出力」を選択すると、図 9 に示すダイアログが表示される。このダイアログでは以下に示すパラメータを設定することができる。

- サンプリング周期
- ログデータの最大サイズ
- ログデータの間引き数
- ファイルの出力先

それぞれのパラメータを設定し、出力ボタンをクリックすると、ブロックの情報と、これらのパラメータを基にして、リアルタイム制御実験用のコードが生成される。

ファイルは、MATX 形式で出力されるが、jmatc[7] を利用して C 言語の形式で出力することができる。

2.3.2 リアルタイム制御実験プログラムの構成

システム制御学習支援システム上でリアルタイム制御実験を行うプログラムの構成を図 10 に示す。

リアルタイム制御実験プログラムは大きく分けてリアルタイムタスク部分とハードウェア処理部分の 2 つに分けられる。このうち、ハードウェア処理部分は、ハードウェアの初期化や入出力などの処理を行っており、既存のドライバを利用できることもある。

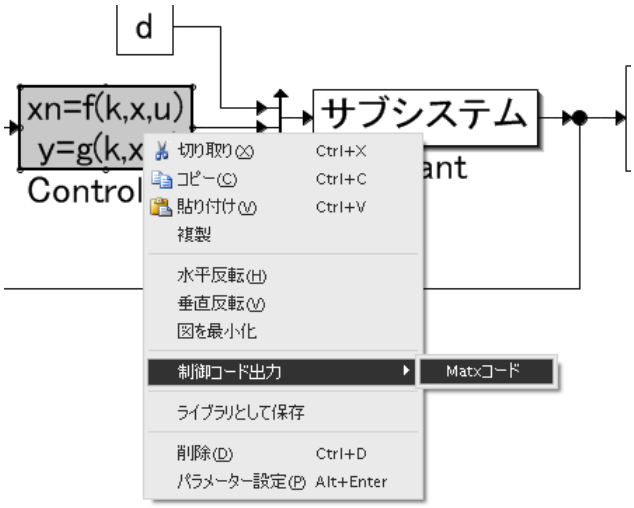


図 8: ブロック線図からの実験用コード生成

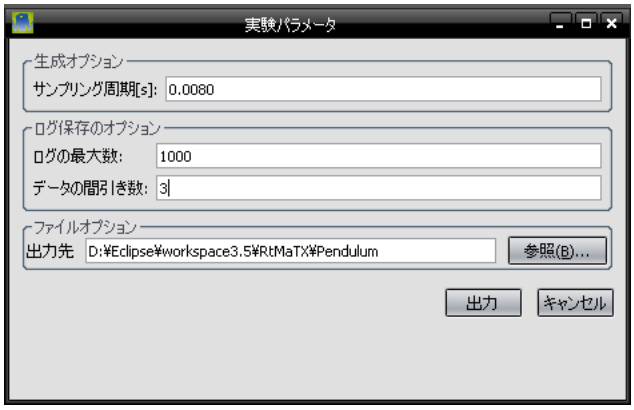


図 9: RT 実験パラメータ設定ダイアログ

リアルタイムタスク部分は、実験に関するリアルタイム処理を行う部分であり、この部分がブロック線図から自動的に生成される。

自動生成されるリアルタイムタスク部分は、2 種類のファイルから構成される。1 つは、ブロック毎に生成されるファイルで、もう 1 つは各ブロック間の結合 (Link) 関係が記述されているファイルである。ブロックごとに出力されるファイルには、各ブロックの状態方程式や出力方程式が記述されている。各ブロックの結合関係が記述されているファイルには、コード出力の際に設定したパラメータに関する情報も含まれている。

3 倒立振子の安定化制御実験への適用

開発した学習環境を、倒立振子の安定化制御へ適用する例を示す。まず、ブロック線図を用いて、モデリング、制御器の設計、シミュレーションを行う。そして、作成したブロック線図からリアルタイム制御実験のためのコードを生成し、安定化制御実験を行うという流れで学

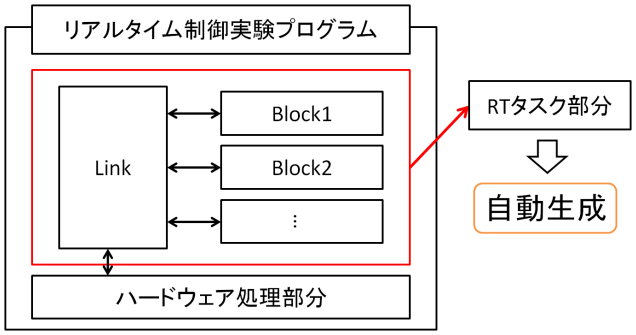


図 10: 制御実験プログラムの構成

習を進める。

利用するブロック線図の例を図 11 に示す。

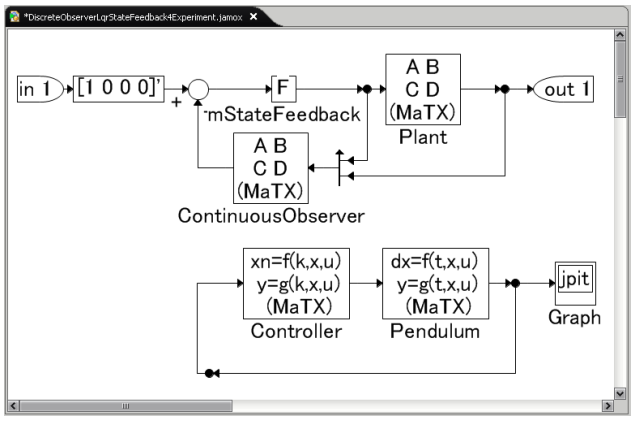


図 11: 倒立振子系のブロック線図

3.1 各ブロックの構成

図 11 に示したブロック線図の各ブロックの構成を以下に示す。

- Pendulum ブロック
  このブロックは制御対象である倒立振子を表すブロックである。ユーザ定義の連続時間 MATX 動的システムブロックを用いて表現される。

- Plant ブロック
  このブロックは、倒立振子の線形近似モデルを表すブロックである。ユーザ定義の連続時間 MATX 状態空間表現ブロックを用いて表現されている。

- mStateFeedback ブロック
  このブロックは、LQ 最適制御により状態フィードバックゲインを表すブロックである。ユーザ定義の MaTX 定数ブロックを用いて表現されている。

- ContiniousObserver ブロック
  このブロックは、制御対象に対する連続時間オブザーバを設計するブロックである。ユーザー定義の連続時間 MATX 状態空間表現ブロックを用いて表現されている。

- Controller ブロック
  このブロックは、mStateFeedback ブロックで設計された状態フィードバックゲインと ContinuousObserver ブロックで設計された連続時間オブザーバの情報を基にして離散化したコントローラを設計するブロックである。このブロックは、ユーザ定義の離散時間 MATX 動的システムブロックを用いて表現されている。 このブロックがリアルタイム制御実験用コード生成の対象ブロックとなる。

3.2 学習の流れ

本システムを用いて制御系開発プロセスを学習する場合の学習の流れと各プロセスにおける特徴を以下に示す。

1. パラメータの同定
   同定したパラメータを用いて、倒立振子ブロックを作成し、ステップ応答やフィードバック応答などのシミュレーションを行うことで同定されたパラメータの妥当性の検証を容易に行える。

2. 制御対象の解析
   Jamox のコンソールを利用することで、倒立振子の特性解析を Jamox 上で行うことができる。

3. 制御器の設計
   mStateFeedback ブロックの Q 行列、R 行列を設計することでフィードバックゲインを設計することができ、ContiniousObserver ブロックのオブザーバの極を設計することで、連続時間オブザーバを設計することができる。

   設計されたフィードバックゲインと連続時間オブザーバを用いて Controller ブロックで離散化されたコントローラを設計する。

4. シミュレーション
   初期状態を設定し、Jamox のシミュレーション開始のボタンを押すことでシミュレーションを開始すると、結果がグラフ表示される。また、得られたデータはファイルへと保存することもできる。

5. 制御実験
   Controller ブロックからリアルタイム制御実験のためのコードを生成でき、容易にシミュレーションのプロセスから実験プロセスへと移行できる。

   実験で得られたデータは、ファイル入力ブロックを用いることで、Jamox 上でグラフを表示することができる。また、シミュレーション結果と重ね合わせて表示することもでき、シミュレーションとの比較も容易に行える。シミュレーション結果は、図 12 に示すようなブロック線図を用いて行うことができる。

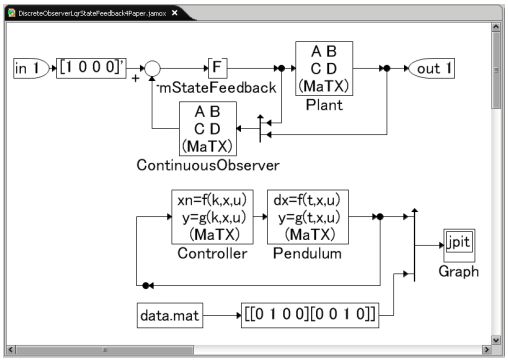
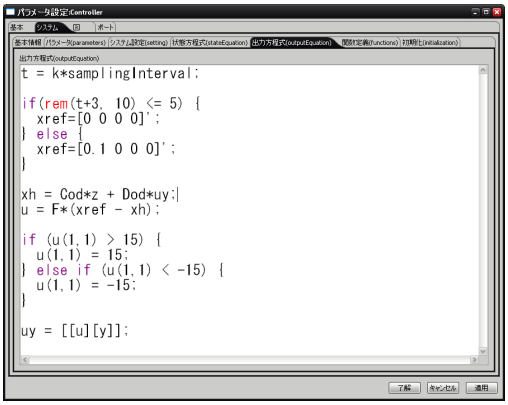


図 12: シミュレーション結果と実験結果の比較

3.3 安定化制御実験例

本システムの性能評価として、倒立振子の安定化制御実験をブロック線図に基づいて行った。このときの Controller ブロックの設計例を図 13 に示す。



```
t = k*samplingInterval;

if(rem(t+3, 10) <= 5) {
  xref=[0 0 0 0]';
} else {
  xref=[0.1 0 0 0]';
}

xh = Cod*z + Dod*uy;
u = F*(xref - xh);

if (u(1,1) > 15) {
  u(1,1) = 15;
} else if (u(1,1) < -15) {
  u(1,1) = -15;
}

uy = [[u][y]];
```

図 13: Controller の設計例

実験は、台車の目標位置を 2 秒のオフセットの後に 5 秒おきに 0.1[m] 変化させる条件の下で行った。実験の結果とシミュレーション結果を重ね合わせたグラフをを図 14、図 15 に示す。

グラフから、台車の位置は約 0.02[m]、振子の角度は約 0.04[rad] の誤差範囲内で安定化できていることが分かる。

4 まとめ

本研究では、制御工学学習を効率的に行うためにブロック線図を用いてリアルタイム制御実験用のコードを生成できるシステムを開発した。GUI 操作によりブロック線図からリアルタイム制御実験用のコードを生成することができ、設計フェーズから実験フェーズへの移行を容易に行うことができる。。

参考文献

[1] MATLAB. http://www.cybernet.co.jp/matlab/.

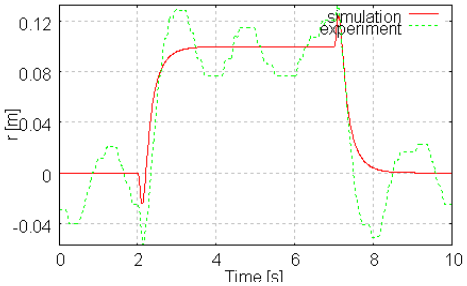


図 14: 台車の応答波形

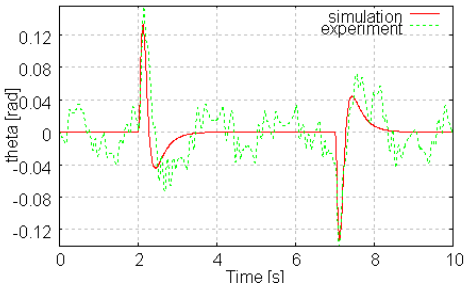


図 15: 振子の応答波形

[2] 大川善邦. Simulink と Real-Time Workshop を使った MATLAB による組み込みプログラミング入門. CQ 出版社, 2005.

[3] 森友一郎, 薬師輝久, 馬場秀忠. RTLinux リアルタイム処理プログラミングハンドブック. 秀和システム, 2000.

[4] 松本明紘, 古賀雅伸. ブロック線図に基づく制御系設計支援プラットフォームの開発. 第 51 回システム制御情報学会研究発表講演会, 2007.

[5] 岸田和也, 古賀雅伸. Linux のリアルタイム制御パッケージを用いた実験環境構築システム. 第 5 回情報科学技術フォーラム, 2006.

[6] KNOPPIX Japanese edition. http://www.rcis.aist.go.jp/project/knoppix/.

[7] 田中俊行, 古賀雅伸. スクリプトインタフェースに基づく 数値計算エンジンの再利用性向上. 第 6 回情報科学技術フォーラム, 2007.

[8] 古賀雅伸. Linux・Windows でできる MATX による数値計算. 東京電機大学出版, 2000.

[9] 山田賢治, 古賀雅伸. グラフのインタラクティブ修正に基づく制御系設計支援ツール. 第 52 回 システム制御情報学会研究発表講演会, 2008.