110059

N95-16474

Matlab as a Robust Control Design Tool

Irene M. Gregory
Dynamics and Controls Branch

This presentation is geared towards introducing Matlab as a tool used in flight control research. The example problem used to illustrate some of the capabilities of this software is a robust controller designed for a Single-Stage-To-Orbit airbreathing vehicle's ascent to orbit. The details of the problem and the control law design are available from reference 1. The global requirements on the controller are to stabilize the vehicle and follow a trajectory in the presence of atmospheric disturbances and strong dynamic coupling between airframe and propulsion. Hence, the need for a robust controller.

Matlab is an interactive program designed for numerical computation, data analysis and visualization as well as a philosophy of open architecture. Fundamentally, Matlab is built upon a foundation of sophisticated matrix software for analyzing linear systems of equations. The relevance of this is that matrices are useful because they can describe so many things in a mathematically efficient and highly flexible way. Matlab serves as a kernel from which several toolboxes are linked. Application toolboxes as the name implies are a collection of predefined functions intended to solve more application-specific problems such as a control design problem that requires system modeling, controller synthesis and analysis. One of the most important tools for modeling complex nonlinear systems and simulating them is Simulink. An example of such a system is presented here. The model consists of an integrated aerodynamics/propulsion database, various information for use with a pilot on approach and landing, and a full 6 d.o.f. rotating earth equations of motion along with an atmospheric model. Not only does Simulink provide a straight forward way to easily build this system, but it also incorporates files written in different languages, in this case FORTRAN and C, in the model without any modifications.

Given the nonlinear system we proceed with trimming the vehicle and deriving linear models. Both of these are predefined functions that can be executed in a single line. Since the system is unstable, the controller is required to both stabilize the vehicle and follow the prescribed trajectory. Typically synthesizing a controller is an iterative procedure and it becomes advantageous to automate the process. An m-file consisting of Matlab commands can be used to define scaling for optimized variables, construct the new linear system that includes these scaling, and perform controller synthesis and analysis. This system model would also include uncertainty that may arise from various physical considerations. Once written, the iterative process can be completed in a few key strokes per iteration. These m-files serve as an example of yet another language that can be utilized in Matlab.

This controller example utilizes some modern control techniques that are available from μ-tools and to some extent from Robust control toolbox. The controller synthesis problem is solved using H∞ optimization and analysis are performed using μ, also known as structured singular value. μ is analogous to Bode plots in the classical control methodologies. The μ plot allows an immediate assessment of whether a controller has fulfilled the specified requirements. Once the desired controller has been found, a model reduction, to reduce its dynamic order, is performed using a number of techniques among them Hankel singular values and residualized truncation. A reduced order controller is then integrated into the nonlinear simulation. Time response of the system is evaluated as both a confirmation of frequency domain μ analysis and another way of evaluating results.

Matlab/Simulink combination also has the capability of automatically generating C code for any block in a diagram. This capability is very useful for transferring controller from the design environment into non-Matlab environments such as real time simulation or even flight test. These capabilities are being currently evaluated.

In summary, a number of different capabilities of Matlab were illustrated in this example. We find Matlab a powerful yet very flexible tool to use in controls research.

References:

Gregory, Irene M.; McMinn, John D.; Chowdhry, Rajiv S. and Shaughnessy, John D.: Hypersonic Vehicle Model and Control Law Development Using H∞ and μ-Synthesis. NASA TM-4562, July 1994.

Matlab as a Robust Control Design Tool

Irene M. Gregory
Dynamics and Controls Branch

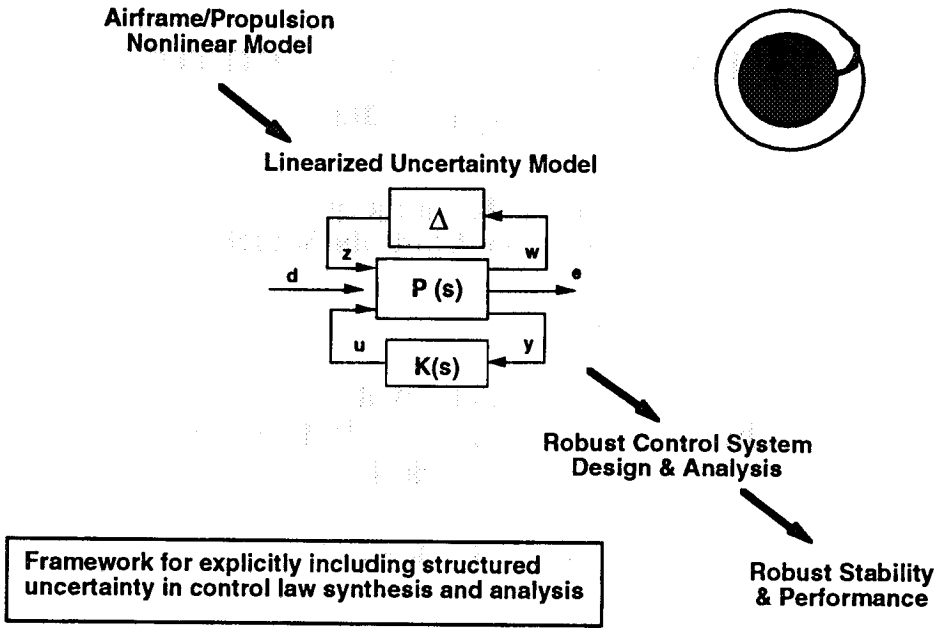
presented at
The Role of Computers In LaRC R&D
1994 Workshop

June 15 - June 16

Presentation Outline

- Robust control law problem
- Introduction to Matlab
- Nonlinear system simulation
- Linear model derivation
- Sample command file
- Controller synthesis and analysis
- Concluding remarks

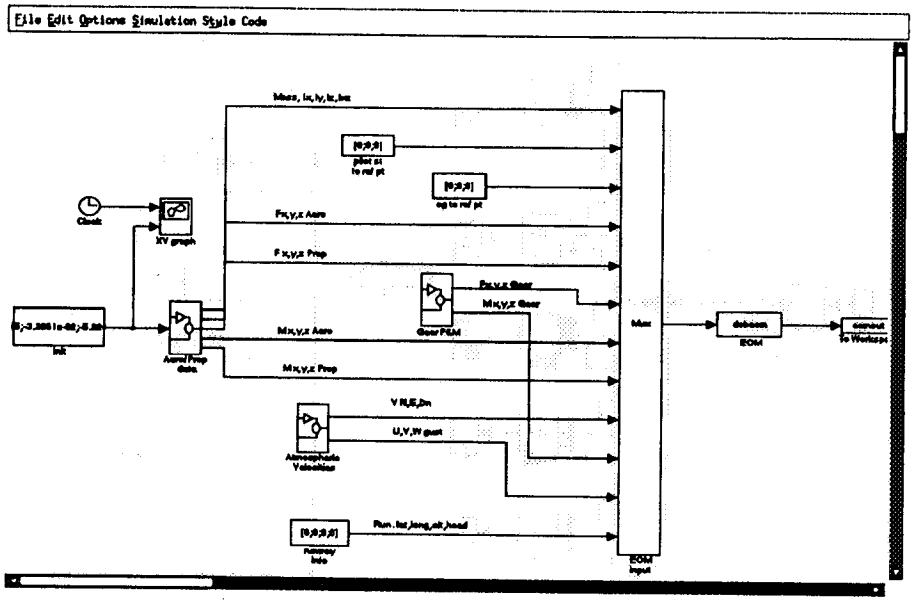
Robust Control Law Framework



Introduction to Matlab

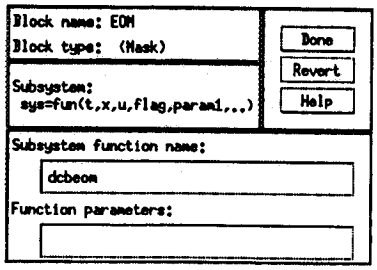
- **Matlab**
  - For numeric computation, visualization, and data analysis
  - The basis of Matlab is matrix manipulation and matrix solving.
  - Matlab is a kernel from which several toolboxes, a collection of predefined functions, are linked
- **Simulink**
  - For advanced nonlinear modeling and simulation
- **Application Toolboxes**
  - For customizing your Matlab environment with special tools to solve more application-specific problems.
    - » e.g. Controls, μ-Tools, Signal Processing, Neural Nets

Matlab Nonlinear Simulation



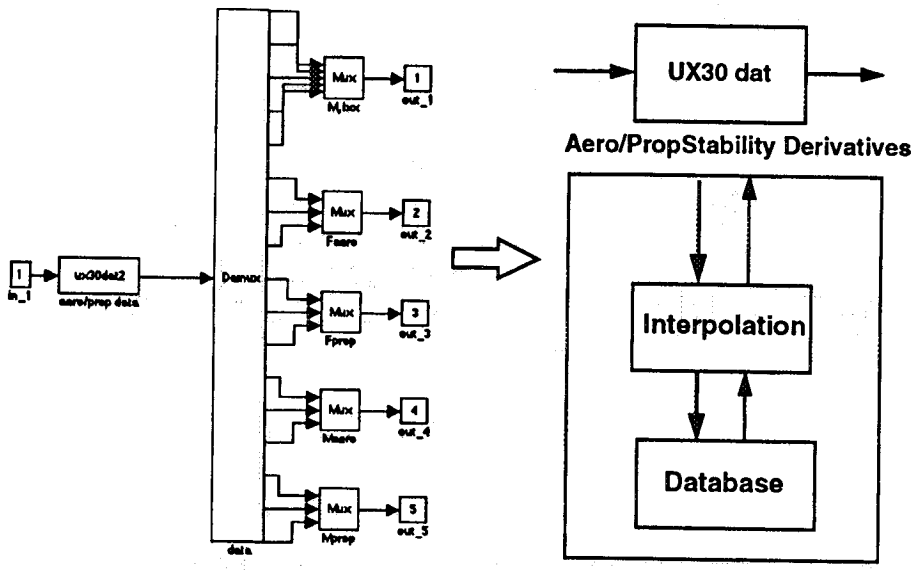
Eqn's of Motion Block

EOM

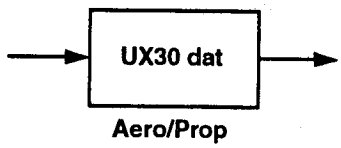


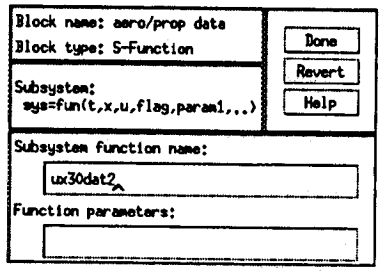

```
* mdlInitializeSizes - initialize the sizes arr
* The sizes array is used by SIMULINK to
    determ
 * characteristics (number of inputs, outputs, s
 */
#define EOMDEBUG
#define NSTATES    7
#define NOUTPUTS   115
#define NINPUTS    39
static void mdlInitializeSizes(S)
   SimStruct *S;
{
   ssSetNumContStates(   S, NSTATES); /*
    numb

   ssSetNumDiscStates(   S, 0);       /* numb
   ssSetNumInputs(       S, NINPUTS); /* numb
   ssSetNumOutputs(      S, NOUTPUTS); /*
    numb
   ssSetDirectFeedThrough(S, 1);      /* dire
   ssSetNumSampleTimes(  S, 1);       /* numb
```

Aero/Propulsion Model



Aero/Prop Model Block






```
c     perform table look-ups
c
      call ux30d(mach,weight,alpha,dela,dele,
     1delf1,delf2,eta,specv25,altito,vela,
                        •
                        •
                        •
c     Mass of vehicle
c
      mass = weight/32.17
c
c     sum aero forces and moments
c
      clift  = clt + cldat + cldet
      cdrag  = cdt + cddat + cddet + cdf1t + cdf2t
c
      sinalf = sin(alpha*d2r)
      cosalf = cos(alpha*d2r)
c
      cx     = -cdrag*cosalf + clift*sinalf
      cy     = cybt*beta + cydat + cydet + cyf1t + cyf2t
      cz     = -clift*cosalf - cdrag*sinalf

      cll    = clbt*beta + cldat + cldet + clf1t + cll
     1 (cllpt * pbody + cllrt * rbody) * bspan/(2.0*vrw)
      cm     = cmat + cmdat + cmdet  + cmf1t + cmf2t +
     1  cmqt*qbody * cbar/(2.0*vrw)
      cn     = cwbt*beta + cwdat + cwdet + cwf1t + cwf2t +
     1 (cwpt*pbody + cwrt * rbody) * bspan/(2.0*vrw)
c
c     final outputs from user code block
c
      lift   = dynp*sarea*clift
      drag   = dynp*sarea*cdrag
      lovd   = lift/drag
c
      xaero  = dynp*sarea*cx
      yaero  = dynp*sarea*cy
```

Linear Model Derivation

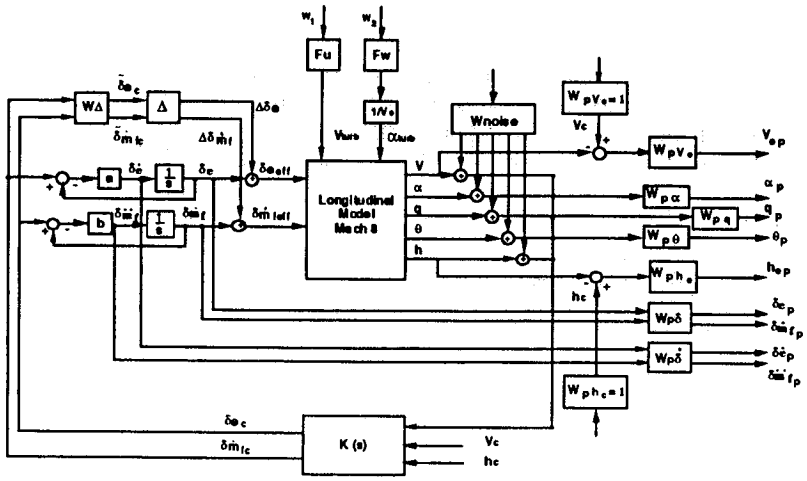
```
>> [ad,bd,cd,dd] = linmod('ux30');
>> ad
ad =
 -2.2037e-02  6.9900e-03         0      -5.6189e-01  -8.4104e-03
 -3.7593e-05 -9.7957e-02  1.0000e+00  -4.2937e-05   2.8210e-04
 -3.7335e-02  3.8823e+00 -1.2216e-01         0      -3.8333e-04
  2.7302e-06  3.9183e-06  1.0000e+00  -3.9183e-06   1.6346e-14
  1.0472e-02 -1.3703e+02         0       1.3703e+02         0

>> bd
bd =
  4.5324e-02  4.1653e-01
 -9.4348e-03 -2.1950e-03
 -2.4606e+00 -2.4238e-03
        0           0
        0           0
```

Linear Model Evaluation

```
>> eig(ad)

 -2.0772e+00
  1.8636e+00
 -5.4707e-02
  1.3076e-02 + 1.9857e-01i
  1.3076e-02 - 1.9857e-01i
```

- Unstable system=> controller requirements
  - Controller stabilize vehicle
  - Controller follows prescribed path

System Block Diagram



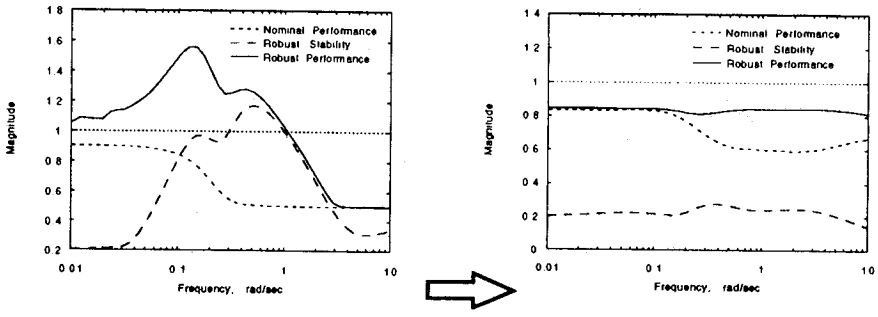
Controller Synthesis

```
>> Olplant_UX1
system:  12 states    16 outputs    11 inputs
Test bounds:     0.5000 <  gamma  <=    10.0000

 gamma    hamx_eig  xinf_eig  hamy_eig   yinf_eig   nrho_xy
   p/f
 10.000   3.9e-02   3.3e-10   9.6e-04    0.0e+00    0.0000    p
  5.250   3.9e-02   3.3e-10   9.6e-04    0.0e+00    0.0000    p
                      :
                      :
  0.648   3.9e-02  -5.8e-05#  9.6e-04    0.0e+00    0.0003    f
  0.723   3.9e-02   3.9e-10   9.6e-04    0.0e+00    0.0005    p
  0.686   3.9e-02  -2.2e-02#  9.6e-04    0.0e+00    0.0419    f
  0.704   3.9e-02   4.0e-10   9.6e-04    0.0e+00    0.0009    p
  0.695   3.9e-02   4.0e-10   9.6e-04    0.0e+00    0.0017    p

Gamma value achieved:    0.6948
```

M-file Example

```
% Performance Weighting Functions on Actuators
Wpe = daug(20,5);          % weighting on dele, deleta
Wpact  = daug(1,1);                % weighting on elevon rate
        :
systemnames = 'ac Wpv Wph Wpa Wpq Wpo Wpe Wn se0 se1 se5 se6 Wpact
   Fv Fh Wcmdv Wcmdh ';

inputvar = '[xt_inp(11)]';

outputvar='[Wpv;Wph;Wpa;Wpq;Wpo;Wpe;Wpact;ac+ Wn;Wcmdv;Wcmdh]';

input_to_ac='[Fh;Fv;se1;se6]';
input_to_Wn='[xt_inp(1:5)]';
input_to_Fh = '[xt_inp(6)]';
        :
% H ° controller calculation
[k1,clp1]=hinfsyn(acolp1,nmeas,ncon,0.5,10,0.01);
```

Controller Evaluation Tools

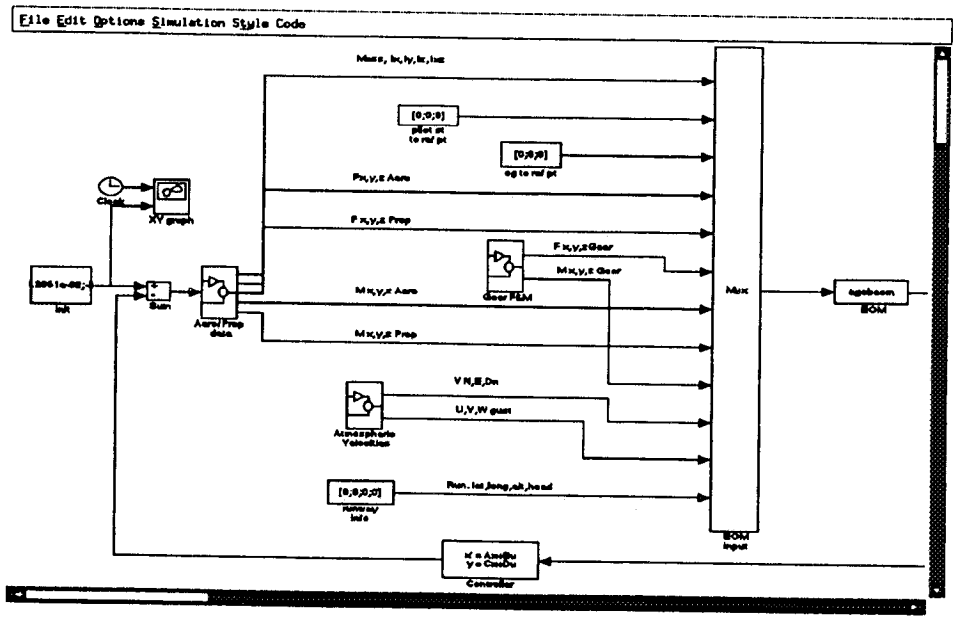


- Frequency domain μ-based performance evaluation plots

Controller Order Reduction

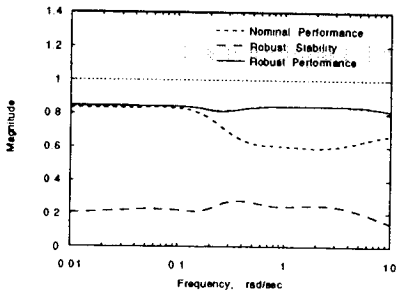
- Nominal controller - 23 states
- Balanced realization and Hankel singular values
  - >> [Kbal, hanksv] = sysbal(Khinf);
  - >> [Kred,Kunst] = hankmr(Kbal,hanksv,13);
- Final reduced-order controller - 13 states

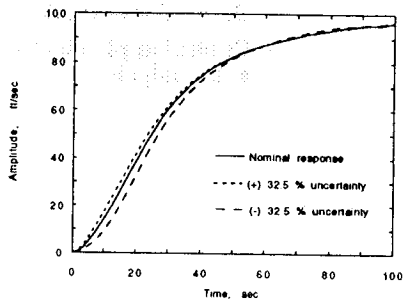
Nonlinear Simulation



Controller Evaluation

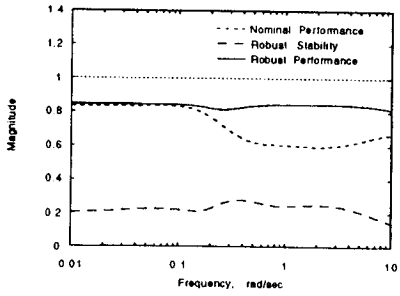
- Reduced order controller evaluation in frequency and time domain
  - μ(closed loop system)
  - nonlinear simulation time response

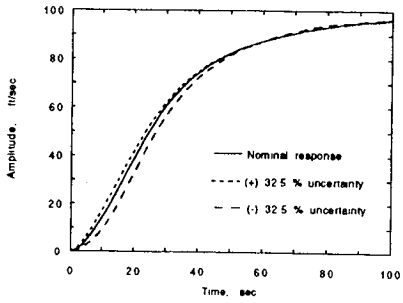




Controller Evaluation

- Reduced order controller evaluation in frequency and time domain





Concluding Remarks

- **Matlab capabilities utilized**
  - Link together FORTRAN, C, and Matlab functions
  - Nonlinear simulation
  - Trim vehicle
  - Derive linear model
  - Control application toolboxes for controller synthesis and analysis