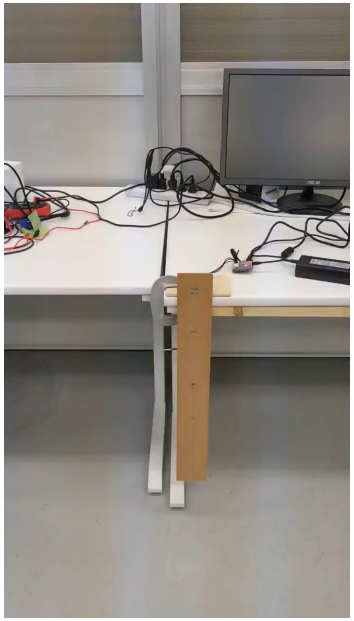
CasADiとDynamixelで振子の振り上げモデル予測制御デモ

ROSConJP2023 LT

2023/09/26

株式会社 Proxima Technology

概要



- モデル予測制御を用いて振子の振り上げをするデモを作ってみました
- https://github.com/proxima-technology/mpc_pendulum

設定

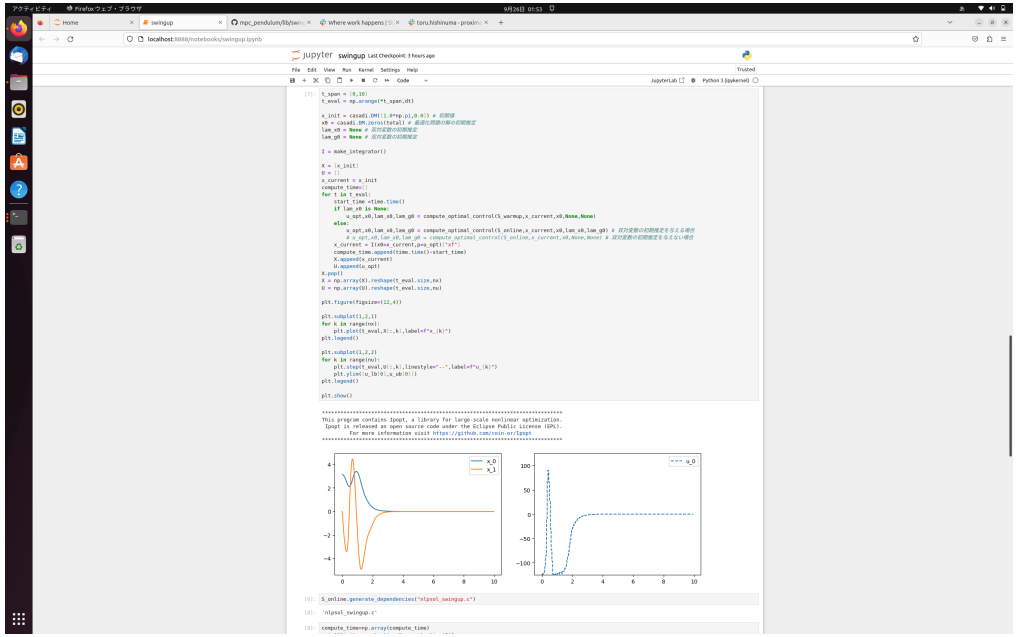
- ハードウェア
  - Dynamixel XM430-W350 (電流制限して利用)
  - 振子（木の板）
  - ノートPC
- ソフトウェア
  - ROS2 humble
  - Dynamixel SDK
  - CasADi

CasADiとは

- ルーヴェン大学のグループが中心となって開発した非線形最適化・最適制御のためのオープンソース・ソフトウェア・フレームワーク
- https://github.com/casadi/casadi
- 特徴
  - C++/Python/Matlab
  - 様々な最適化ソルバーへのインターフェース
  - Cコード生成
  - 自動微分

今回デモ中でのCasADi利用方法

- ①制御ロジック開発（Python）→Cコード生成＆コンパイル
- ②制御ロジック組み込み（C++）



可視化が得意なPythonでSim挙動を眺めつつコスト等を検討

今回デモ中でのCasADi利用方法

- ①制御ロジック開発（Python）→Cコード生成＆コンパイル
- ②制御ロジック組み込み（C++）

```
// [MPC] initialize
RCLCPP_INFO(rclcpp::get_logger("do_experiment_node"), "mpc initialize start.");
Dict opts;
opts["print_time"] = false;
opts["ipopt"] = Dict{{"print_level", 0}};
std::string lib_nlpsol_swingup_path = std::getenv("COLCON_PREFIX_PATH");
lib_nlpsol_swingup_path += "/../src/mpc_pendulum/lib/nlpsol_swingup.so";
Function solver_warmingup = nlpsol("solver", "ipopt", lib_nlpsol_swingup_path, opts);
opts["ipopt"] = Dict{{"print_level", 0},{"max_iter",3}};
Function solver_online = nlpsol("solver", "ipopt", lib_nlpsol_swingup_path, opts);

int nx = 2; // state dimension
int nu = 1; // control dimension
int nh = 50; // horizon
std::vector<double> x0(nx+(nx+nu)*nh, 0);
std::vector<double> xmin(nx+(nx+nu)*nh, -std::numeric_limits<double>::infinity());
std::vector<double> xmax(nx+(nx+nu)*nh, +std::numeric_limits<double>::infinity());
xmin.at(0) = measured_position_rad; // current state
xmax.at(0) = measured_position_rad; // current state
xmin.at(1) = measured_velocity_radpersec; // current state
xmax.at(1) = measured_velocity_radpersec; // current state
for(int i=0;i<nh;i++)
{
  xmin.at(nx*(nh+1)+i) = -current_max_mA; // control limit
  xmax.at(nx*(nh+1)+i) = current_max_mA; // control limit
}
std::vector<double> gmin(nx*nh, 0), gmax(nx*nh, 0);
std::map<std::string, DM> arg = {{"lbx", xmin}, {"ubx", xmax}, {"lbg", gmin}, {"ubg", gmax}, {"x0",  0}};
double optimization_start_time = ros_clock.now().nanoseconds()/1.0e9;
auto res = solver_warmingup(arg); // solving optimal control problem (warmup)
auto res_x = res.at("x");
auto res_lam_x0 = res.at("lam_x");
auto res_lam_g0 = res.at("lam_g");
double command_current_mA = (double) res_x(nx*(nh+1));
double optimization_time = ros_clock.now().nanoseconds()/1.0e9 - optimization_start_time;
RCLCPP_INFO(rclcpp::get_logger("do_experiment_node"), "mpc warmup optimization time %f [sec]",optimization_time);
RCLCPP_INFO(rclcpp::get_logger("do_experiment_node"), "mpc initialize end.");
```

コード生成&コンパイルした
制御ロジック（最適化ソルバー）を読み込み

最適化ソルバーに与える変数・制約条件を定義

最適化の実行

ROSにも簡単に組み込める！