The Worst Toolbox Documentation

Stephanie Tsuei

Version 0.1a

1 Introduction

This toolbox implements the algorithm described in:

> J. E. Tierno, R. M. Murray, J. C. Doyle, and I. M. Gregory, "Numerically Efficient Robustness Analysis of Trajectory Tracking for Nonlinear Systems," *J. Guid. Control. Dyn.*, vol. 20, no. 4, pp. 640-647, Jul. 1997.

which uses theory presented in:

> A. Bryson and Y. Ho, *Applied optimal control: optimization, estimation, and control*, vol. 59, no. 8. 1975.

1.1 The Robust Trajectory Tracking Problem

Consider the dynamical system pictured in Figure 1.1 with dynamics

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{U}(t), \mathbf{u}(t), \mathbf{v}(t), \boldsymbol{\delta})$$

and outputs

$$\begin{aligned} \mathbf{Y}(t) &= g(\mathbf{x}(t), \mathbf{U}(t), \mathbf{u}(t), \mathbf{v}(t), \boldsymbol{\delta}) \\ \mathbf{z}(t) &= h(\mathbf{x}(t), \mathbf{U}(t), \mathbf{u}(t), \mathbf{v}(t), \boldsymbol{\delta}) \end{aligned}$$

where $\mathbf{x}(t)$ is the state, $\mathbf{U}(t)$ is a nominal input signal, $\mathbf{u}(t)$ is a disturbance signal, δ is a vector of uncertain parameters, Δ is an uncertain block of unit norm, and $\mathbf{v}(t)$ and $\mathbf{z}(t)$ are signals representing any possible unmodeled dynamics. Therefore, the nominal trajectory, is given by

$$\mathbf{Y}_n(t) = g(\mathbf{x}(t), \mathbf{U}(t), 0, 0, \boldsymbol{\delta}_n)$$

where $\boldsymbol{\delta}_n$ are the nominal values of the uncertain parameters and the quantity $\mathbf{y}(t) = \mathbf{Y}(t) - \mathbf{Y}_n(t)$ is the error signal.

We make the following assumptions:

- $\|\mathbf{v}(t)\| = \|\mathbf{z}(t)\|$
- $\mathbf{v}(t)$ and $\mathbf{z}(t)$ can be written in block form $\mathbf{v}(t) = \begin{bmatrix} \mathbf{v}_1(t) & \mathbf{v}_2(t) & \dots & \mathbf{v}_p(t) \end{bmatrix}^T$ and $\mathbf{z}(t) = \begin{bmatrix} \mathbf{z}_1(t) & \mathbf{z}_2(t) & \dots & \mathbf{z}_p(t) \end{bmatrix}^T$. The total number of blocks in $\mathbf{v}(t)$ and $\mathbf{z}(t)$ must be the same, but $\mathbf{v}_i(t)$ and $\mathbf{z}_i(t)$ can be of different dimensions.
- $\|\mathbf{u}(t)\|$ is a known constant scalar M if there is only one disturbance input. If there are multiple disturbance inputs $\mathbf{u}_1(t)$, $\mathbf{u}_2(t)$, then each disturbance input i has a known norm M_i.

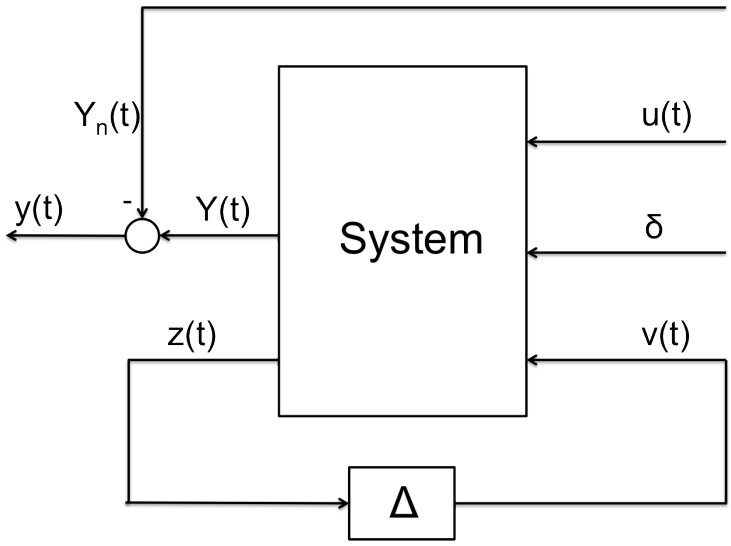


Figure 1: A block diagram dipicting the robust trajectory generation problem.

- $\underline{\boldsymbol{\delta}} \leq \boldsymbol{\delta}_n \leq \bar{\boldsymbol{\delta}}$ (elementwise) where $\underline{\boldsymbol{\delta}}$ and $\bar{\boldsymbol{\delta}}$ are known constants.
- We are only interested in the finite time interval (t_i, t_f) and for a vector-valued time signal $\mathbf{w}(t)$,

$$\|\mathbf{w}(t)\| = \left(\frac{1}{t_f - t_i}\int_{t_i}^{t_f} \mathbf{w}^T(t)\mathbf{w}(t)dt\right)^{\frac{1}{2}}$$

Given the information above, this toolbox computes the functions $\mathbf{u}(t)$, $\mathbf{v}(t)$ and the value of $\boldsymbol{\delta}(t)$ that *maximizes* the quantity below:

$$J = \|\mathbf{y}(t)\| = \left(\frac{1}{t_f - t_i}\int_{t_i}^{t_f} \mathbf{y}(t)^T\mathbf{y}(t)dt\right)^{\frac{1}{2}}$$

1.2 Notes on the Algorithm

1. This algorithm is a power algorithm that is not a contraction. Therefore, it is not guaranteed to converge, but does in practice.
2. The algorithm searches for local extrema (local minima *and* local maxima). Therefore the value of J that is computed is a lower bound on the actual value of J.
3. Initial values for the worst case disturbance are randomly generated and the algorithm is sensitive to the initial value of the disturbance. Therefore, it is helpful to run the algorithm several times and plot a histogram of the values that it computes, as seen in the examples.
4. Let $\mathbf{y}_1(t) = \begin{bmatrix} y_{1,1}(t) & y_{1,2}(t) & \dots & y_{1,m}(t) \end{bmatrix}$ be the error output signal from the previous iteration and $\mathbf{y}_2(t) = \begin{bmatrix} y_{2,1}(t) & y_{2,2}(t) & \dots & y_{m,2}(t) \end{bmatrix}$ be the error output signal from the current iteration. The algorithm declares that it has converged when

$$\max_{i=1,2,\dots,m} \frac{\sum_{j=1}^{T}(y_{1,i}(j) - y_{2,i}(j))^2}{\sum_{j=1}^{T} y_{1,i}(j)^2} < e$$

where e is a small positive number and T is the number of timesteps.

1.3 System Requirements

Requirements:

- Matlab R2012b or later
- Simulink

The toolbox may work on some earlier versions of Matlab and Simulink, but was developed on Matlab R2012b.

1.4 Install Instructions

To install the Worst Toolbox, place the folder `worst/` anywhere on your local machine and add it (but not its subdirectories) to the Matlab path. This is equivalent to the running the command:

```
addpath('/path/to/robust/worst')
```

2 Using Worst

2.1 Simulink Model Configuration

The Worst toolbox makes some assumptions about the configuration of Simulink models:

Order of Inputs

The program assumes that models have up to four input signals and two output signals. Each of the signals may have arbitrary dimension to accomodate multiple inputs, disturbances, and feedback signals. The order assumed is:

1. Nominal inputs, $\mathbf{U}(t)$
2. Disturbance inputs, $\mathbf{u}(t)$
3. Unmodeled feedback inputs, $\mathbf{v}(t)$
4. Uncertain Parameters, $\boldsymbol{\delta}$

Order of Outputs

Models are allowed to have an arbitrary number of outputs. If the Simulink model has a total of r outputs, and the first $m \leq r$ outputs are outputs included in computing the performance measure J, then the remaining $r - m$ outputs are unmodeled feedback outputs; the coordinates of z.

2.2 Syntax of `worst`

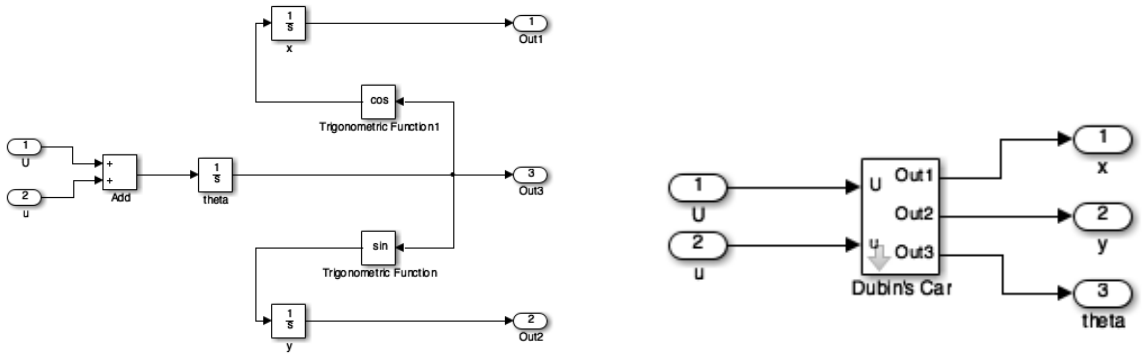
The syntax for `worst` is:

```
worst(system_name, output_dimension, 'Parameter1', value1, 'Parameter2', value2, ...)
```

`system_name` is the name of the Simulink model (make sure the model is in the MATLAB path!) and `output_dimension` is the dimension of the system's output. The parameters are optional values, described in the table below:

Name	Default Value	Description
ti	0	Simulation start time in seconds
tf	10	Simulation end time in seconds
params	[ ]	A p by 3 matrix where p is the number of uncertain scalar parameters. Each row has the format: [lower_bound, nominal_value, upper_bound]
disturbance_specs	[ ]	A d by 2 matrix where d is the number of disturbances. Each row has the format: [disturbance_dimension, disturbance_norm]
unmodeled_io	[ ]	A b by 2 matrix where b is the number of unmodeled input-output pairs. The left column has all the dimensions of the unmodeled inputs $\mathbf{v}$ and the right column has all the corresponding dimensions of the unmodeled outputs $\mathbf{z}$.
max_iter	40	The maximum number of times to iterate the algorithm when computing a single value of J.
error_tol	.01	The value e in section 1.2
num_iter	10	The number of different values of J to compute.
nominal_time	[ ]	The time axis, a column vector for the nominal input, which is described below.
nominal_input	[ ]	A T by q matrix describing the nominal input signal, where T is the length of nominal_time and q is the dimension of the nominal input. If a value for this parameter is specified, then a value for the nominal time axis must also be specified.
nominal_output	[ ]	A T by p matrix containing the nominal output, where T is the length of nominal_time and p is the dimension of the nominal output. If this option is specified, then the program will not compute a nominal output like it would otherwise. If this option is specified, then a value for the nominal time axis (same one that corresponds to nominal_input) must also be specified.
averaging	1 (on)	Boolean value (0 or 1) that switches filtering on and off. Sometimes the filtering will push the distribution of costs towards the largest value.
plot_cost	0 (off)	Boolean value (0 or 1). The value of the current cost will be plotted at the end of each iteration when equal to 1.
plot_d	0 (off)	Boolean value (0 or 1). Will plot the newly computed disturbance signal at the end of each iteration if set to 1.
plot_v	0 (off)	Boolean value (0 or 1). Will plot the value of the unmodeled feedback input at each iteration if set to 1.
plot_parm	0 (off)	Boolean value (0 or 1). Will plot the current value of the uncertain parameters at each iteration if set to 1.
plot_error	0 (off)	Boolean value (0 or 1). Plots the current error signal if set to 1.

Optional inputs whose default value is [ ] do not need to be specified and are not necessary for the program to run. If there are no uncertain parameters in the system, do not specify any uncertain parameters.



(a) Original Simulink Diagram

(b) Diagram with S-Function

Figure 2: A model of Dubin's Car compiled into an S-function.

2.3 Output of `worst`

The output of `worst` is a struct containing the fields:

1. `converged`: A 1 by `num_iter` array containing 0 for the calculations that did not converge and 1 for the calculations that converged.
2. `costs`: A 1 by `num_iter` array containing the final cost calculated during each iteration.
3. `time_axis`: A 1 by `num_iter` cell array. Each entry contains the time axis for the worst case disturbance and feedback inputs of a computation of γ. Each time axis is a T_i by 1 array, where T_i is the number of timesteps in computation i.
4. `v`: A 1 by `num_iter` cell array. Each entry contains the worst case unmodeled feedback input, an array with T_i rows, from a computation of γ. If there are multiple unmodeled feedback inputs, then the inputs are just concatenated horizontally into one array. The time axis entry i in `v` is `time_axis{i}`.
5. `d`: Same as for `v` above, except with the worst case disturbances.
6. `p`: A 1 by `num_iter` cell array. Each entry of the cell array contains an array containing the worst parameter values from each computation of γ.

2.4 Speeding up the Program

The program repeatedly calls `linmod` in order to build and simulate an adjoint system. This greatly slows down the program because Simulink will recompile the model with every call. To hasten computation, compile your Simulink model into an S-function before running the program, as shown in Figure 2.4. The more blocks in the original Simulink model, the greater the improvement in speed.

3 Examples

A few examples are below. Since compiled S-functions are not compatible across all systems, the Simulink models of the examples are not compiled.

3.1 A Simple Linear System

Consider the system

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}-3 & a\\ 2 & -5\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1\\ 2\end{bmatrix}u(t) + \begin{bmatrix}2\\ 1\end{bmatrix}v(t)$$
$$Y(t) = x_1(t) + v(t)$$
$$z(t) = -x_2(t) + v(t)$$

where $-1 \leq a \leq 1$ is an uncertain parameter, $u(t)$ is a norm 1 disturbance, and $v(t)$ is the unmodeled feedback input. The code for the system is below.

```
% This example computes the worst possible disturbance of a linear system
% with the following dynamics:
%
%     x1'(t) = -3*x1(t) + a*x2(t) + u(t) + 2*v(t)
%     x2'(t) = 2*x1(t) - 5*x2(t) + 2*u(t) + v(t)
%     y(t)   = x1(t) + v(t)
%     z(t)   = -x2(t) + v(t)
%
% where (x1, x2)' is the state, u(t) is a one-dimensional disturbance of norm
% 1, y(t) is the output, and v(t) and z(t) form a feedback loop that models
% uncertain dynamics; v(t) and z(t) have the same finite-time 2-norm.



clear; clc; close all;


model_name = 'linear1_model';

disturbance_specs = [1 1];
unmodeled_io = [1 1];
params = [-1 0 1];

ti_val = 0;
tf_val = 10;

output_dim = 1;

error_tol = .01;


output_struct = ...
    worst(model_name, output_dim, 'ti', ti_val, 'tf', tf_val, ...
          'disturbance_specs', disturbance_specs, 'error_tol', error_tol, ...
          'unmodeled_io', unmodeled_io, 'params', params, 'plot_cost', 1, ...
          'plot_d', 1, 'plot_v', 1, 'plot_parm', 1);

% Plot the distribution of worst values
figure(1)
hist(output_struct.costs)
```

3.2 A Linear System with Missing Pieces

Consider the system

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}-3 & a\\ 2 & -5\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}1\\ 2\end{bmatrix}u(t)$$
$$y(t) = x_1(t)$$

where $-1 \leq a \leq 1$ is a scalar uncertain parameter and $u(t)$ is a one-norm disturbance. There are no uncertain feedback loops or nominal input signals.

```
% This example computes the worst possible disturbance of a linear system
% with the following dynamics:
%
%     x1'(t) = -3*x1(t) + a*x2(t) + u(t)
%     x2'(t) = 2*x1(t) - 5*x2(t) + 2*u(t)
%     y(t)   = x1(t)
%
% where (x1, x2)' is the state, u(t) is a one-dimensional disturbance of norm
% 1, and y(t) is the output.



clear; clc; close all;


model_name = 'linear2_model';

disturbance_specs = [1 1];
params = [-1 0 1];

ti_val = 0;
tf_val = 10;

output_dim = 1;

error_tol = .01;


output_struct = ...
    worst(model_name, output_dim, 'ti', ti_val, 'tf', tf_val, ...
          'disturbance_specs', disturbance_specs, 'error_tol', error_tol, ...
          'params', params, 'num_iter', 1);

% Plot the distribution of worst values
hist(output_struct.costs)
```

3.3 Dubin's Car

The kinematics of Dubin's Car are

$$\begin{aligned}\dot{x} &= \cos(\theta) \\ \dot{y} &= \sin(\theta) \\ \dot{\theta} &= U(t) + u(t)\end{aligned}$$

where $\begin{bmatrix} x & y & \theta \end{bmatrix}^T$ is the state, $U(t)$ is a scalar nominal input, and $u(t)$ is a scalar disturbance with $\|u\| = 2$. The output of the system is the full state. There are no uncertain feedback loops.

```
% This example computes the worst possible disturbance of a Dubin's car
% with the following dynamics:
%
%     x' = cos(theta)
%     y' = sin(theta)
%     theta' = U(t) + u(t)
%
% where (x, y, theta)' is the state, U(t) is a one-dimensional nominal input
```

```
% signal and u(t) is a one—dimensional disturbance of norm 2. The system's
% output is the full state: (x, y, theta)'.



clear; clc; close all;


model_name = 'dubin';

disturbance_specs = [1 2];

ti_val = 0;
tf_val = 10;

output_dim = 3;

error_tol = .01;

nominal_input = linspace(10,0,100)';
nominal_time = linspace(0,10,100)';


output_struct = ...
    worst(model_name, output_dim, 'ti', ti_val, 'tf', tf_val, ...
          'disturbance_specs', disturbance_specs, 'error_tol', error_tol, ...
          'nominal_input', nominal_input, 'nominal_time', nominal_time, ...
          'plot_cost', 1, 'plot_d', 1, 'plot_error', 1);

% Plot the distribution of worst values
figure(1)
hist(output_struct.costs)
```