# Worst-case Simulation With the GTM Design Model

Peter Seiler, Gary Balas, and Andrew Packard

peter.j.seiler@gmail.com, balas@musyn.com

September 29, 2009

## 1 Overview

We applied worst-case simulation analysis to NASA's Generic Transport Model (GTM), a remote-controlled 5.5 percent scale commercial aircraft. NASA has constructed a high fidelity 6 degree-of-freedom Simulink model of the GTM. This model was modified to include 18 parametric uncertainties in the aerodynamic coefficients and 8 dynamic actuator uncertainties. We then used worst-case simulation to investigate the effects the parametric uncertainties in the aerodynamic coefficients.

In this report we first formulate a worst-case simulation analysis problem for nonlinear dynamical systems with uncertain parameters. We then describe how worst-case simulation analysis can be performed on uncertain Simulink models. Robust Control Toolbox (RCT) uncertain state space (USS) blocks can be used to create Simulink models with uncertain parameters. The `wcsim` function uses gradient-based optimization to approximately solve the worst-case simulation problem for uncertain Simulink models. Finally, we present worst-case simulation results for the uncertain GTM model.

## 2 Worst-case Simulation

### 2.1 Problem Formulation

For many systems, the robustness with respect to variations in the model is an important property. Worst-case simulation is a robustness analysis performed directly on a nonlinear simulation model. Consider the following nonlinear system:

$$\begin{aligned} \dot{x}(t) &= f(x(t), t, p) \\ y(t) &= h(x(t), t, p) \\ x(0) &= x_0 \end{aligned} \tag{1}$$

where $t \in \mathbb{R}$ is time, $x(t) \in \mathbb{R}^{n_x}$ is the state at time $t$, $y(t) \in \mathbb{R}^{n_y}$ is the output at time $t$, and $x_0 \in \mathbb{R}^{n_x}$ is the initial condition at time $t = 0$. $p \in P \subseteq \mathbb{R}^{n_p}$ is a constant parameter vector upon which the model depends and $P$ is the set of allowable parameter values. $f : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ is the vector field and $h : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_y}$ is the output equation.

There is a rich body of mathematical results providing technical conditions which ensure the existence and uniqueness of solutions over a time interval $[0, t_f]$. We assume that the vector field, output equation, and initial condition are such that for each $p \in P$ there exists a unique solution $x \in L_2^{n_x}[0, t_f]$ and output $y \in L_2^{n_y}[0, t_f]$. The solution and output depend on the parameter vector $p$ and where necessary we explicitly denote this dependence using the notation $x_p(t)$ and $y_p(t)$.

We are interested in studying the variation in the output with respect to the model parameters. We assume the performance of the output can be measured with a scalar quantity. Let $G : L_2^{n_y}[0, t_f] \to \mathbb{R}$ denote an objective function that quantifies the performance of the output by $G(y)$. For example, $G(y) := \|y\|_\infty$ is the peak magnitude of $y$ and $G(y) := \left[\int_0^{t_f} y^T(t) y(t) dt\right]^{1/2}$ is the $L_2$ norm. Let $C : L_2^{n_y}[0, t_f] \to \mathbb{R}^m$ denote a

constraint function that defines a set of $m$ constraints on the output. The worst-case simulation problem is:

$$\begin{aligned} &\max_{p \in P} G(y_p) \\ &\text{subject to: } y_p \text{ is the output of the nonlinear system (Equation 1)} \\ &\qquad\qquad l \leq H(y_p) \leq u \end{aligned} \tag{2}$$

$l \in \mathbb{R}^m$ and $u \in \mathbb{R}^m$ specify the lower and upper bounds on the constraint function. Maximization is without loss of generality since $\min_{p \in P} G(y_p) = -\max_{p \in P} [-G(y_p)]$.

We have made no assumptions about the objective function $G$, constraint function $H$, or how the parameter vector $p$ enters into the system dynamics and output equation. Consequently, this maximization is a computationally difficult problem to solve. In general, it is not a concave optimization and it may have many local optima that are not global optima. Our goal will be to use gradient-based optimization to find a parameter vector that achieves a local maxima. This will generally not find a global maxima. However, it does provide a means to improve upon 'bad' parameter vectors found with other heuristic methods. For example, the structured singular value $\mu$ and worst-case gain problems are robustness analysis tools for linear systems. A heuristic for the worst-case simulation problem is: 1) Linearize the nonlinear model about an operating point or trajectory to obtain a linear system with a rational dependence on the parameters. 2) Solve $\mu$ or worst-case gain problems using the linear model to obtain a set of bad parameter vectors. Frequency gridding in either of these problems will produce a worst-case parameter vector at each frequency in the grid. 3) Simulate the nonlinear system at the bad parameter vectors found with linear analysis and further investigate those that achieve the largest value of the nonlinear objective function. In contrast, gradient-based worst-case simulation is directly applied to the nonlinear system and to the nonlinear objective function. It can also be used in conjunction with linear robustness analysis. Specifically, the parameter vectors generated by linear robustness analysis can be used to seed the gradient-based optimization.

## 2.2 Matlab/Simulink Implementation

Robust Control Toolbox (RCT) Uncertain State Space (USS) blocks can be used to create models of parameter-dependent nonlinear systems within Simulink. These USS blocks model system components that have a rational dependence on uncertain real, complex, and/or linear time-invariant objects. Uncertain real parameters are specified by a nominal value and an interval of allowable values. Thus a Simulink model that has USS blocks depending on $n_p$ real parameters is in the form of the nonlinear system in Equation 1. The allowable set of parameter vectors is in the form $P := \{p \in \mathbb{R}^{n_p} \ : \ \underline{p}_i \leq p \leq \bar{p}_i, \ i = 1, \ldots, n_p\}$. More information on the uncertain objects and the USS Simulink block can be found in the Robust Control Toolbox documentation.

The objective function for a worst-case simulation is specified with an RCT `Objective Function` block. The block dialog box has a pull-down menu to select simple objective function types, e.g. $L_2$ norm or $L_\infty$ norm. Generic objective functions can be specified via an m-file interface. The dialog box also allows the user to select between maximization or minimization of the objective function. The `Objective Function` block is similar to a `To Workspace` block with the `Save Format` set to `Structure With Time`. Simulating the system will create an output variable in the workspace with all the fields generated by a `To Workspace` block: `time`, `signals`, and `blockName`. The output variable will have the additional field `objective`. The Objective function value is stored in `objective.value`.

The constraints for a worst-case simulation are specified with RCT `Constraint Function` blocks. The `Constraint Function` blocks are similar to the `Objective Function` blocks. Simple constraint functions can be selected from a pull-down menu and more complicated constraint functions can be specified via an m-file interface. The dialog box also has fields for the upper and lower bounds on the constraint. The output variable generated by a `Constraint Function` will have the following fields in addition to the normal `To Workspace` fields: `constraint.value`, `constraint.lowerbound`, and `constraint.upperbound`.

The `wcsim` function uses a gradient-based optimization to find a local maxima for the worst-case simulation problem specified in Equation 2. A Simulink model specifies the parameter-dependent nonlinear system and the objective function is specified by the `RCT Objective Function` block. `wcsim` returns the worst-case uncertainties in the structure `wcuvars`. The gradient-based optimization is performed by `fmincon` and thus requires the optimization toolbox. At each iteration of an unconstrained problem `fmincon` evaluates the objective function at the current parameter values as well as at small perturbations along each parameter direction. If the model contains $n_p$ uncertain parameters, `fmincon` will perform $n_p + 1$ simulations at each

iteration. Simulating the system will typically be responsible for the bulk of the computation time to perform a worst-case simulation. Thus the total time for `wcsim` with no constraint blocks will be roughly $(n_p+1)M\tau$ where $\tau$ is the computation time for one simulation and $M$ is the number of iterations. If the model contains constraint blocks then additional function evaluations (simulations) are typically required.

The convergence of `fmincon` depends on the starting value of the parameter vector. `wcsim` initializes the optimization search with the values specified in the `Uncertainty Values` field of the USS blocks. `wcsim` also allows for some parameter values to be held fixed at the values specified in the `Uncertainty Values` field. This is useful to restrict the optimization search space when a Simulink model contains many uncertain real parameters. `wcsim` currently only optimizes over uncertain real parameters; uncertain complex and uncertain linear time invariant objects that exist in the Simulink model are held fixed at the values specified in the `Uncertainty Values` field. The help documentation for `wcsim` (Appendix A) provides additional syntax information. Information on simulation and optimization settings can be found in the documentation for `sim` and `fmincon`.

# 3 GTM Worst-Case Simulation

In this section we apply worst-case simulation analysis to the NASA GTM Simulink model. We briefly describe the real and dynamic uncertainty introduced into the NASA GTM Simulink model. Then we provide worst-case simulation results. Further details on the baseline GTM model (without the uncertainty models) are provided in the GTM documentation.

## 3.1 Parametric Aerodynamic Coefficient Uncertainty

The forces and moments in the GTM model are calculated using six body-axis aerodynamic coefficients: $C_u$, $C_v$, $C_w$, $C_p$, $C_q$, and $C_r$. Each of these six coefficients is a sum of three terms: 1) basic airframe aerodynamic coefficients computed as functions of angle of attack and sideslip, 2) increments to account for the effect of the control surfaces, and 3) increments for dynamic derivatives. These aerodynamic coefficients are not precisely known and it is useful to study the effects of aerodynamic uncertainty on the system performance. We use a multiplicative uncertainty model for each aerodynamic coefficient term. For example, we model 20% uncertainty in the basic airframe coefficients as $C = (I_6 + \Delta)C_{nom}$ where $C_{nom} \in \mathbb{R}^6$ is a vector of the nominal airframe coefficients computed with look-up tables. $\Delta := diag(p_1, \ldots, p_6) \in \mathbb{R}^{6\times 6}$ satisfies $|p_i| \leq 0.2$ for $i = 1, \ldots, 6$. This $\Delta$ was created in Matlab using the `ureal` command:

```
% Body aero coefficients, increments from nominal (+/- 20%)
gainCu = ureal('gain_Cu',0,'PlusMinus',0.2);
gainCv = ureal('gain_Cv',0,'PlusMinus',0.2);
gainCw = ureal('gain_Cw',0,'PlusMinus',0.2);
gainCp = ureal('gain_Cp',0,'PlusMinus',0.2);
gainCq = ureal('gain_Cq',0,'PlusMinus',0.2);
gainCr = ureal('gain_Cr',0,'PlusMinus',0.2);
bai_usys_C6 = diag([gainCu, gainCv, gainCw, gainCp, gainCq, gainCr]);
```

The commands to create the uncertainties for the other two terms of the aero coefficients are given in Appendix B.

These parametric uncertainties were introduced into the Simulink model using Robust Control Toolbox Uncertain State Space (USS) blocks. Figure 1 shows the USS blocks introduced to model the parametric uncertainties in each of the three coefficient terms (magenta USS blocks). Each of the three USS block contains uncertainties for the six aero coefficients for a total of eighteen parametric uncertainties ($n_p = 18$). Figure 2 shows the dialog box for the basic airframe USS block. Note that it contains the `bai_usys_C6` uncertainty which accounts for the uncertainty to the basic airframe coefficients. The "Uncertainty value" is set to nominal which corresponds to $\Delta = diag(0, 0, 0, 0, 0, 0)$.

## 3.2 Dynamic Actuator Uncertainty

The GTM model was also modified to include actuator uncertainty. The GTM model has a total of nine control inputs: left/right ailerons, left/right spoilers, elevator, rudder, flaps, stab, and landing gears. The landing gears are not used for flight control and are not considered in this analysis. The actuators for the

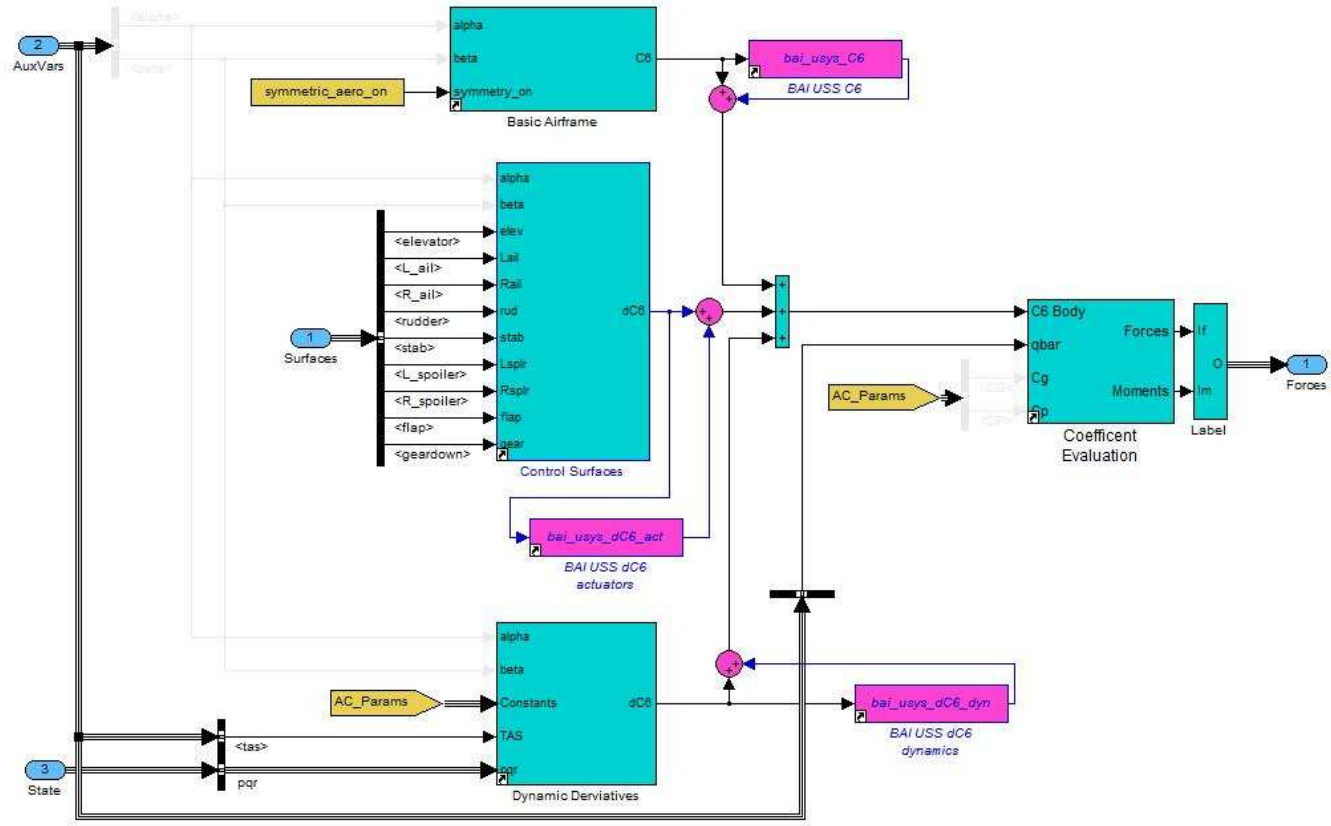


Figure 1: Aerodynamic Coefficient Uncertainty Model

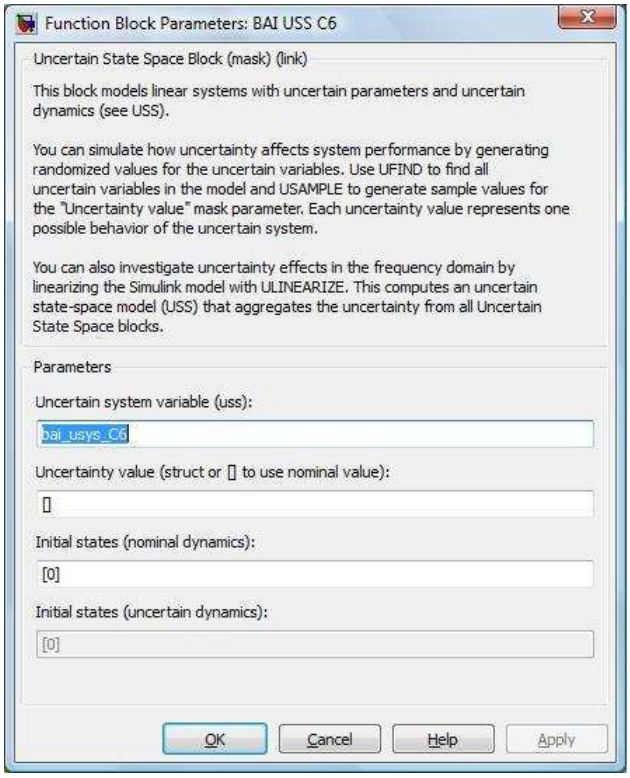


Figure 2: Aero USS Dialog Box

remaining eight control inputs are modeled by first order transfer functions with rate and position limits. This dynamic actuator model is rather simple and it would be useful to study the effects of uncertainties in these dynamics. We use a multiplicative uncertainty model for each actuator. For example, the uncertainty model for the rudder actuator is:

$$\text{Rudder Position} = A_{rud}(s)(1 + W_{rud}(s)\Delta(s)) \cdot \text{Rudder Command}$$

where $A_{rud}(s) = \frac{\omega}{s+\omega}$ is the nominal first-order actuator behavior, $\Delta(s)$ is a norm-bounded uncertainty ($\|\Delta\|_\infty \leq 1$), and $W_{rud}(s)$ is a weighting function which models the magnitude of the rudder actuator uncertainty at each frequency. The $(1 + W_{rud}(s)\Delta)$ term was created using the `ultidyn` command:

```
W_rudder = 1;
bai_usys_rudder = 1+W_rudder*ultidyn('gain_rudder',[1 1]);
```

`W_rudder = 1` is a simplification and more complicated uncertainty models can be created. Our focus will be on the parametric uncertainties and so detailed actuator uncertainty modeling will not be explored. The commands to create all actuator uncertainties (assuming W=1) are given in Appendix B.

These multiplicative uncertainties were introduced into the Simulink model using the Robust Control Toolbox Uncertain State Space (USS) block. Figure 3 shows the USS blocks introduced to model the eight actuator uncertainties. Figure 4 shows the dialog box for the rudder USS block. Note that it contains the `bai_uss_rudder` uncertainty specified above. The "Uncertainty value" is set to nominal which corresponds to `bai_uss_rudder` = 1.

### 3.3 Worst-Case Simulation Results

For each attitude signal we performed a worst-case simulation with all combinations of maximizing and minimizing the $L_\infty$ and $L_2$ norms. We set the optimization options to restrict `fmincon` to three iterations:

```
optimopt = optimset('display','iter','MaxIter',3);
```

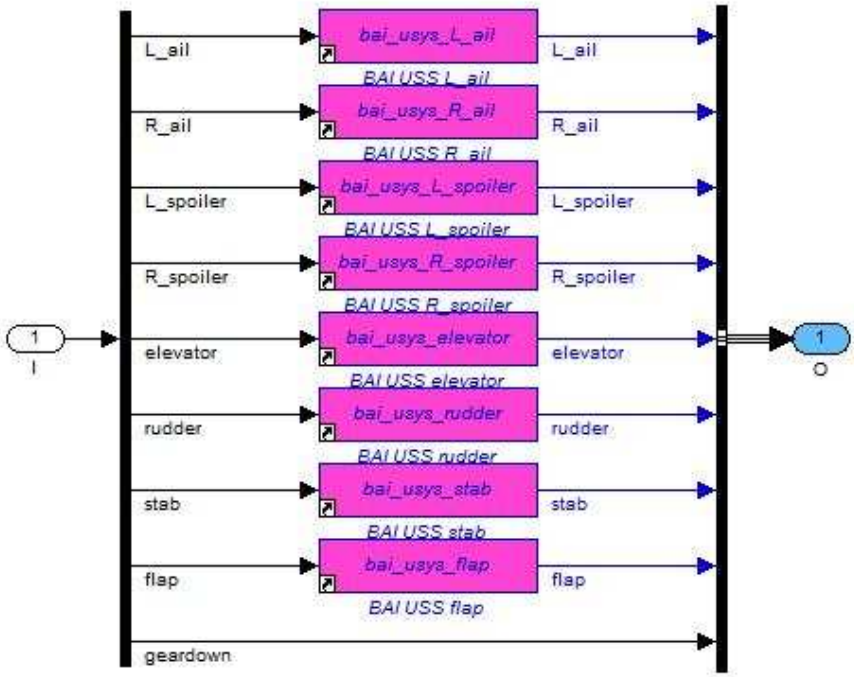


Figure 3: Actuator Uncertainty Model

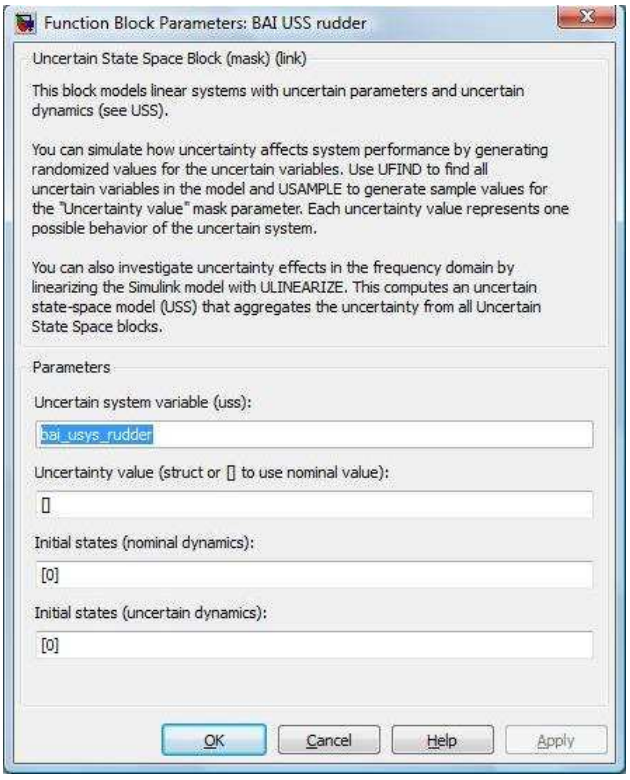


Figure 4: USS Rudder Dialog Box

```
ufixed = [];
wcuvars = wcsim('gtm_design',ufixed,optimopt)
```

All parameters were initialized at their nominal values and all dynamic actuator uncertainties were held fixed at their nominal values. In all simulations, the baseline GTM controller was used with the pilot inputs shown in Figure 7.

The results for all possible combinations of the objective function, max/min, and attitude signal are summarized in Table 1. The first column of the table gives the objective function for the worst-case simulation. The second column provides the total computation time for the worst-case simulation. Columns 3-6 provide the cost function evaluated at each iteration of the optimization. The cost function in column 3 (Step 0) is the cost evaluated at the nominal parameter values. The cost function in column 6 (Step 3) is the final worst-case cost found with by `fmincon`. For each case, the units for the objective function are in radians. Note that in each case `fmincon` appears to converge close to a local optima within the first few steps. The optimization for $\max L_\infty(\psi)$ terminated after Step 1 since the solver determined that no improvement could be made in the objective function.

It takes roughly $\tau = 11.72$sec to perform one simulation of the GTM Simulink model. Thus the total time to run `wcsim` is expected to be roughly 891 sec $\approx$ 15 min [1]. This rough estimate is in fair agreement with the actual computation times provided in Table 1.

Figure 8 shows the aircraft attitude and rates for the nominal system and for the uncertain parameters found by `wcsim` for the $\max L_\infty(\phi)$ objective function. The equivalent airspeed and angle of attack are also shown. The uncertain parameters returned by `wcsim` were:

```
wcuvars =
           gain_Cp: -0.2000
           gain_Cq: -0.2000
           gain_Cr: 0.2000
           gain_Cu: 0.2000
           gain_Cv: 0.0106
           gain_Cw: -0.1988
        gain_L_ail: 0
    gain_L_spoiler: 0
        gain_R_ail: 0
    gain_R_spoiler: 0
      gain_dCp_act: 0.2000
      gain_dCp_dyn: -0.4000
      gain_dCq_act: 0.2000
      gain_dCq_dyn: -0.0053
      gain_dCr_act: -0.2000
      gain_dCr_dyn: -0.4000
      gain_dCu_act: -0.0072
      gain_dCu_dyn: -0.3997
      gain_dCv_act: -0.0030
      gain_dCv_dyn: 4.8835e-004
      gain_dCw_act: -0.2000
      gain_dCw_dyn: 0.0406
     gain_elevator: 0
         gain_flap: 0
       gain_rudder: 0
         gain_stab: 0
```

The aerodynamic coefficients are time-varying due to their dependence on angle of attack, sideslip, attitude rates, and control surfaces. The uncertain parameters listed in `wcuvars` indicate that some aero coefficients have minimal impact on the roll, e.g. `gain_Cv:0.0106` indicates $C_v$ has minimal impact. Other coefficients are set to their maximal perturbations and these should be further investigated. Figure 9 shows the aircraft

[1] The model contains no constraint blocks and has $n_p = 18$ real parametric uncertainties. `fmincon` starts at iteration = 0 and so specifying `MaxIter`=3 will results in $M = 4$ total iterations. Thus the time for `wcsim` is expected to be $(M+1)(n_p+1)\tau$=4*19*11.72 sec.

| Cost Func. | Comp. Time | Step 0 | Step 1 | Step 2 | Step 3 |
|---|---|---|---|---|---|
| $\max L_\infty(\phi)$ | 977.4 | 1.11 | 1.56 | 1.98 | 2.00 |
| $\min L_\infty(\phi)$ | 888.0 | 1.11 | 0.26 | 0.25 | 0.24 |
| $\max L_2(\phi)$ | 935.5 | 4.29 | 6.05 | 6.36 | 6.36 |
| $\min L_2(\phi)$ | 867.8 | 4.29 | 0.82 | 0.81 | 0.81 |
| $\max L_\infty(\theta)$ | 905.2 | 0.64 | 0.65 | 0.88 | 0.89 |
| $\min L_\infty(\theta)$ | 893.9 | 0.64 | 0.53 | 0.50 | 0.48 |
| $\max L_2(\theta)$ | 1015.0 | 1.74 | 2.37 | 2.78 | 2.78 |
| $\min L_2(\theta)$ | 907.8 | 1.74 | 1.60 | 1.46 | 1.44 |
| $\max L_\infty(\psi)$ | 1045.6 | 7.87 | 11.13 | 15.08 | 15.10 |
| $\min L_\infty(\psi)$ | 843.8 | 7.87 | 2.54 | X | X |
| $\max L_2(\psi)$ | 1039.2 | 26.79 | 50.81 | 51.02 | 51.06 |
| $\min L_2(\psi)$ | 1014.1 | 26.79 | 11.90 | 11.88 | 11.84 |

Table 1: Comparison of worst-case attitude and computation times

attitude, attitude rates, equivalent airspeed, and angle of attack for the nominal system and for the uncertain parameters found by `wcsim` for the $\min L_\infty(\phi)$ objective function. Figures 8 and 9 show the range of roll tracking behaviors that can be observed over the specified range of aerodynamic coefficients. It is interesting that the angle of attack is not very sensitive to the perturbations generated by this objective function.

Finally, we demonstrate a worst-case simulation with signal constraints. One RCT `Constraint Function` block was added to the GTM Simulink model to specify a constraint on equivalent airspeed (EAS). The input to this block is the equivalent airspeed and the dialog box for this block is shown in Figure 10. In this case the dialog box specifies that the peak magnitude of the equivalent airspeed must remain less than 175 knots. The objective function is set to $\max L_\infty(\phi)$, i.e. `wcsim` maximizes the $L_\infty$ norm of $\phi$ while constraining the equivalent airspeed to be less than 175 knots. We set the optimization options to restrict `fmincon` to five iterations. Five iterations were needed for convergence in this example. The results for this worst-case simulation are shown in Figure 11. This figure shows the nominal results, worst-case simulation with out constraints, and worst-case simulation with equivalent airspeed constraint. The green line in the EAS subplot shows the 175 knot constraint. `wcsim` returns parameter values which cause the EAS to just touch this constraint. After five iterations, the objective function is $\|\phi\|_\infty = 1.66$. For comparison, the unconstrained worst-case simulation achieved an objective function of 2.00 (first row of Table 1). The objective function value after each iteration of the constrained `wcsim` was: 0) -1.11, 1) -1.11, 2) -1.11, 3) -1.53, 4) -1.70 [Constraint not satisfied], and 5) -1.66. As mentioned previously, constrained problems can take more than $n_p + 1$ function evaluations (model simulations) per iteration. For this problem, iterations 1 and 2 required 33 function evaluations and all other iterations required only 19 function evaluations. The total computation time was 2169.2 sec.

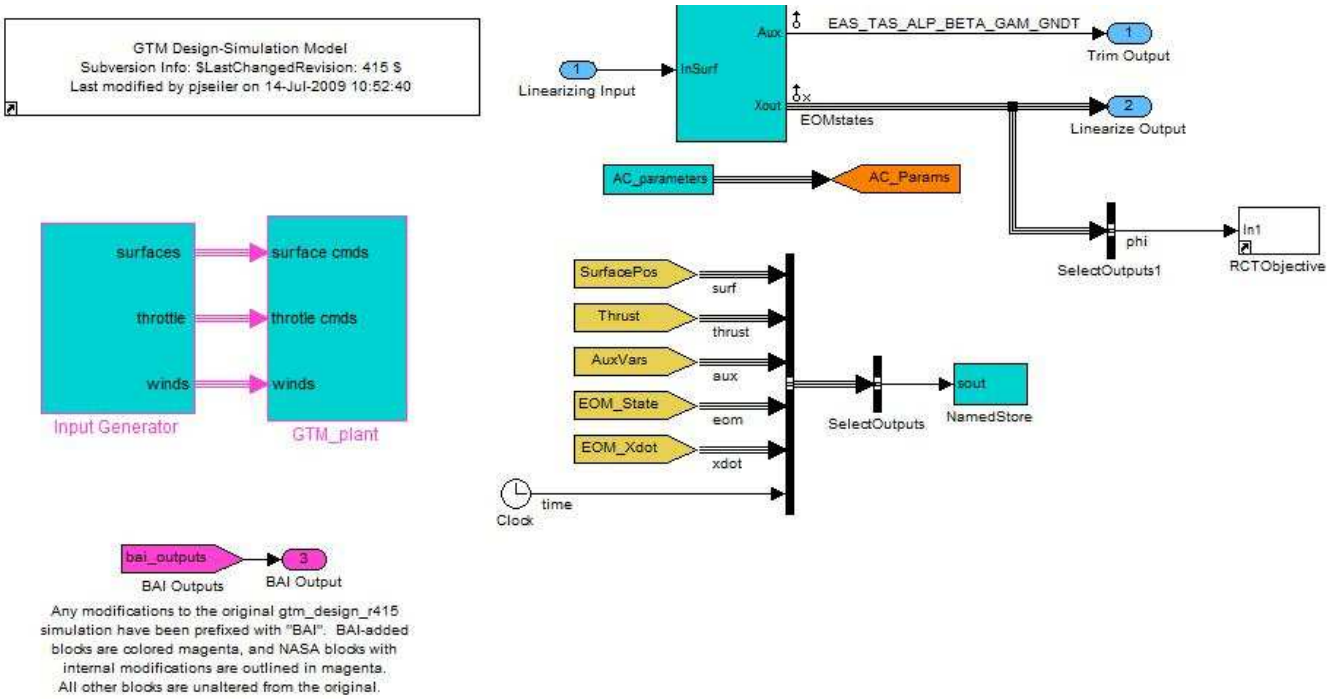


Figure 5: Worst-Case Simulation Objective Function

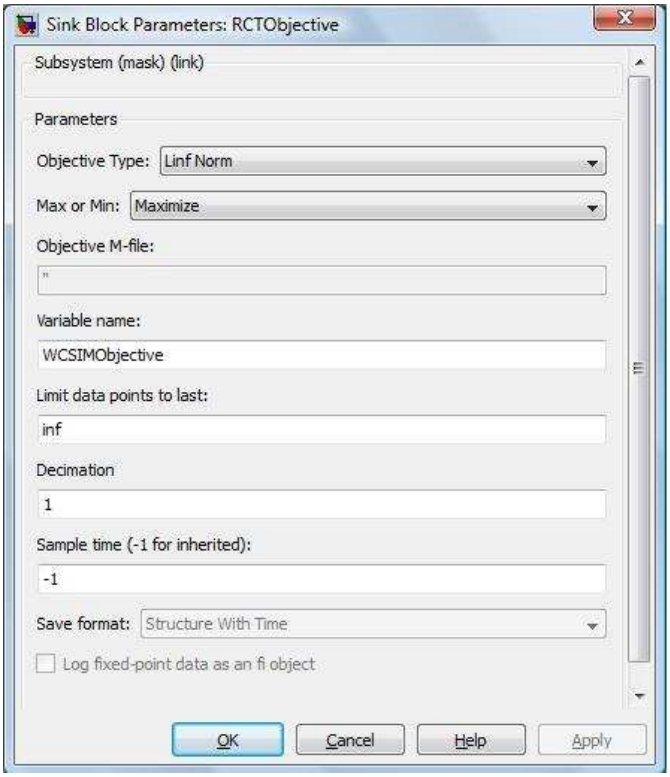


Figure 6: Objective Function Dialog Box

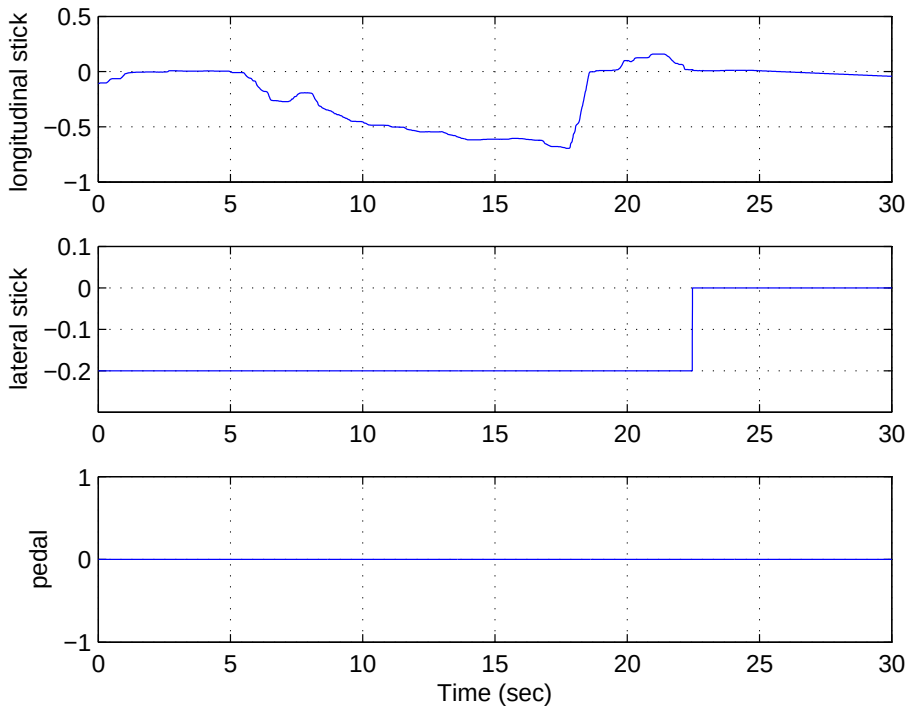


Figure 7: Pilot Inputs for Worst-Case Simulations

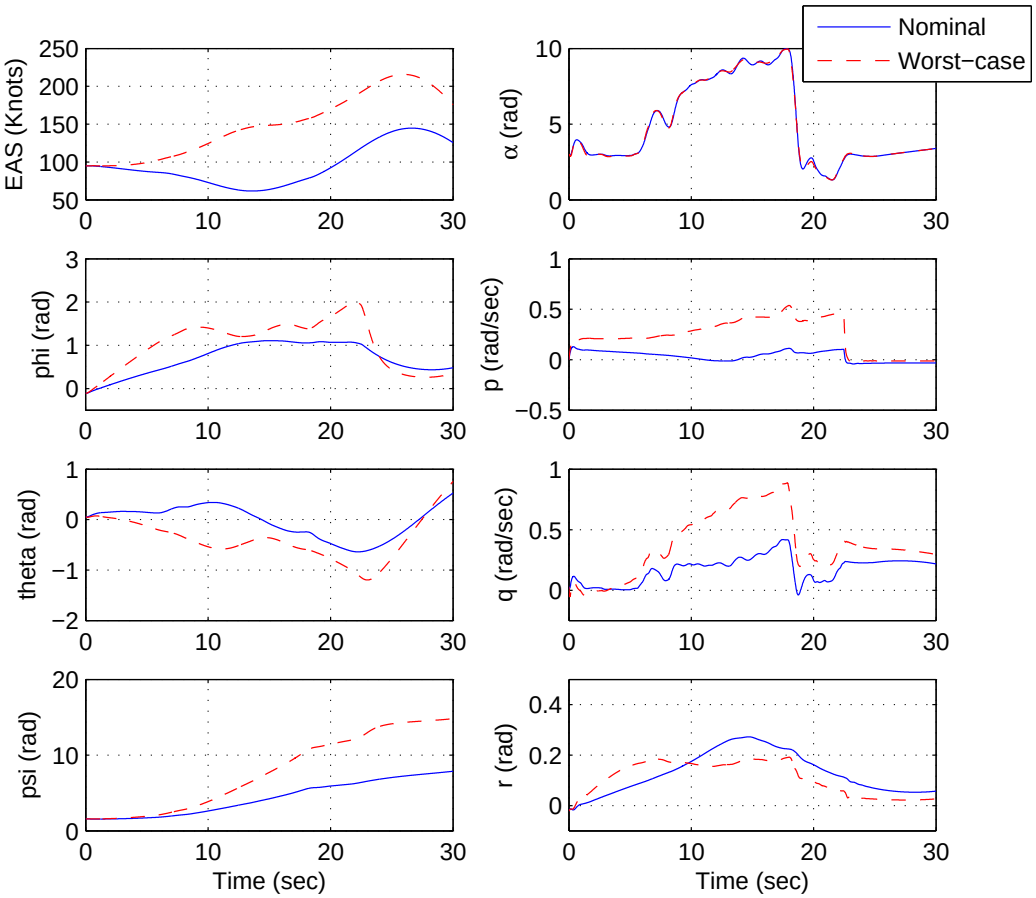


Figure 8: Nominal and wcsim Roll for max $L_\infty(\phi)$

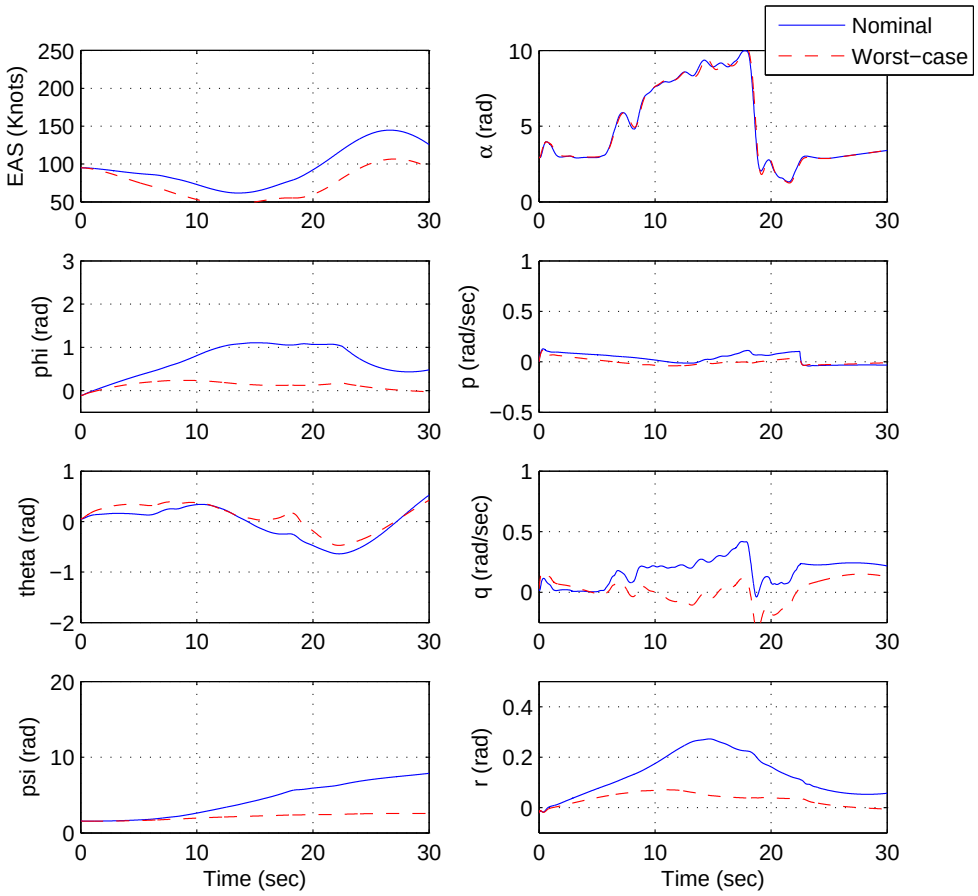


Figure 9: Nominal and wcsim Roll for $\min L_\infty(\phi)$

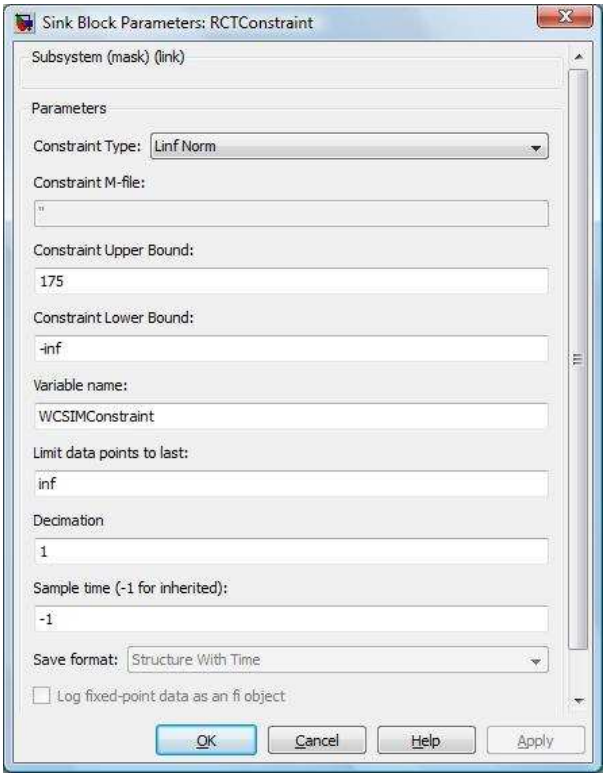


Figure 10: Constraint Function Dialog Box

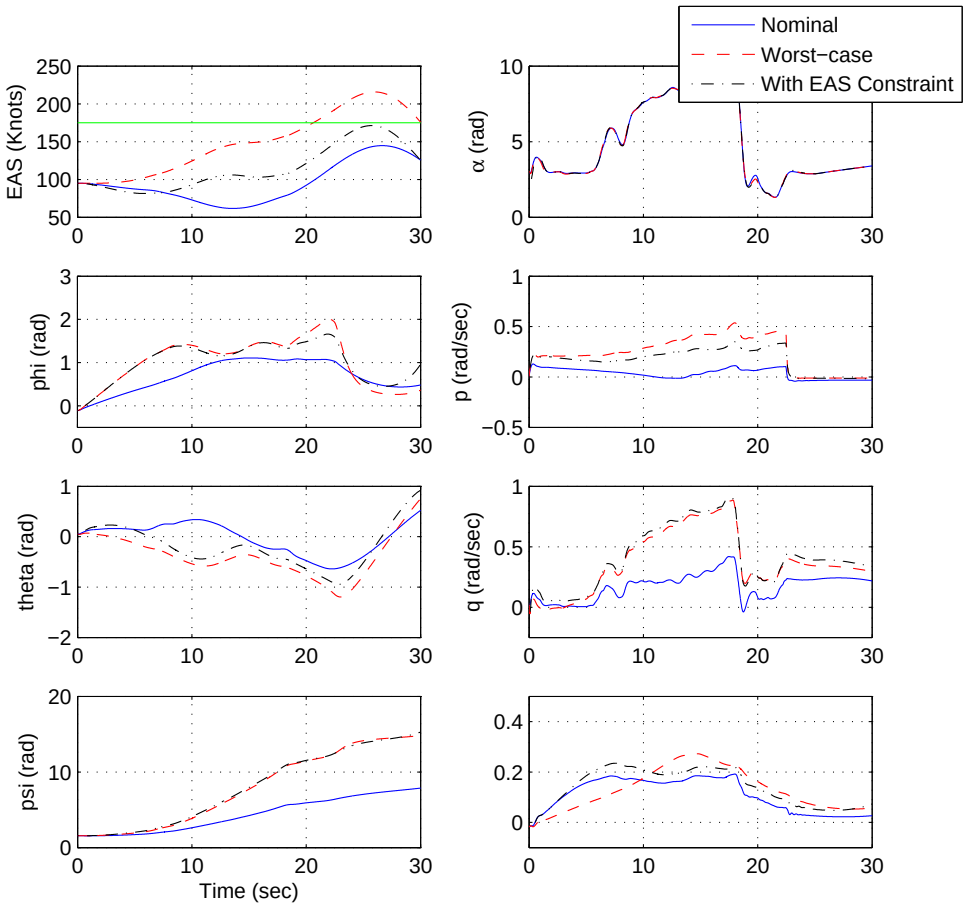


Figure 11: Nominal, wcsim, wcsim with EAS Constraint (wcsim Objective max $L_\infty(\phi)$). Constraint on EAS shown in green.

# A `wcsim` Documentation

```
WCSIM finds worst case uncertainty values in a Simulink model
   WCUVARS = WCSIM('mdl') optimizes the USS blocks in the Simulink model
   'mdl' and returns the optimized values in the structure WCUVARS.  The
   field names and values of WCUVARS are the names of the uncertain
   variables and the optimized values. The optimization objective and
   constraint functions are specified with RCTObjective and RCTConstraint
   blocks. The optimization is performed with FMINCON and requires
   repeated simulations of the model. Only uncertain UREAL parameters are
   varied in the optimization. They are initialized to the values
   specified in the Uncertainty Values field of the USS block. All other
   uncertain variable types are held fixed during the optimization at the
   values specified in the Uncertainty Values field.

   WCUVARS = WCSIM('mdl',UFIXED) specifies additional parameters to hold
   fixed during the optimization.  UVARS is a cell array of strings that
   specifies uncertain UREAL parameters to be held fixed. Set UFIXED=[]
   to optimize over all UREAL parameters.

   WCUVARS = WCSIM('mdl',UFIXED,OPTIMOPT) specifies optimization options
   to be passed to FMINCON.  See FMINCON help for details.

   WCUVARS = WCSIM('mdl',UFIXED,OPTIMOPT,SIMOPT) specifies simulation
   options to be passed to SIM.  All trailing input arguments of wcsim
   are passed directly to SIM.  Set SIM help for details.  WCSIM
   currently forces the 'SrcWorkspace' and 'DstWorkspace' options to be
   set to 'base'.

 See also sim, fmincon, optimset, optimget, ufind
```

# B Uncertainty Object Creation

```
% Body aero coefficients, increments from nominal (+/- 20%)
gainCu = ureal('gain_Cu',0,'PlusMinus',0.2);
gainCv = ureal('gain_Cv',0,'PlusMinus',0.2);
gainCw = ureal('gain_Cw',0,'PlusMinus',0.2);
gainCp = ureal('gain_Cp',0,'PlusMinus',0.2);
gainCq = ureal('gain_Cq',0,'PlusMinus',0.2);
gainCr = ureal('gain_Cr',0,'PlusMinus',0.2);
bai_usys_C6 = diag([gainCu, gainCv, gainCw, gainCp, gainCq, gainCr]);

% Increments to aero coefficients, due to actuators
gainCu = ureal('gain_dCu_act',0,'PlusMinus',0.2);
gainCv = ureal('gain_dCv_act',0,'PlusMinus',0.2);
gainCw = ureal('gain_dCw_act',0,'PlusMinus',0.2);
gainCp = ureal('gain_dCp_act',0,'PlusMinus',0.2);
gainCq = ureal('gain_dCq_act',0,'PlusMinus',0.2);
gainCr = ureal('gain_dCr_act',0,'PlusMinus',0.2);
bai_usys_dC6_act = diag([gainCu, gainCv, gainCw, gainCp, gainCq, gainCr]);

% Increments to aero coefficients, due to dynamic derivs
gainCu = ureal('gain_dCu_dyn',0,'PlusMinus',0.4);
gainCv = ureal('gain_dCv_dyn',0,'PlusMinus',0.4);
gainCw = ureal('gain_dCw_dyn',0,'PlusMinus',0.4);
gainCp = ureal('gain_dCp_dyn',0,'PlusMinus',0.4);
gainCq = ureal('gain_dCq_dyn',0,'PlusMinus',0.4);
gainCr = ureal('gain_dCr_dyn',0,'PlusMinus',0.4);
bai_usys_dC6_dyn = diag([gainCu, gainCv, gainCw, gainCp, gainCq, gainCr]);

% Actuator Dynamic Uncertainties (Weight=1 for all uncertainties)
bai_usys_L_ail = ultidyn('gain_L_ail',[1 1]) + 1;
bai_usys_R_ail = ultidyn('gain_R_ail',[1 1]) + 1;
bai_usys_L_spoiler = ultidyn('gain_L_spoiler',[1 1]) + 1;
bai_usys_R_spoiler = ultidyn('gain_R_spoiler',[1 1]) + 1;
bai_usys_elevator = ultidyn('gain_elevator',[1 1]) + 1;
bai_usys_rudder = ultidyn('gain_rudder',[1 1]) + 1;
bai_usys_stab = ultidyn('gain_stab',[1 1]) + 1;
bai_usys_flap = ultidyn('gain_flap',[1 1]) + 1;
```