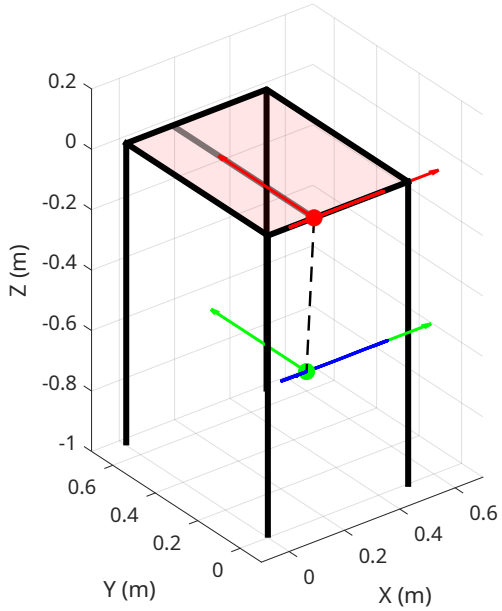
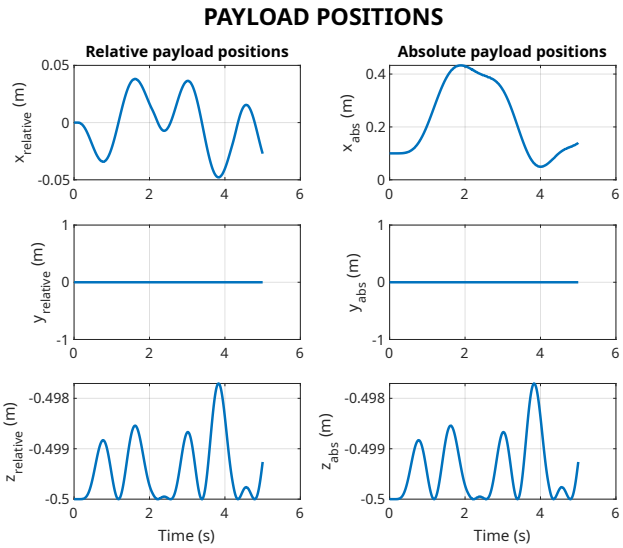
Crane3DSim V2.1: 3D Overhead Crane Simulator

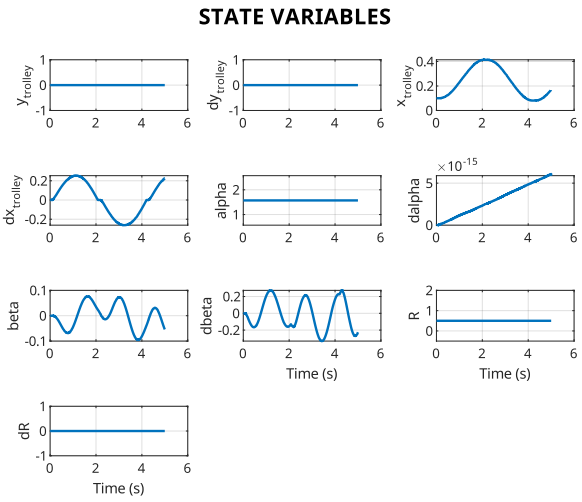
J. Vicente-Martinez, E. Ramirez-Laboreo

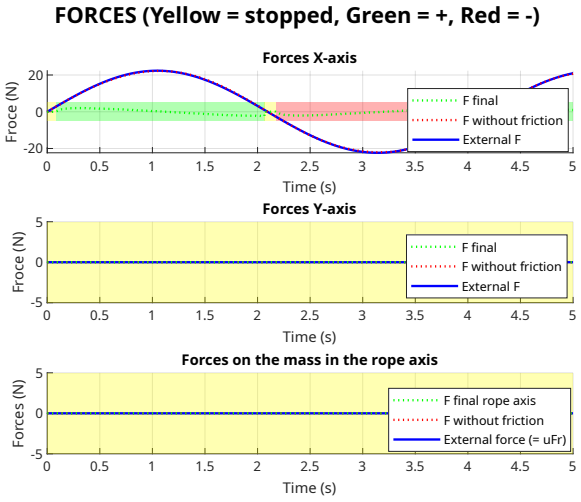
Abstract

This document provides a comprehensive guide to the `Crane3DSim`, a MATLAB-based 3D crane simulator. It covers its two main variants: one for continuous input signals and another for discrete control applications. The document details the setup, usage, and key functionalities of the simulator, aiming to facilitate its understanding and application for both simulation and control design purposes. It includes in-depth analysis of the provided MATLAB scripts and functions.









Contents

1 Change log

1.1 V2.1

- The errors in the rope tension when the rope-axis has no novement has been corrected.
  Updated `Crane3DSim_1state_fast.m`, `Crane3DSim_disc.m`, `Crane3DSim_cont.m`
- Fixed a BUG on the viscous friction $KAIRE\_B$ on the dynamics of the X-axis and simplified the checking of the state on the dynamics.
  Updated `Crane3DSim_disc.m`, `Crane3DSim_cont.m`, `Crane3DSim_1state_fast.m`

1.2 V2.0

- The errors in the equations describing the motion of the cart along the x and y axes have been corrected (the effect of viscous friction at the corners was missing).
  Updated `Crane3DSim_1state_fast.m`, `Crane3DSim_disc.m`, `Crane3DSim_cont.m`
- Fixed a BUG on the rope axis checks after an event
  Updated `Crane3DSim_disc.m`, `Crane3DSim_cont.m`
- Changes on Force graphs
  Updated `Crane3DSim_cont.m`

2 Introduction

This document serves as the documentation for the **Crane3DSim V2.1**, a MATLAB-based simulator designed to model the 3D dynamics of an overhead crane. The simulator is provided in two main variants:

- A simulator variant for continuous input functions.
- A simulator variant for discrete input functions, suitable for controller implementation.

3 Simulator Variants Overview

The `Crane3DSim` offers two distinct simulation environments,

3.1 Continuous Input Simulator – `Crane3DSim_cont.m`

This variant of the simulator is designed to perform the full dynamic simulation of the crane using an Ordinary Differential Equation (ODE) solver with event detection capa-

bilities. It is suitable for scenarios where the control signals (motor forces) are defined as continuous functions of time.

3.1.1 Key Features

- Simulates crane dynamics using an ODE with event detection.
- Requires motor force signals (in PWM [-1 to 1]) to be defined as time-dependent functions, typically using MATLAB function handles @(t).
- Generates detailed plots of state variables, forces, and a 3D animation of the crane's movement.

3.1.2 Launching the Simulation

The continuous simulation is launched using the `Launcher_cont.m` script. Simulation parameters can be directly configured within this file:

- **Physical Properties**: `mc` (payload mass [kg]), `mw` (trolley mass [kg]), `ms` (rail mass [kg]), `g` (gravity [m/s^2]).
- **Physical Limits (`ph_limits`)**: Defines the operational boundaries for the trolley (X and Y axes) and the rope length. These are `xlim_positive`, `xlim_negative`, `ylim_positive`, `ylim_negative`, `rlim_positive`, `rlim_negative` All in [m].
- **Initial Conditions (`x0`)**: The initial state vector of the crane (see Section 5 for details).
- **Simulation Time (`tfin`)**: Total simulation duration in seconds.
- **Input Forces (`uPWM_x`, `uPWM_y`, `uPWM_z`)**: Time-dependent functions (e.g., `uPWM_x = @(t) 0.8 * amp * sin(frec * t + phase) + bias)` defining the PWM [-1 to 1] applied to each motor.

> You can apply input force signal functions directly to [N]. To do this, define your function on [N] and use the conversion constants `X_PWM_TO_F`, `Y_PWM_TO_F` and `Z_PWM_TO_F` to convert to PWM. Alternatively, you can modify the `Crane3DSim_cont.m` simulation script in the `FORCE INPUTS` section to remove the multiplication for these constants and use [N] input forces instead.

```
% Example for continuous PWM input definition in Launcher_cont.m
amplitud = 0.5; % Example amplitude
bias = 0.1;     % Example bias
frecuencia_rad_seg = 1.50; % Example frequency

uPWM_x = @(t) 0.8 * amplitud * sin(frecuencia_rad_seg * t) + bias;
uPWM_y = @(t) 0; % No input on y-axis for this example
uPWM_z = @(t) 0; % No input on rope-axis for this example
```

An identification parameters file (e.g., `Example_identification.mat`) must be loaded by this launcher script, which provides conversion constants and friction parameters (see Section 4.1 to know the variables that this `.mat` must have). After configuration, simply running `Launcher_cont.m` executes the simulation and displays the results.

Important Note: No modifications are required within the `Crane3DSim_cont.m` function itself. All the changes must be done on the `Launcher_cont.m` file. At first, it comes pre-configured with example data, allowing for immediate execution.

3.1.3 Function Inputs of `Crane3DSim_cont.m`

```
function [T_out DATA_OUT ESTADO_OUT] = Crane3DSim_cont(frictions,IMOT,
    IMOTx,IMOTy,ks,uPWM_x,uPWM_y,uPWM_z,masas,x0,tsimul,g,ph_limits)
```

The `Crane3DSim_cont.m` function expects the following input arguments:

- `frictions`: A structure containing various friction parameters. Its fields are:
  - `frictions.Tsx_positive`: Coefficient of static friction of the trolley on axis +X (function dependent on position).
  - `frictions.Tsx_negative`: Coefficient of static friction of the trolley on axis -X (function dependent on position).
  - `frictions.Tsy_positive`: Coefficient of static friction of the trolley on axis +Y (function dependent on position).
  - `frictions.Tsy_negative`: Coefficient of static friction of the trolley on axis -Y (function dependent on position).
  - `frictions.Tsr_positive`: Coefficient of static friction of the trolley on axis +L (function dependent on position).
  - `frictions.Tsr_negative`: Coefficient of static friction of the trolley on axis -L (function dependent on position).
  - `frictions.Tdy_positive`: Coefficient of dynamic friction of the trolley on axis +Y.
  - `frictions.Tdy_negative`: Coefficient of dynamic friction of the trolley on axis -Y.
  - `frictions.Tdx_positive`: Coefficient of dynamic friction of the trolley on axis +X.

  - `frictions.Tdx_negative`: Coefficient of dynamic friction of the trolley on axis -X.
  - `frictions.Tdr_positive`: Coefficient of dynamic friction of the trolley on the +rope axis.
  - `frictions.Tdr_negative`: Coefficient of dynamic friction of the trolley on the -rope axis.
  - `frictions.K_AIREA`: Coefficient of dynamic friction of the alpha angle.
  - `frictions.K_AIREB`: Coefficient of dynamic friction of the beta angle.

- `IMOT`, `IMOTx`, `IMOTy`: Equivalent inertia for rope, X-axis, and Y-axis respectively.
- `ks`: A vector `[X_PWM_TO_F Y_PWM_TO_F Z_PWM_TO_F]` containing constants to convert PWM to newton force.
- `uPWM_x`, `uPWM_y`, `uPWM_z`: Function handles `@(t)` for continuous PWM input signals for X, Y, and rope motors.
- `masas`: A vector `[mc mw ms]` containing payload, trolley, and rail masses.
- `x0`: Initial state vector (see Section 5).
- `tsimul`: Total simulation time.
- `g`: Gravity constant.
- `ph_limits`: Physical limits vector (as defined in `Launcher_cont.m`).

Note on Friction Units: The simulator is designed to receive friction parameters in units derived from the identification process (effectively in PWM-scaled units, as generated by `ParametersMATgenerator.m`). If you wish to introduce friction parameters directly in International System (SI) units (Newtons for forces, N·s/m for dynamic friction coefficients), you must remove the respective multiplications by `X_PWM_TO_F`, `Y_PWM_TO_F`, and `Z_PWM_TO_F` in the `FRICTION` section of the `Crane3DSim_cont.m` file.

3.1.4 Function Outputs of `Crane3DSim_cont.m`

The function returns the following outputs:

- `T_out`: Time vector of the simulation results.
- `DATA_OUT`: Matrix containing all state variables and derived quantities at each time step (see Section 5).
- `ESTADO_OUT`: (Potentially for internal use/debugging) Provides the state of static friction and movement for each axis.

3.2 Discrete Input Simulator – `Crane3DSim_disc.m`

This simulator variant is designed to emulate how control would be implemented in a real-world discrete system. It features an outer loop that executes at a specified sampling period (`T_sample`), within which the control action remains constant, and the system's dynamics are simulated continuously using an ODE solver. This setup is ideal for testing discrete-time controllers.

3.2.1 Key Features

- Simulates 3D crane dynamics with discrete control signals applied at regular intervals.
- Includes a dedicated section within the code for users to implement their own Feedback (FB) or Feedforward (FF) controllers.
- The provided example within the simulator's code demonstrates a PI controller for the trolley's position, illustrating how to integrate a reference signal (`refs`) for control.
- Generates detailed plots of state variables, forces, and a 3D animation of the crane's movement.

3.2.2 Launching the Simulation

The discrete simulation is launched using the `Launcher_disc.m` script. Within this file, users can configure:

- **Physical Properties**: `mc`, `mw`, `ms`, `g` (same as continuous variant).
- **Physical Limits (`ph_limits`)**: Same as continuous variant.
- **Initial Conditions (`x0`)**: The initial state vector of the crane (see Section 5).
- **Simulation Time (`tfin`)**: Total simulation duration in seconds.
- **Sampling Time (`T_sample`)**: The discrete time step at which control actions are updated.
- **Input Forces (`uPWM_x_matriz`, `uPWM_y_matriz`, `uPWM_r_matriz`)**: Column vectors where each element specifies the PWM action [-1 to 1] to apply at each `T_sample` interval.

  > You can apply input force signal functions directly to [N]. To do this, define your matrix on [N] and use the conversion constants `X_PWM_TO_F`, `Y_PWM_TO_F` and `Z_PWM_TO_F` to convert to PWM. Alternatively, you can modify the `Crane3DSim_disc.m` simulation script in the `FORCE INPUTS` section to remove the multiplication for these constants and use [N] input forces instead.

- **Reference Signals (`refs`)**: A matrix containing reference values for controlled variables, synchronized with `T_sample`.

An identification parameters file (e.g., `Example_identification.mat`) is also loaded. After running, the simulation executes and generates graphs of state variables, forces, and a 3D animation.

Important Note: The `Launcher_disc.m` script includes example data and can be run immediately. To implement custom feedback controllers, users are expected to modify the designated section within `Crane3DSim_disc.m` itself.

3.2.3 Function Inputs of `Crane3DSim_disc.m`

```
function [T_out DATA_OUT ESTADO_OUT] = Crane3DSim_disc(fricciones,IMOT,
    IMOTx,IMOTy,ks,uPWM_x_matriz,uPWM_y_matriz,uPWM_r_matriz,masas,x0,
    tsimul,T_sample,g,ph_limits,refs)
```

The `Crane3DSim_disc.m` function expects the following input arguments:

- `fricciones`: A structure containing various friction parameters. Its fields are identical to those described for `Crane3DSim_cont.m`.
- `IMOT`, `IMOTx`, `IMOTy`: Equivalent inertia.
- `ks`: A vector `[X_PWM_TO_F Y_PWM_TO_F Z_PWM_TO_F]`.
- `uPWM_x_matriz`, `uPWM_y_matriz`, `uPWM_r_matriz`: Matrices defining discrete PWM inputs over time.
- `masas`: A vector `[mc mw ms]` containing masses.
- `x0`: Initial state vector (see Section 5).
- `tsimul`: Total simulation time.
- `T_sample`: Discrete sampling period.
- `g`: Gravity constant.
- `ph_limits`: Physical limits vector.
- `refs`: A matrix of reference values for control, synchronized with `T_sample`.

Note on Friction Units: Similar to the continuous simulator, if you wish to introduce friction parameters in International System (SI) units (Newtons, N·s/m), you must remove the respective multiplications by `X_PWM_TO_F`, `Y_PWM_TO_F`, and `Z_PWM_TO_F` in the `FRICTION` section of the `Crane3DSim_disc.m` file.

3.2.4 Function Outputs of `Crane3DSim_disc.m`

The function returns the same outputs as the continuous version:

- `T_out`: Time vector of the simulation results.
- `DATA_OUT`: Matrix containing all state variables and derived quantities at each time step (see Section 5).
- `ESTADO_OUT`: (Potentially for internal use/debugging) Provides the state of static friction and movement for each axis.

4 Auxiliary Files

4.1 `ParametersMATgenerator.m`

This file generates the system parameter data based on the parametric estimation described in the paper associated with this work. If a different estimation is used, or if the physical data for friction, inertia, or transformation constants are known, they can be modified directly by hand in the Parameters calculation section. They can also be included directly in the launcher file using variables with the same name as those generated by this file.

This script is essential for generating a MATLAB `.mat` file (e.g., `Example_identification.mat`) that contains all the necessary parameters derived from the crane's identification process, as described in the associated paper. Users can input various physical and friction-related parameters, and the script calculates and saves the following key variables:

- `X_PWM_TO_F`: Constant to convert PWM signals to force on the X-axis (N/PWM).
- `Y_PWM_TO_F`: Constant to convert PWM signals to force on the Y-axis (N/PWM).
- `Z_PWM_TO_F`: Constant to convert PWM signals to force on the rope axis (N/PWM).
- `fsecaXPOSITIVO_fnc`, `fsecaXNEGATIVO_fnc`, `fsecaYPOSITIVO_fnc`, `fsecaYNEGATIVO_fnc`: Function handles for static friction, dependent on position.
- `fsecaXPOSITIVO_coefs`, `fsecaXNEGATIVO_coefs`, `fsecaYPOSITIVO_coefs`, `fsecaYNEGATIVO_coefs`: Coefficients of the fitted dry friction functions.
- `Tsr_positivo`, `Tsr_negativo`: Static friction forces for rope motor (+/- direction) [N].
- `Tdy_positive`, `Tdy_negative`, `Tdx_positive`, `Tdx_negative`, `Tdr_positive`, `Tdr_negative`: Dynamic friction coefficients for Y, X, and rope axes respectively.
- `IMOTr`, `IMOTx`, `IMOTy`: Equivalent inertia for rope, X-axis, and Y-axis respectively.

- K_AIREA, K_AIREB: Dynamic friction coefficients for alpha and beta angles.

These parameters are crucial for the accurate dynamic behavior of the crane model.

4.2 Crane3DSim_1state_fast.m

This is a secondary function used internally by both Crane3DSim_cont.m and Crane3DSim_disc.m functions. Its primary purpose is to evaluate the system constraints and detect events (like reaching physical limits or changing friction states) just before starting each ODE integration step. It performs a single, approximated simulation step to check for any events and derive the new states, which are then used as accurate initial conditions for the subsequent ODE solver call. This helps ensure precise event handling and state transitions.

4.2.1 Inputs

The function receives numerous inputs, primarily related to the current state, control actions, and physical/friction parameters:

- Tdiscret: A small discretization period used for the approximation step.
- X_k: The previous full state vector.
- uF_x_k1, uF_y_k1, uF_r_k1: The control force inputs (in Newtons) for X, Y, and rope motors at the next instant. These are derived from the PWM inputs using ks constants.
- mw, ms, mc: Masses of trolley, rail, and payload.
- Tsx_fun_positivo, Tsx_fun_negativo, Tsy_fun_positivo, Tsy_fun_negativo, Tsr_fun_positivo, Tsr_fun_negativo: Function handles for static friction.
- Tdy_positivo, Tdy_negativo, Tdx_positivo, Tdx_negativo, Tdr_positivo, Tdr_negativo: Dynamic friction coefficients.
- IMOT, IMOTx, IMOTy: Motor inertias.
- g: Gravity constant.

4.2.2 Outputs

The function returns updated state flags and the refined initial state for the next ODE step:

- qx_new, qy_new, qr_new: Flags indicating the new movement state (e.g., 0 for static, 1 for positive movement, -1 for negative movement) for X, Y, and rope axes respectively.
- X_k1: The refined new state vector that should be used as initial conditions for the next ODE solver call, ensuring continuity and correct event handling.
- T_k1: The time instant at k+1, typically T_k + Tdiscret.

Note: Users should generally not modify this file as it is an internal utility for the main simulation functions.

4.3 Example_identification.mat

This `.mat` file contains the pre-calculated identification parameters generated by `ParametersMATgenerator.m`. It is loaded by both `Launcher_cont.m` and `Launcher_disc.m` to provide the necessary physical and friction constants for the crane model.

4.4 Example_tray_disc.mat

This `.mat` file contains example data specifically for the discrete simulator. It includes predefined force inputs (`uPWM_x_matriz`, `uPWM_y_matriz`, `uPWM_r_matriz`) and corresponding reference signals (`refs`) to demonstrate a typical trajectory for the discrete control scenario. This file is loaded by `Launcher_disc.m` to run an example simulation.

5 System State Variables

Both `Crane3DSim_cont.m` and `Crane3DSim_disc.m` use a consistent state vector `x0` for initial conditions and output `DATA_OUT` which represents the system's state variables.

5.1 Initial Conditions (x0)

The initial state vector `x0` is a 10-element column vector structured as follows:

- `x0(1)`: Initial trolley position on y-axis [m] (`y0`).
- `x0(2)`: Initial trolley velocity on y-axis [m/s] (`dy0`).
- `x0(3)`: Initial trolley position on x-axis [m] (`x0`).
- `x0(4)`: Initial trolley velocity on x-axis [m/s] (`dx0`).
- `x0(5)`: Initial alpha angle [rad] (`alpha0`).
- `x0(6)`: Initial alpha angular velocity [rad/s] (`dalpha0`).
- `x0(7)`: Initial beta angle [rad] (`beta0`).
- `x0(8)`: Initial beta angular velocity [rad/s] (`dbeta0`).
- `x0(9)`: Initial rope length [m] (`R0`).
- `x0(10)`: Initial rope velocity [m/s] (`dR0`).

5.2 Output Data (DATA_OUT)

The output DATA_OUT from the simulation functions typically contains a matrix where each column corresponds to a specific variable over time:

- Column 1: yw (Trolley y-axis position [m])
- Column 2: dyw (Trolley y-axis velocity [m/s])
- Column 3: ddyw (Trolley y-axis acceleration [m/s^2])
- Column 4: xw (Trolley x-axis position [m])
- Column 5: dxw (Trolley x-axis velocity [m/s])
- Column 6: ddxw (Trolley x-axis acceleration [m/s^2])
- Column 7: alpha (Alpha angle position [rad])
- Column 8: dalpha (Alpha angle velocity [rad/s])
- Column 9: ddalpha (Alpha angle acceleration [rad/s^2])
- Column 10: beta (Beta angle position [rad])
- Column 11: dbeta (Beta angle velocity [rad/s])
- Column 12: ddbeta (Beta angle acceleration [rad/s^2])
- Column 13: R (Rope length [m])
- Column 14: dR (Rope velocity [m/s])
- Column 15: ddR (Rope acceleration [m/s^2])
- Column 16: yc (Load y-axis position [m])
- Column 17: dyc (Load y-axis velocity [m/s])
- Column 18: ddyc (Load y-axis acceleration [m/s^2])
- Column 19: zc (Load z-axis position [m])
- Column 20: dzc (Load z-axis velocity [m/s])
- Column 21: ddzc (Load z-axis acceleration [m/s^2])
- Column 22: xc (Load x-axis position [m])
- Column 23: dxc (Load x-axis velocity [m/s])
- Column 24: ddxc (Load x-axis acceleration [m/s^2])

6 Physical Parameters and Limits

Both `Launcher_cont.m` and `Launcher_disc.m` define key physical characteristics and operational limits for the crane:

- **Masses**:
  - `mc`: Payload mass [Kg] (e.g., 0.457 Kg).
  - `mw`: Trolley mass [Kg] (e.g., 1.155 Kg).
  - `ms`: Rail mass [Kg] (e.g., 2.20 Kg).
  - `g`: Gravity [m/s^2] (e.g., 9.81 m/s^2).
- **Physical Limits (`ph_limits`)**: A vector defining the operational boundaries for the trolley and the rope. These values prevent the crane components from exceeding realistic physical constraints.
  - `xlim_positive`: Maximum X-axis coordinate of the trolley [m].
  - `xlim_negative`: Minimum X-axis coordinate of the trolley [m].
  - `ylim_positive`: Maximum Y-axis coordinate of the trolley [m].
  - `ylim_negative`: Minimum Y-axis coordinate of the trolley [m].
  - `rlim_positive`: Maximum rope length [m].
  - `rlim_negative`: Minimum rope length [m].

7 Example Usage

To get started quickly, you can run the `Launcher_cont.m` or `Launcher_disc.m` scripts directly. They are pre-configured with example input signals and initial conditions. You can modify the parameters within these launcher files, and into the simulation scripts to experiment with different scenarios and implement your own control strategies.