

NASA's X-43A Scramjet Achieves Record-Breaking Mach 10 Speed Using MathWorks Tools for Model-Based Design



The X-43A on its record-setting flight.

THE CHALLENGE

To design and automatically generate flight control software for a scramjet vehicle traveling at Mach 10 speed

THE SOLUTION

Use Simulink to model and validate control systems, Real-Time Workshop to automatically generate flight code, and MATLAB to process and analyze postflight data

THE RESULTS

- Reduced development time by months
- Accurately predicted separation clearance
- Aided in achieving SEI CMM Level 5 process rating

On November 16, 2004, NASA made history by launching the X-43A, the first-ever air-breathing hypersonic vehicle, into the atmosphere, achieving Mach 10 speed. The X-43A separated from its booster and accelerated on scramjet power at nearly ten times the speed of sound (7,000 MPH) at roughly 110,000 feet. The experiment enabled NASA to validate key propulsion and related technologies for air-breathing hypersonic aircraft.

Dubbed Hyper-X, the project was a collaborative effort involving engineers from a variety of organizations, including NASA Dryden Flight Research Center, NASA Langley Research Center, Analytical Mechanics Associates (AMA), and Boeing PhantomWorks. These teams used MathWorks tools for Model-Based Design to develop and automatically generate flight code for the vehicle's propulsion and flight control systems. They also used MATLAB® to analyze preflight assumptions and post-flight results.

THE CHALLENGE

NASA was tasked with developing controls for the X-43A and its subsystems, including flight control, propulsion, actuators, and sensors. These controls would keep the unmanned vehicle stable within a half-degree angle-of-attack and ensure sufficient clearance between the research vehicle and the adaptor on the front of the booster when the two parts separated. The engineers would need to complete the project under a wide range of environmental conditions and an uncharted flight regime.

Because this unique project involved multiple teams and a highly complex design, NASA would need a common modeling environment and a proven design process based on reliable models.

With a high likelihood that system requirements and models would change as the program matured, they also sought to automate development and minimize manual coding and debugging.

Finally, NASA would need tools for efficiently analyzing gigabytes of multidimensional telemetry data.



The X43-A vehicle components, including the controller, actuator, and FMU.

THE SOLUTION

The Guidance, Navigation, and Control team at NASA worked with Boeing and AMA to develop the propulsion and flight control laws for the X-43A scramjet and integrate them into the onboard system. All teams collaborated on the project by applying Model-Based Design with MathWorks tools.

“There aren’t any software packages out there that can match the capabilities of MathWorks tools,” says Dave Bose, vice president of modeling and simulation at AMA. “From the team’s perspective, it really was an easy decision to choose MathWorks tools.”

Designing, Simulating, and Validating the Onboard Control Systems

NASA and AMA used Simulink® to design control law gains and ensure acceptable stability margins. Simulink also helped them move quickly through the simulation stages, which included running Monte Carlo simulations on a host computer and validating the design with preflight hardware-in-the-loop (HIL) tests on real-time computers.

Engineers implemented a linear model of the flight control system in Simulink and used the Control System Toolbox to design loop gains and analyze stability margins.

Working in Simulink, AMA developed sophisticated nonlinear models of the entire vehicle and subsystems, including a six-degree-of-freedom (6 DOF) plant environment, control systems with complex filters, high-fidelity actuator models, and detailed sensor representations. They used MATLAB and Simulink to reconcile those models with actual flight data.

“Building algorithms in Simulink is much easier than in Fortran because you’re building subsystems instead of subroutines, and these are more intuitive to organize,” says Luis Miranda, systems analyst at Boeing. “Also, I can’t think of a more efficient way to deliver software requirements than with a Simulink model.”

Engineers from AMA and NASA used MATLAB and Simulink to model and simulate the separation event to ensure that the booster’s adaptor and the research vehicle did not make contact. They acquired sensor data from ground tests on the separation pistons and used MATLAB to measure and analyze test data and the Optimization Toolbox to match parameters to the data. They then used the data to develop an accurate model in Simulink, which served as the truth model for validating simulations and preflight test results.

Generating and Integrating the C Code into the Flight Management Unit (FMU)

NASA and Boeing used Real-Time Workshop® to automatically generate C code for the X-43A’s propulsion and flight control systems. They used the code to run 6 DOF simulations for non-real-time actuator testing and HIL tests. The automatically generated code also ran inside the Honeywell H-764 FMU that flew on the X-43A.

Engineers used Real-Time Workshop to generate the header, registration, parameter, and main algorithm C files. Because the parameters were easy to access, they inspected them for the flight control and propulsion systems during testing and on the day of flight.

Boeing automated two major testing stages with Simulink and Real-Time Workshop: component testing and HIL testing. They validated the software requirements and performed structural coverage analysis, avoiding inspections of the automatically generated code while achieving a transparent verification and validation process.

“To make system modifications, we updated the Simulink diagram, automatically generated the code, installed the generated code, and hit the build button,” says Paul Seigman, software engineer at Boeing PhantomWorks. “We experienced a significant increase in productivity and avoided the pitfalls of hand coding.”

For component testing, Boeing used Simulink to run tests with their stimulus models on the host processor before components were integrated into the embedded code. Using Real-Time Workshop, they then automatically generated C code and checked for potential differences between the simulation results and the generated code.

For HIL testing, they tested the functionality of the full software application, including the automatically generated code on the FMU. Boeing used an inertial simulator as a flight table that fed velocity data into the FMU, which instructed it to “fly” at various speeds. During the HIL tests, engineers monitored the entire flight trajectory from boost to separation to splash down, collecting telemetry data off the FMU bus controller for postprocessing with MATLAB.

Boeing also used Real-Time Workshop to provide NASA with build updates to meet various milestones of the vehicle integration and testing.

“We wouldn’t have been able to provide NASA with interim builds as efficiently without automatically generated code because the control laws changed frequently,” says Seigman. “We never found any errors in the automatically generated code, so we were confident in creating a quick prototype for NASA.”

Analyzing and Postprocessing Postflight Data

To accurately estimate the vehicle’s trajectory after launch, NASA used MATLAB to build Kalman filters that removed noise from telemetry sources, such as raw inertial measurements, atmospheric conditions, and information from their GPS antennas.

To analyze the separation event, NASA used MATLAB to automate the processing of large eight-dimensional aerodynamic data tables that model the interference effects between the adaptor and research vehicle. They visualized the data with MATLAB plots and graphs.

“The size and complexity of the data posed a tough challenge,” says John Martin, systems analyst at NASA. “I couldn’t have solved that problem without MATLAB.”

NASA is now using MathWorks tools to advance hypersonics technology by researching ways to extend burn time and achieve higher Mach speeds for possible future missions.

“

Our autopilot worked on the first try, which is amazing given that a vehicle like this had never been flown before. MathWorks tools helped us design and implement control systems that kept the vehicle stable throughout the flight.

”

Dave Bose, Analytical Mechanics Associates

THE RESULTS

■ **Reduced development time by months.** “Automatically generating code with Real-Time Workshop saved us several months,” explains Seigman. “I don’t believe we would have met our deadline if we had to write all our code by hand.”

■ **Accurately predicted separation clearance.** “Because our animation software package could not inform us how close the adaptor and the vehicle would come during separation, we couldn’t design the control strategy to maximize clearance,” explains Martin. “We calculated the proximity easily with MATLAB, and our postprocessing results proved our predictions were accurate.”

■ **Aided in achieving Software Engineering Institute Capability Maturity Model (SEI CMM) Level 5 process rating.** “With our rigorous process for developing flight code, we were rated at SEI Level 5 for this project,” explains Seigman. “We used our component and HIL testing of the automatically generated code as one of our process improvements, which satisfied the Level 5 criteria.”

For more information about NASA and the Hyper-X project, visit www.nasa.gov/missions/research/x43-main.html

APPLICATION AREAS

- Aerospace and defense
- Automatic code generation
- Model-Based Design
- Control design
- Simulation
- Data analysis

PRODUCTS USED

- MATLAB
- Simulink
- Real-Time Workshop
- Control System Toolbox
- Optimization Toolbox

www.mathworks.com