

MathWorks News&Notes

The Magazine for the MATLAB® and Simulink® Community



Modeling and Simulating Next-Generation Wave Farm Technology

Perception Systems for
Automated Driving

First-Ever Detection of
Gravitational Waves

Training a Deep
Learning Network

Cleve's Corner:
Extrapolating
Census Data

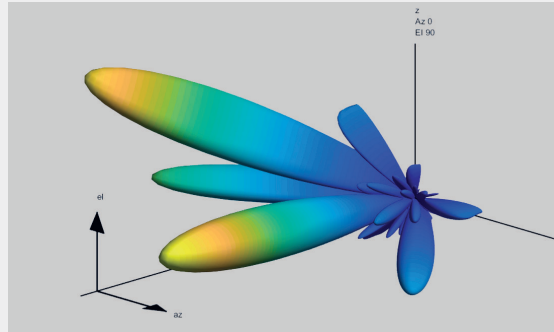
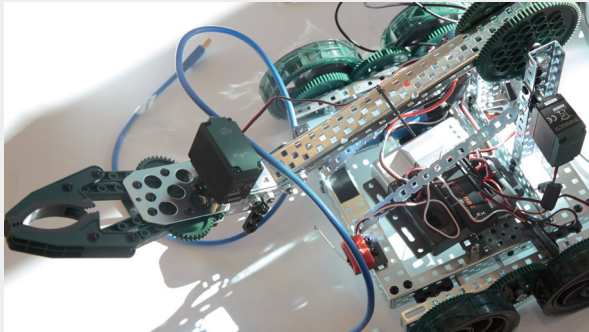
Designing for
the 5G Standard



About the Cover

The cover shows two of the submerged buoys in Carnegie Clean Energy's CETO 5 wave energy system. In a CETO system, pumps actuated by the motion of the buoys pressurize water to drive hydroelectric conversion devices, generating up to 240 kW of zero-emission electricity. Carnegie engineers are now working on CETO 6, which has the potential to generate more than four times the electric power of CETO 5 per buoy.

Read the full story on page 6.



MATLAB® & SIMULINK®

MathWorks is the leading developer of mathematical computing software. Engineers and scientists worldwide rely on its products to accelerate the pace of discovery, innovation, and development.

What will *you* do with MATLAB and Simulink?

MathWorks News & Notes

>> FEATURES


- 6 Modeling and Simulating Next-Generation Wave Farm Technology**
Carnegie Clean Energy engineers are developing the next generation of the company's wave-energy technology using a full-scale wave-to-wire model of the entire multidomain system built with Simulink and Simscape.
- 10 Optimizing Automotive Manufacturing Processes with Discrete-Event Simulation**
Daimler engineers run simulations with Simulink and SimEvents to aid operational decision-making, forecast the outcomes of manufacturing process changes, and improve production-line efficiency.
- 14 Visualize, Label, and Fuse Sensor Data for Automated Driving**
How do you make sense of all the input data you have to work with? How do you reconcile conflicting inputs from different sensors? And once you've developed an algorithm, how do you evaluate its results? This guide to MATLAB and Automated Driving System Toolbox capabilities can help you address these questions.
- 18 Accelerating Drone Research with a Ready-to-Fly Hexacopter and Flight Control Software**
IntelinAir's RD-100 ready-to-fly drone is equipped with customizable autopilot flight software capable of maintaining stability in flight and automating takeoffs, landings, and waypoint navigation.
- 24 Teaching Mechatronics with MATLAB, Simulink, and Arduino Hardware**
Rensselaer Polytechnic Institute mechatronics students put theory into practice with labs based on a take-home kit that includes an Arduino-based microcontroller, a DC motor, and MATLAB and Simulink software.
- 28 Confirming the First-Ever Detection of Gravitational Waves**
"When colleagues in Europe notified me of the extraordinary discovery, I immediately downloaded the laser interferometer data, opened MATLAB, and began analyzing the recorded signals—confirming that they really did come from a gravitational wave."
- 32 Classifying Handwritten Japanese Characters with Deep Learning**
Could a deep learning network be trained to identify characters from ancient Japanese manuscripts? MathWorks consultant Akira Agata thought so. Find out how he did it.
- 36 Robot Prints 3D Building in Half a Day**
Researchers from MIT's Mediated Matter Group designed a robotic platform that can 3D-print architecture, then printed a building. Find out how it was done.
- 37 *grimsel* Sets Electric Vehicle Acceleration Record**
"0–60 miles per hour in 1.513 seconds! Once we'd grasped the significance of that result, the awesome feeling that we'd just broken the world record set in."



©2017 The MathWorks, Inc.

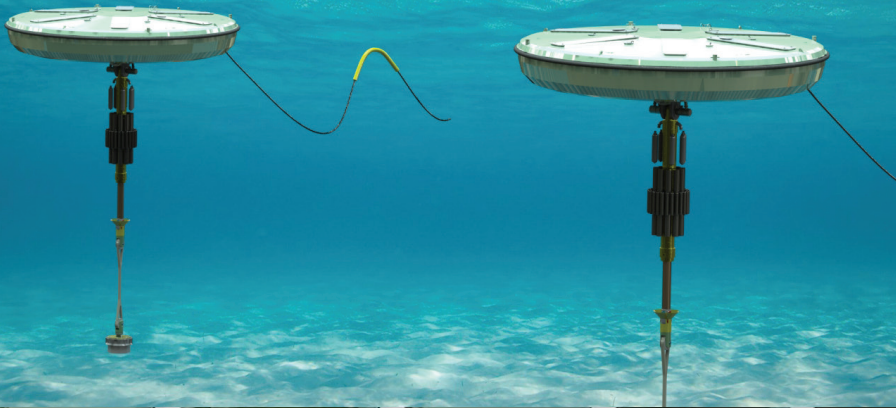
MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.



 Printed on 30% post-consumer waste materials

Made in the U.S.

6



36



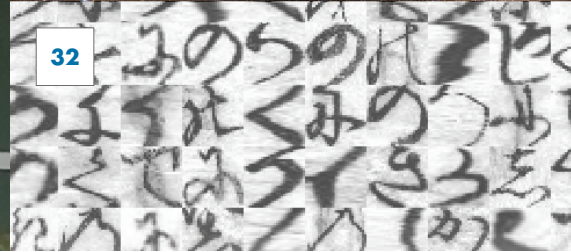
18



28



32



>> DEPARTMENTS

4 **MATLAB and Simulink in the World:**
Data Analytics

22 **Tech Spotlight:** *Modeling the Emerging
5G Standard with the 5G Library*

23 **Third-Party Products:** *Solutions for Developing
Custom Motor Control Applications*

33 **Cleve's Corner:** *Fitting and Extrapolating
U.S. Census Data*

MANAGING EDITOR

Linda Webb

EDITOR

Rosemary Oxenford

ART DIRECTOR

Vic Cevoli

GRAPHIC DESIGNERS

Gabrielle Lydon
Chris Roth

TECHNICAL WRITER

Jack Wilber

PRODUCTION EDITOR

Julie Cornell

PRINTER

DS Graphics

DIGITAL PRODUCTION SPECIALIST

Alex Pollack

PRINT LIAISON

Jill Mespelli

EDITORIAL BOARD

T. Andrzejek, S. Gage, C. Hayhurst,
M. Hirsch, S. Lehman, D. Lluch,
M. Maher, A. May, C. Moler,
M. Mulligan, L. Shure, J. Tung

CONTRIBUTORS AND REVIEWERS

A. Agata, L. Andresen, M. Apfelbeck,
M. Barberis, G. Bourdon, D. Cohen,
M. Corless, S. Deland, D. Doherty,
J. Doke, G. Drayer Andrade, G. Dudgeon,
M. Evans, J. Friedman, M. Gemeinhardt,
J. Ghidella, C. Hahn, L. Harvey,
H. Heinzmann, T. Hubscher-Younger,
J. Hurst, K. Karnofsky, P. Kassebaum,
L. Kempler, T. Kush, J. Lanfrey, T. Leman,
D. Lim, D. Lluch, K. Lorenc, B. Mairs,
R. Mani, S. Miller, J.M. Modisette,
A. Nehemiah, S. Oliver, D. Orofino,
A. Pichard, A. Poon, O. Pujado, G. Rose,
R. Rovner, G. Sandmann, M. Schalk,
B. Sharma, L. Tabolinsky, S. Tamayo,
A. Turevskiy, X. Wang

SUBSCRIBE

mathworks.com/subscribe

CONTACT US

mathworks.com/contact

FIND US ONLINE





Large-scale commercial buildings can reduce energy costs by 10–25% with BuildingIQ's energy optimization system.

Data Analytics

Engineers and scientists are finding new and creative applications for data analytics technologies. They use machine learning, big data, and optimization to make critical decisions in medical diagnosis, stock trading, energy load forecasting, and more. In MATLAB® they can acquire and preprocess data from a variety of sources, build predictive models, compare machine learning algorithms, and integrate algorithms into production systems.

CZECH ACADEMY OF SCIENCES

Developing algorithms to reduce false alarms in ICUs

False alarms from electrocardiographs and other patient monitoring devices are a serious problem in intensive care units (ICUs). Noise from false alarms disturbs patients' sleep, and frequent false alarms desensitize clinical staff to genuine warnings. Competitors in the PhysioNet/Computing in Cardiology Challenge were tasked with developing algorithms that could distinguish between true and false alarms in signals recorded by ICU monitoring devices. Czech Academy of Sciences researchers won first place in the real-time category of the challenge with MATLAB algorithms that can detect QRS complexes (deflections of the heartbeat trace from its baseline in an ECG), distinguish between normal and ventricular heartbeats, and filter out false QRS complexes caused by cardiac pacemaker stimuli. The algorithms produced a true positive rate (TPR) and true negative rate (TNR) of 92% and 88%, respectively.

mathworks.com/czech-academy

FREIGHTOS

Analyzing big data for online freight logistics

Freightos developed an online freight routing and pricing system that uses Google® BigQuery to manage and store multiple databases for thousands of freight contracts, millions of freight quotes, and a wide array of other shipping data. Engineers export results from queries performed on BigQuery to the cloud, where they are downloaded and analyzed in MATLAB. One analysis evaluated 120,000 rows of freight quotes, each with more than 30 columns, to identify fluctuations in freight pricing based on different sales teams, companies, and shipping modes.

mathworks.com/freightos

BUILDINGIQ

Optimizing HVAC energy usage in large buildings

Heating, ventilation, and air-conditioning (HVAC) systems in large-scale commercial buildings are often inefficient because they do not

take into account changing weather patterns, variable energy costs, or the building's thermal properties. BuildingIQ's cloud-based software platform uses advanced algorithms to continuously process gigabytes of information from power meters, thermometers, and HVAC pressure sensors. Machine learning is used to segment data and determine the relative contributions of gas, electric, steam, and solar power to heating and cooling processes. Optimization is used to determine the best schedule for heating and cooling each building throughout the day. The BuildingIQ platform reduces HVAC energy consumption in large-scale commercial buildings by 10–25% during normal operation.

mathworks.com/buildingiq

MONDI

Statistics-based health monitoring and predictive maintenance for manufacturing processes

Mondi Gronau's plastic production plant delivers about 18 million tons of plastic and thin film products annually. Machine failures that result in downtime and wasted raw materials cost millions of euros each month. To minimize these costs and maximize plant efficiency, Mondi developed a health monitoring and predictive maintenance application that uses bagged decision trees and other machine learning algorithms to identify potential issues, enabling workers to take corrective action before a machine breaks down. A standalone executable version of the application is now used in production at the plant.

mathworks.com/mondi

ASML

Developing virtual metrology technology for semiconductor manufacturing

Photolithography is the process used to create the patterned layers of a silicon microchip. ASML's TWINSCAN photolithography system uses overlay control to ensure precise alignment of the layers. So many overlay marks are required for proper overlay model correction that it is not feasible to measure every wafer coming out of a TWINSCAN system. ASML developed virtual overlay metrology software that applies machine learning techniques to come up with a predicted estimate of overlay metrology for every wafer, using alignment metrology data. The network identified systematic and random overlay errors that might otherwise have gone undetected.

mathworks.com/asml

ABERDEEN ASSET MANAGEMENT

Implementing portfolio allocation models in the cloud

Aberdeen Asset Management bases many of its trade decisions and multi-asset class mandates on portfolio models generated with advanced machine learning algorithms. Analysts trained the models using factors such as monetary policy, corporate profits, interest

rates, and implied volatilities. They backtested the trained models on more than 15 years of historical data using MATLAB Distributed Computing Server™ in the Microsoft® Azure cloud. After refining the techniques and incorporating them into their asset allocation algorithms, they now run the algorithms with large financial data sets on a distributed computing cluster.

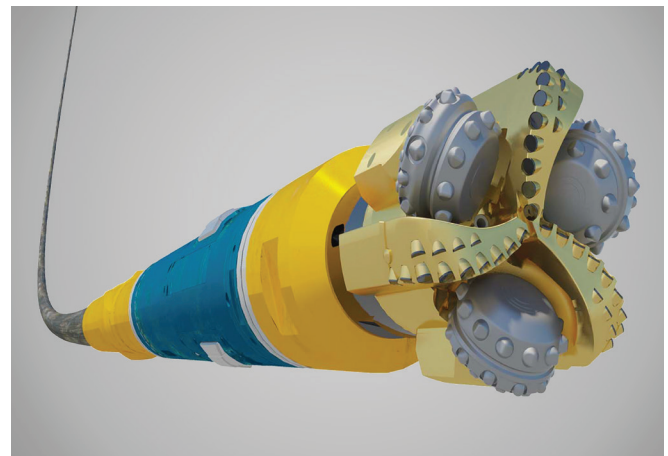
mathworks.com/aberdeen

BAKER HUGHES

Training neural networks to predict maintenance requirements for gas and oil extraction equipment

Baker Hughes trucks are equipped with positive displacement pumps that inject a mixture of water and sand deep into drilled wells. With pumps accounting for about \$100,000 of the \$1.5 million total cost of the truck, Baker Hughes needed to determine when a pump was about to fail. They processed and analyzed up to a terabyte of data collected at 50,000 samples per second from sensors installed on 10 trucks operating in the field, and trained a neural network to use sensor data to predict pump failures. The software is expected to reduce maintenance costs by 30–40%—or more than \$10 million.

mathworks.com/baker-hughes

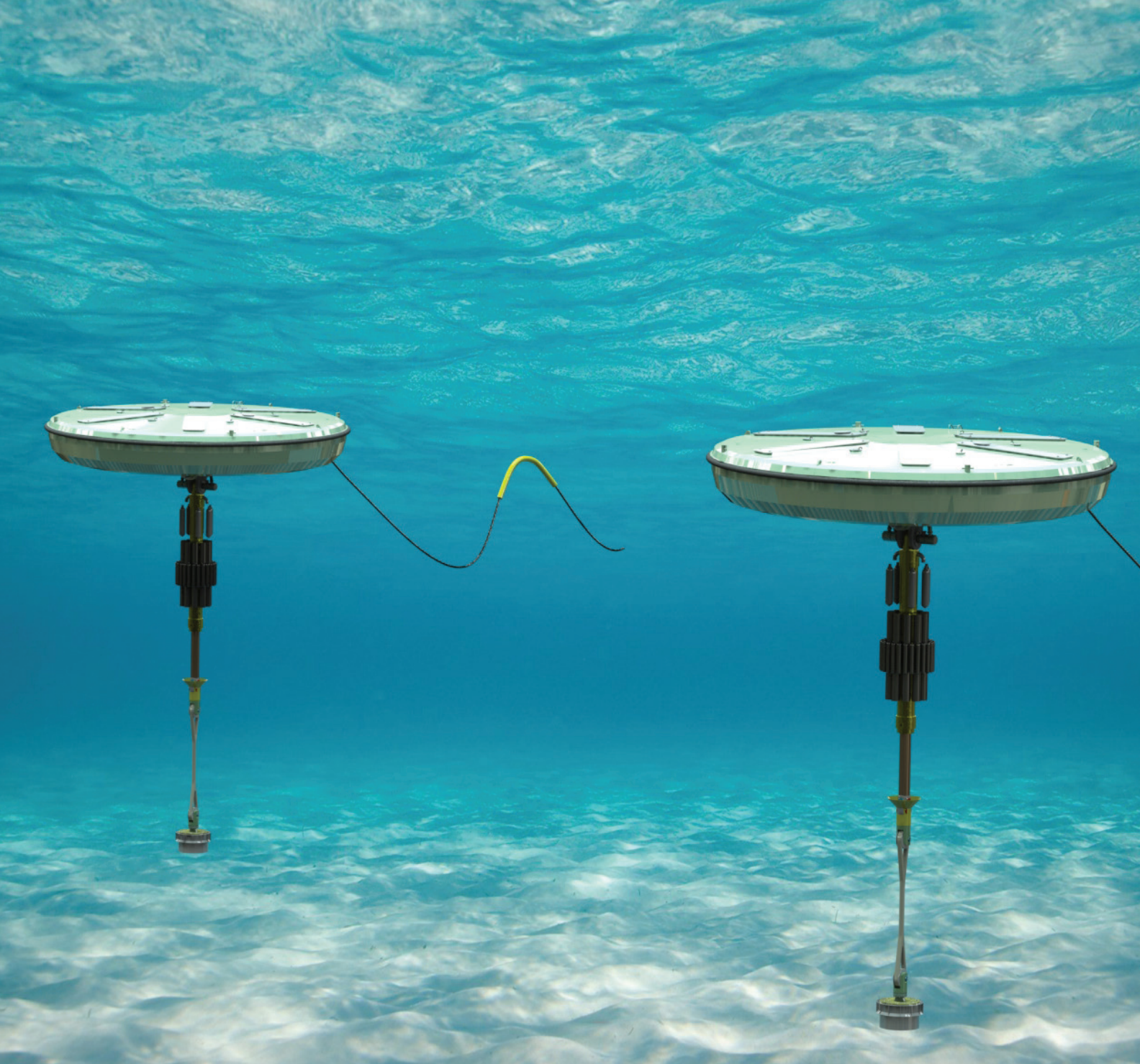


The AutoTrak™ Curve Rotary Steerable System, developed by Baker Hughes.

LEARN MORE

Data Analytics with MATLAB
mathworks.com/data-analytics

User Stories
mathworks.com/user-stories



Modeling and Simulating Next-Generation Wave Farm Technology

By Alexandre Pichard, Carnegie Clean Energy



For four seasons, the three submerged buoys

bobbing off the coast of Garden Island in Australia harnessed wave energy to generate electricity and fresh water for the country's largest naval base. The 14,000 hours of cumulative operation logged on the project established a world record for a grid-connected wave energy system and proved the viability and reliability of CETO 5 wave energy technology.

Last year, Carnegie Clean Energy successfully completed the objectives of CETO 5 deployment, and we are now developing the next generation of this technology. CETO 6 has the potential to generate more than four times the electric power of CETO 5 per buoy while lowering maintenance costs and enabling the wave farm to be located farther from shore.

Our design for CETO 6 is based on a full-scale wave-to-wire model of the entire multidomain system that we are building with Simulink® and Simscape™. We have steadily increased the technology readiness level for CETO 6 by using Simulink simulations to rapidly try out new ideas and iteratively improve designs. We use Simulink to model and simulate each step in the power generation chain, including the mechanics involved when a wave acts on the buoy, the conversion of mechanical energy into hydraulic energy, and the conversion of hydraulic energy into electrical energy.

Lessons Learned from CETO 5

CETO technology uses the movement of buoys to harness wave energy and generate electrical energy. The CETO 5 buoys were 11 meters in diameter and were submerged about two meters below the ocean's surface. Each buoy was connected via a tether to a hydraulic cylinder mounted on the sea floor. The vertical motion of the buoys drove a piston in the cylinder, creating pressure that forced water through underwater pipes to an onshore power generation facility (Figure 1). There, the high-pressure water was used to

drive hydroelectric turbines to generate electricity and produce desalinated water through reverse osmosis.

The year-long CETO 5 project not only proved the feasibility of the technology, it also provided us with a wealth of information that we are applying to CETO 6. CETO 5 was equipped with about 300 sensors that monitored pressures, flows, and temperatures in the hydraulic components; voltage and current frequency in the electrical components; and load, displacement, and acceleration of mechanical components. We analyzed and visualized the terabytes of data collected by the sensors in MATLAB®, using the results to inform our design decisions and validate our Simulink models.

Instead of installing and maintaining kilometers of high-pressure pipes between the wave farm and the shore, we can now use

an electrical umbilical to carry the generated power to land, lowering the cost of the system and enabling us to deploy the buoys much farther out to sea, where wave conditions are often more favorable for power generation. In CETO 6, the hydraulic pump is mounted on the buoy instead of on the sea floor (Figure 2). This change makes it easier to make repairs and perform maintenance because we can tow the entire system back to shore instead of working underwater.

Modeling and Optimizing the Power Takeoff

The heart of a CETO 6 system is the power takeoff, the subsystem that converts hydraulic energy from the pump into electrical energy. Our goal in designing the power takeoff is to extract the maximum amount of available power from wave energy. From theoretical

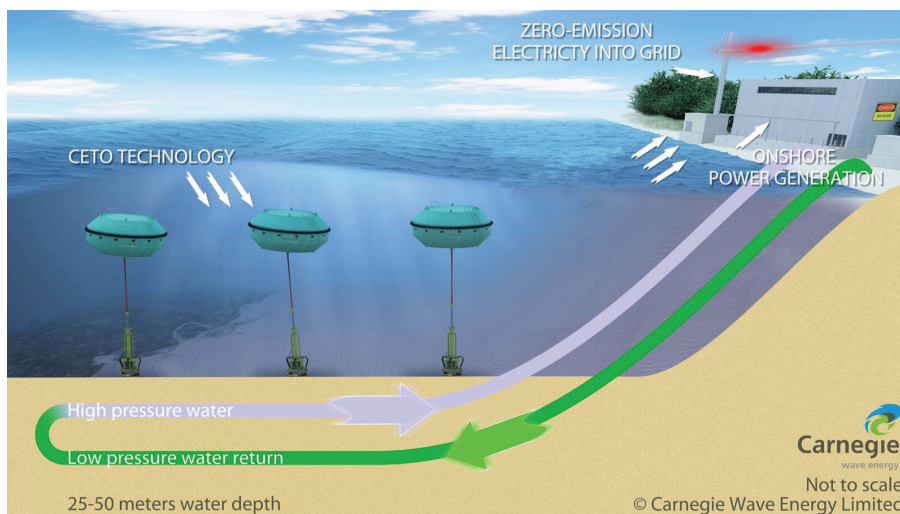


FIGURE 1. Schematic of a deployed CETO 5 system.

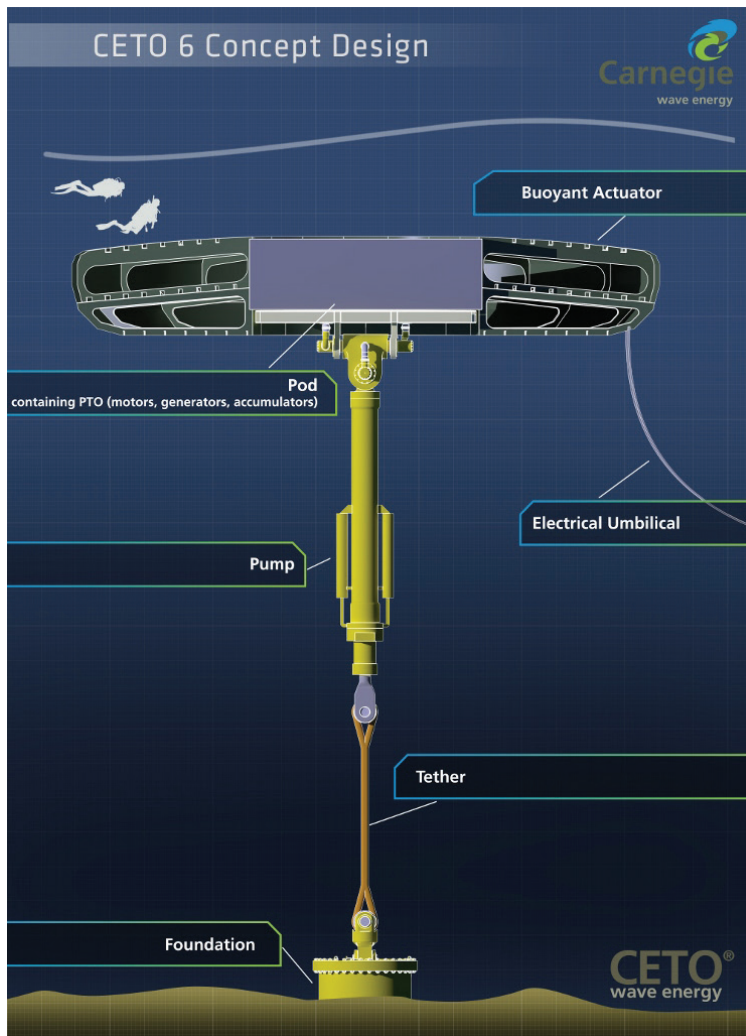


FIGURE 2. Diagram of a CETO 6 system.

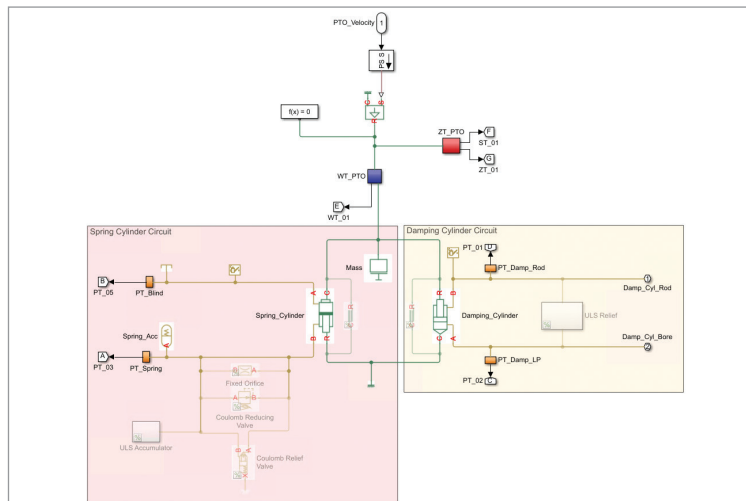


FIGURE 3. Simulink model showing the CETO 6 hydraulic components.

studies, we know how much power a perfectly efficient system could extract. Simulink models enabled us to configure and size a power takeoff subsystem that comes as close as possible to this ideal level of efficiency.

After modeling the mooring tether, linkages, and other mechanical components with Simscape Multibody™, we focused on the hydraulic subsystem. We created a complete model of this subsystem with Simulink and Simscape Fluids™ (Figure 3). The model includes multiple hydraulic accumulators, pipes, pressure release valves, and the hydraulic motor, which converts hydraulic pressure and flow into the torque that is used to drive the electric generator.

We ran tens of thousands of simulations in Simulink to optimize this design. We ran sweeps to find optimal configurations and parameter values, running them concurrently on a multicore processor using Parallel Computing Toolbox™. The simulations yielded some unexpected insights. For example, they revealed that a particular control parameter, which we assumed would need to vary significantly depending on the sea state, actually had little effect on performance, regardless of sea state. This insight enabled us to simplify our design and reduce the overall cost of the system.

Validating Simulation Results and Assembling a Wave-to-Wire Model

To test our early designs, we constructed a 1:20 scale version of the CETO 6 system and placed it in a wave tank, where we exposed it to carefully controlled wave conditions (Figure 4). During the tests we gathered data from sensors, as we had done with the real-world CETO 5 deployment, and analyzed the sensor data in MATLAB.

In preparation for developing a complete wave-to-wire model, we are using Simscape Electronics™ to model the electrical components of the system, including the generator and capacitors, which will be used for power smoothing. At the other end of the system, we have created a custom Simulink

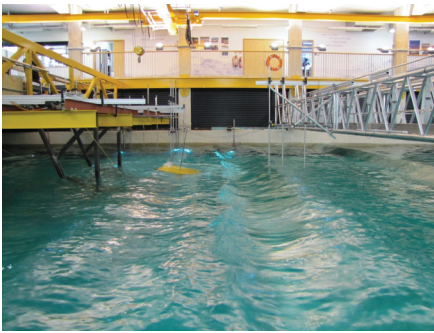


FIGURE 4. The scale prototype in the wave tank.

hydrodynamic model for simulating the interaction of the buoy with ocean waves. We now have an opportunity to transition from this custom model to WEC-SIM, an open-source wave energy converter simulation tool that is becoming more widely used across our industry. WEC-SIM was developed by the National Renewable Energy Laboratory (NREL) and Sandia National Laboratories using MATLAB, Simulink, and Simscape Multibody, which will make it easy to integrate into our workflow.

The complete wave-to-wire model will enable us to compute the displacement of the buoy for a specific wave profile, the pressure



FIGURE 5. Wave Hub marine energy testing facility in Cornwall, UK.

in the hydraulic cylinders from that displacement, the torque applied to the generator shaft by the hydraulic motor based on that pressure, and the electrical power produced by the generator from that torque.

CETO 6 Deployment and Beyond

Carnegie Clean Energy plans to deploy a grid-connected CETO 6 system at the Wave Hub marine energy development and testing facility located 16 km off the coast of Cornwall, in the UK (Figure 5).

We have also received a commitment from our local government for funding to deploy a wave energy farm off the coast of Albany in

Western Australia.

As we work toward those deployments, we continue to explore ways to improve CETO 6 technology. One improvement that we are currently evaluating is related to the optimization of the mooring configuration that is holding the buoy. The current CETO 6 vertical mooring line mainly captures energy from the buoy's heave motion (Figure 6).

This new configuration would capture pitch and surge motion, as well, potentially tripling the power output. Finally, with our company's recent expansion into microgrids, we are working on integrating CETO 6 wave energy technology with solar power and energy storage systems with the goal of developing and operating the world's first wave-solar-battery microgrid. ■

LEARN MORE

CETO Technology

www.carnegiece.com/wave/what-is-ceto

Data-Driven Insights with MATLAB Analytics: An Energy Load Forecasting Case Study

mathworks.com/energy-load

Modeling Wind Power Grid Integration (42:48)

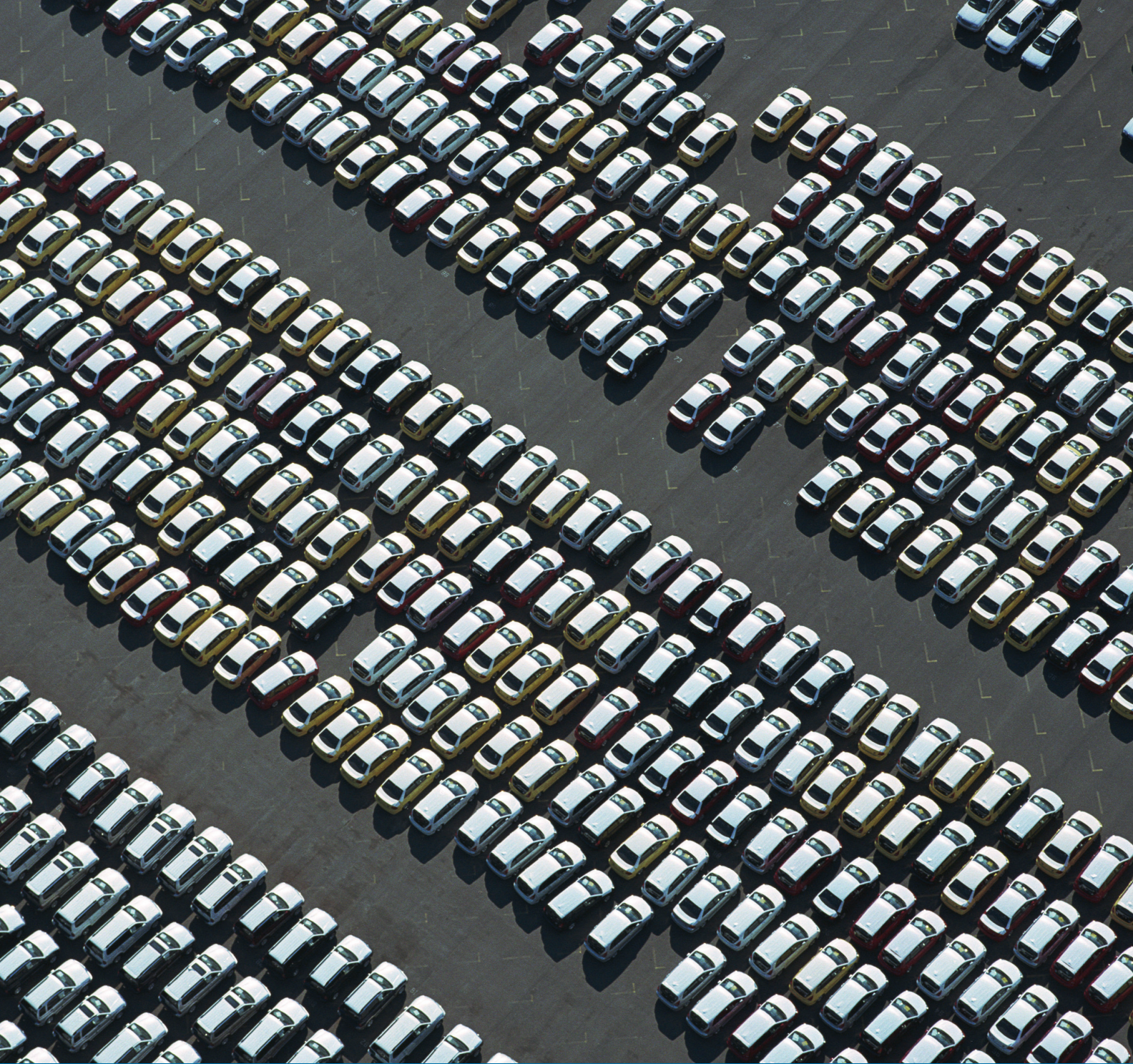
mathworks.com/video-81773

Modeling and Simulation of PV Solar Power Inverters (1:00:20)

mathworks.com/video-81813



FIGURE 6. Rendition of CETO 6 buoys showing mooring lines.



Optimizing Automotive Manufacturing Processes with Discrete-Event Simulation

By Marius Gemeinhardt, Daimler AG



Before new vehicles leave the production line

they undergo a series of end-of-line checks including static and dynamic tests. In static tests, both technicians and automated test procedures run electronic diagnostics; in dynamic tests, technicians, testing software, a dynamometer, and other test stations work jointly to check the engine and adjust the suspension or other components.

Orchestrating and coordinating the workers, machines, and vehicles involved in end-of-line testing is a complex task. Many companies do not have a formal method for optimizing the process, instead relying on the subjective recommendations of senior engineers; on best practices from other manufacturing plants, which may have different requirements; or even on trial and error.

To maximize production throughput and capacity while minimizing manpower and waste, I developed a platform for running simulations with Simulink® and SimEvents®. The simulations are used to aid operational decision-making, forecast the outcomes of proposed manufacturing process changes, and improve the efficiency of Daimler production lines (Figure 1).

Challenges of End-of-Line Test Optimization

Several factors complicate the optimization of end-of-line testing. First, it is difficult to estimate processing times at any given test station. Variances in suspension, for example, mean that some vehicles require more time at the suspension adjustment station than others. Second, introducing new equipment that can complete tests faster can also disrupt established processes. Likewise, introducing new technologies into the vehicles results in new optional extras that require new test procedures.

Third, the sheer complexity of the process improvement options available makes it almost impossible even for an expert to predict how changes will affect overall process performance. Adding workers, completing tests in parallel, handling reworked cars, inserting buffers (queues) before each test station, permitting vehicles to cross between test stations, advancing the cycle time—the expert would need to understand the effect of every possible combination of these options to find the best configuration.

Gathering and Managing Data

I knew that my simulations would need to take into account an immense amount of

data. Often in simulation studies, data is exchanged between disparate software packages, risking loss of precision and completeness. With MATLAB® and Simulink, I use the same environment for collection, analysis, and preparation of data as for the optimizations and simulations based on that data. Plus, I can accelerate processing by running analyses on multiple computing cores with Parallel Computing Toolbox™.

Each test station generates a log file for each vehicle. If 1000 vehicles are tested at three test stations, then 3000 data sets are logged. For a single vehicle on one station, the log files contain up to 200,000 lines of information. Each log file contains only a small fraction



FIGURE 1. A Mercedes-Benz S-Class vehicle leaving the assembly line.

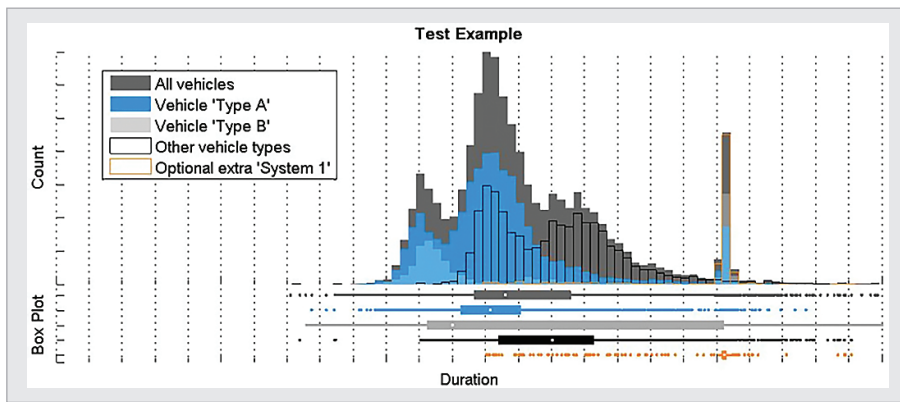


FIGURE 2. Histogram showing testing duration for various vehicle types.

of the necessary information, which includes vehicle details, the results of each test, and how long each test took to complete. To extract this data rapidly I create one DOS-based batch-file, call it for each log file, and distribute these jobs on each available core.

Analyzing the Existing Process

Before I developed the simulation, I needed to understand the current testing process. I collected the log files from every test station and analyzed the data numerically and graphically in MATLAB. I plotted histograms

and bar graphs of testing times and vehicle variations, and performed statistical analysis to correlate these variables (Figure 2). I accelerated the parsing and processing of the log files by a factor of almost four by using Parallel Computing Toolbox to execute these tasks on a four-core processor.

After interactively exploring and analyzing the data, I created an interface in MATLAB to simplify common analysis tasks (Figure 3). I packaged the interface and the analysis functions I developed in MATLAB as a standalone Microsoft® Windows® application, PARSE (Process Analysis Routine for Site-overlapping Exploration). Created with MATLAB Compiler™, PARSE enables my colleagues at Daimler to explore end-of-line testing data without installing MATLAB. PARSE also provides the database for the following modeling and simulation.

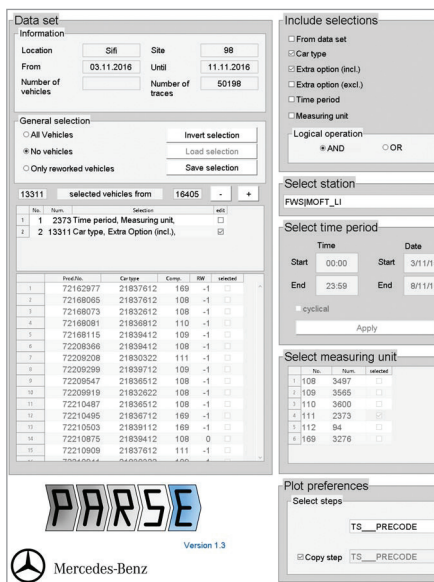


FIGURE 3. PARSE application, developed in MATLAB, for processing, analyzing, and exploring test station data.

base-line elements has the advantage that all functional, logical, and strategical behaviors of the modeled system are known from the beginning. A programmatic approach makes it possible to run optimization algorithms that can both adjust model parameters and generate new models. It also enables models to be defined via a second interface that I built in MATLAB.

This interface enables engineers to define testing processes by specifying the number and configuration of test stations, the number of workers, and so on. The engineer's selections are captured in a data model that the MATLAB script uses to generate a SimEvents model with station and worker subsystems (Figure 4).

In the generated model, which contains about 1500 blocks, worker and vehicle entities are brought together at each station with an entity combiner. The stations are represented by multiple single servers that represent individual processes within the station. The time spent at each station is calculated by an event-based random number block that uses an arbitrary discrete distribution based on the processed log data for that station.

Logical behaviors at the stations, as well as strategical control of the entities, are modeled using MATLAB scripts incorporated into the model as S-Function blocks. The model saves statistics from each station, including how many vehicles were processed, how long each vehicle spent at the station, and how much time it spent waiting between stations, as well as from peripheral processes such as delivery of vehicles, worker flow, and pause time. I use MATLAB to postprocess and visualize this data (Figure 5).

One of the first models I created using the interface and model generator simply replicated the existing factory setup with a database built on real-world raw data. I ran simulations of this model and compared its results with real-world results from the factory floor to validate the model and the model generation script.

Modeling End-of-Line Testing Processes

Most engineers create models for discrete-event simulations by linking together queue, server, entity, and other blocks from predefined libraries. The predefined elements in most simulation environments make it difficult to understand their fundamental functionality and their impact on the simulated system. I decided to take a different approach: I developed a MATLAB script to construct the SimEvents model programmatically. Building a model with SimEvents

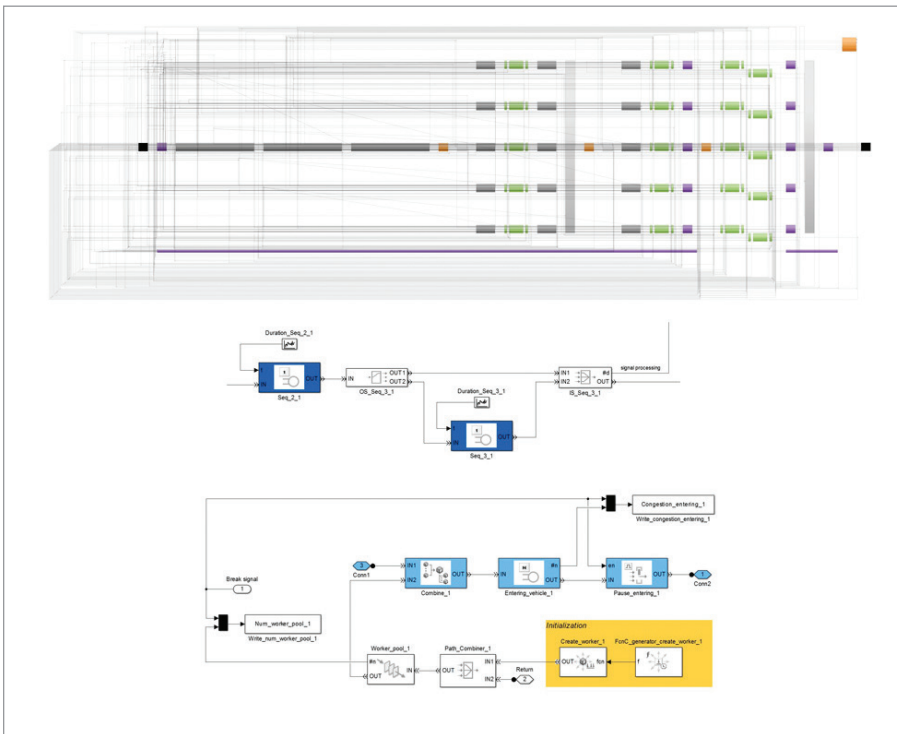


FIGURE 4. Top: An end-of-line testing process modeled in SimEvents. Middle: A station subsystem from the model. Bottom: A worker subsystem.

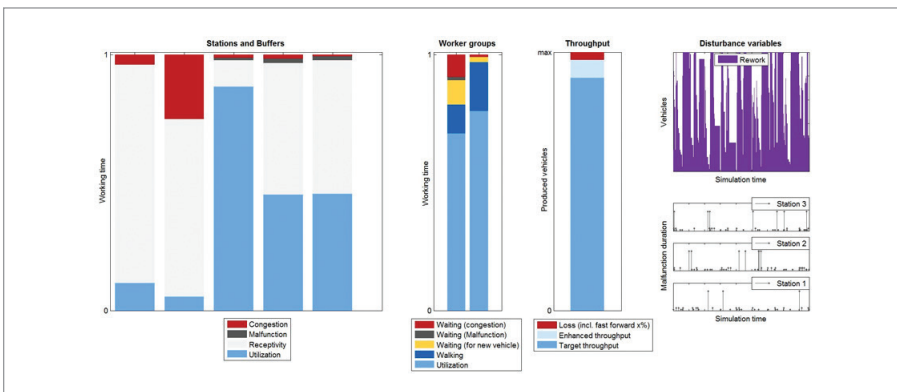


FIGURE 5. Visualizations of simulation results.

Running Simulations to Optimize the Process

Once I had a way to process and analyze log data and programmatically generate models, I could begin running systematic simulations to optimize end-of-line testing performance. In the simulations, the optimization algorithms make structural changes to reflect different factory layouts as well as parameter changes on in-

dividual test stations. I provide boundaries and initial values and then apply a pattern search algorithm in Global Optimization Toolbox to optimize for factors such as throughput, required production equipment, manpower, and waste. It would take thousands of experiments to assess all possible model variants. I can achieve the same results with a fraction of this number using the pattern search algorithm.

The SimEvents models enable me to adjust boundary values to run what-if scenarios. I run simulations, for example, to see how vehicle variations affect the time required for specific tests, enabling me to identify the variations that most affect process performance.

Traditionally, automotive manufacturers have expended considerable effort on shortening test durations, with little awareness of how end-of-line layouts affect the overall process. At Daimler, my simulations studies changed this. The simulations and optimizations I conducted with SimEvents provided insights into the influence of changes in plant structure. Before designing a new manufacturing plant, Daimler can now evaluate how factors such as the size of the provision area and buffers, the number of stations, enabling junctions, and personnel will affect plant testing performance. ■

LEARN MORE

Integrating Discrete-Event and Time-Based Models with Optimization for Resource Allocation
mathworks.com/resource-allocation

Lockheed Martin Builds Discrete Event Models to Predict F-35 Fleet Performance
mathworks.com/fleet-performance

Visualize, Label, and Fuse Sensor Data for Automated Driving

By Avinash Nehemiah and Mark Corless, MathWorks

>> Engineers developing perception algorithms for ADAS

or fully automated driving generally have a wealth of input data to work with. But this data, coming from many different types of sensors, including radar, LiDAR, cameras, and lane detectors, initially raises more questions than it answers. How do you make sense of the data, bring it to life? How do you reconcile conflicting inputs from different sensors? And once you've developed an algorithm, how do you evaluate the results it's producing?

Here's a guide to features and capabilities in MATLAB® and Automated Driving System Toolbox™ that can help you address these questions. We'll focus on four key tasks: visualizing vehicle sensor data, labeling ground truth, fusing data from multiple sensors, and synthesizing sensor data to test tracking and fusion algorithms.

Visualizing Vehicle Sensor Data

Making sense of the data is the primary challenge in the early stages of perception system development. Sensors deliver their output in different formats and at different rates. For example, cameras provide images in 3D matrices, LiDAR provides a list of points, and embedded or intelligent cameras provide object lists with details on vehicles, lanes, and other objects. The disparate nature of these outputs makes it difficult to see the overall picture (Figure 1).

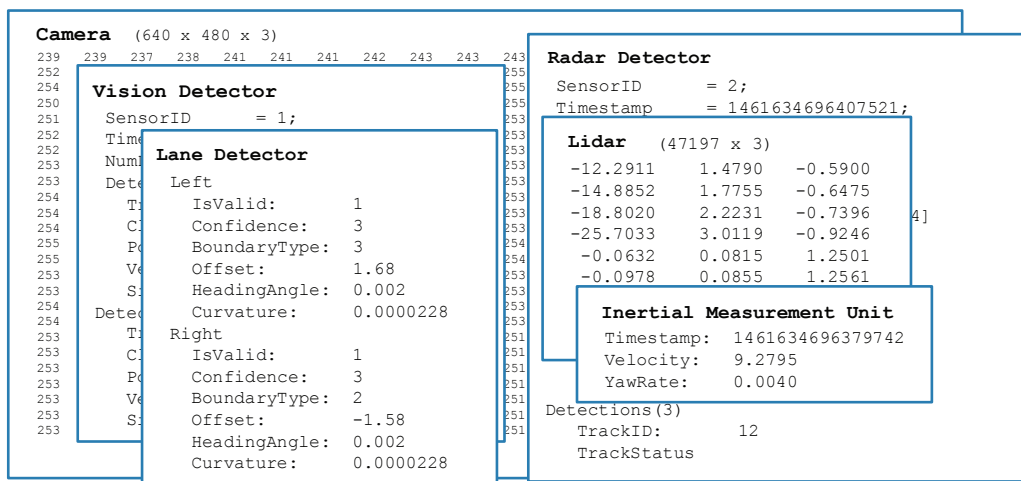


FIGURE 1. Examples of vehicle sensor data.

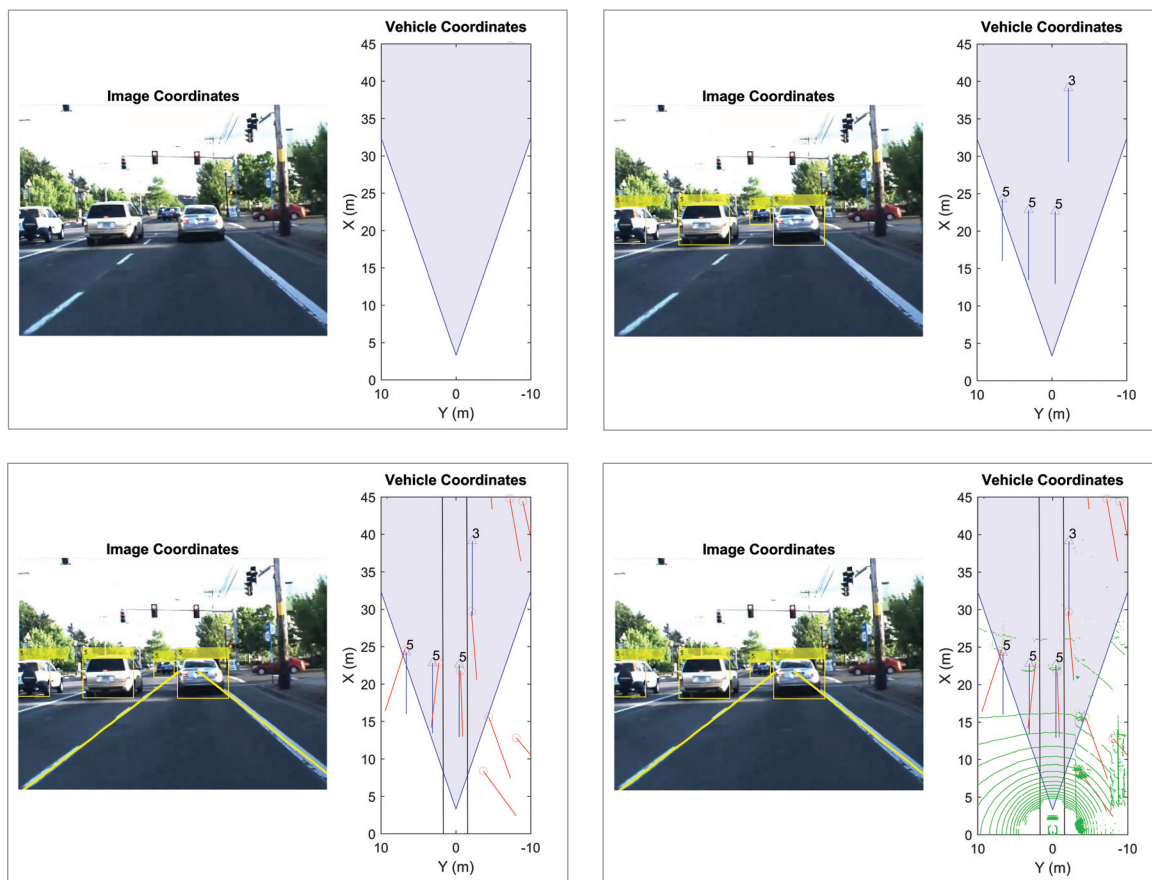


FIGURE 2. (Clockwise from top left) Plotting the sensor coverage area, transforming vehicle coordinates to image coordinates, plotting lanes and radar detections, and plotting the LiDAR point cloud.

At this early stage, we need to know exactly how the sensors are representing the environment around the vehicle. The best type of visualization to use for this is a bird's-eye plot because it allows us to visualize all the data from the different sensors in one place.

To create birds-eye plots we use the visualization tools in MATLAB and Automated Driving System Toolbox. We then add more detail to the views with these objects:

- The `coverageAreaPlotter`, which displays the sensor coverage area
- The `detectionPlotter`, which displays lists of objects detected by vision, radar, and LiDAR sensors
- The `laneBoundaryPlotter`, which overlays lane detections onto images

We now have accurate visualizations of sensor coverage, detections, and lane boundaries (Figure 2).

Automating Ground Truth Labeling

Ground truth is required to train object detectors using machine learning or deep learning techniques. It is also essential to evaluate ex-

isting detection algorithms. Establishing ground truth is often a labor-intensive process, requiring labels to be inserted into a video manually, frame by frame. The Ground Truth Labeler app in Automated Driving System Toolbox includes computer vision algorithms to accelerate the process of labeling ground truth. The app has three key features (Figure 3):

- The **vehicle detector** automates the detection and labeling of vehicles in keyframes by using an aggregate channel feature (ACF).
- The **temporal interpolator** labels the objects detected in all frames between selected keyframes.
- The **point tracker** uses a version of the Kanade-Lucas-Tomasi (KLT) algorithm to track regions of interest across frames.
- The **add algorithm** lets you add custom algorithms and facilitates iterative development of object detectors.

Fusing Data from Multiple Sensors

Virtually every perception system uses input from several complementary sensors. Reconciling data from these sensors can be challenging

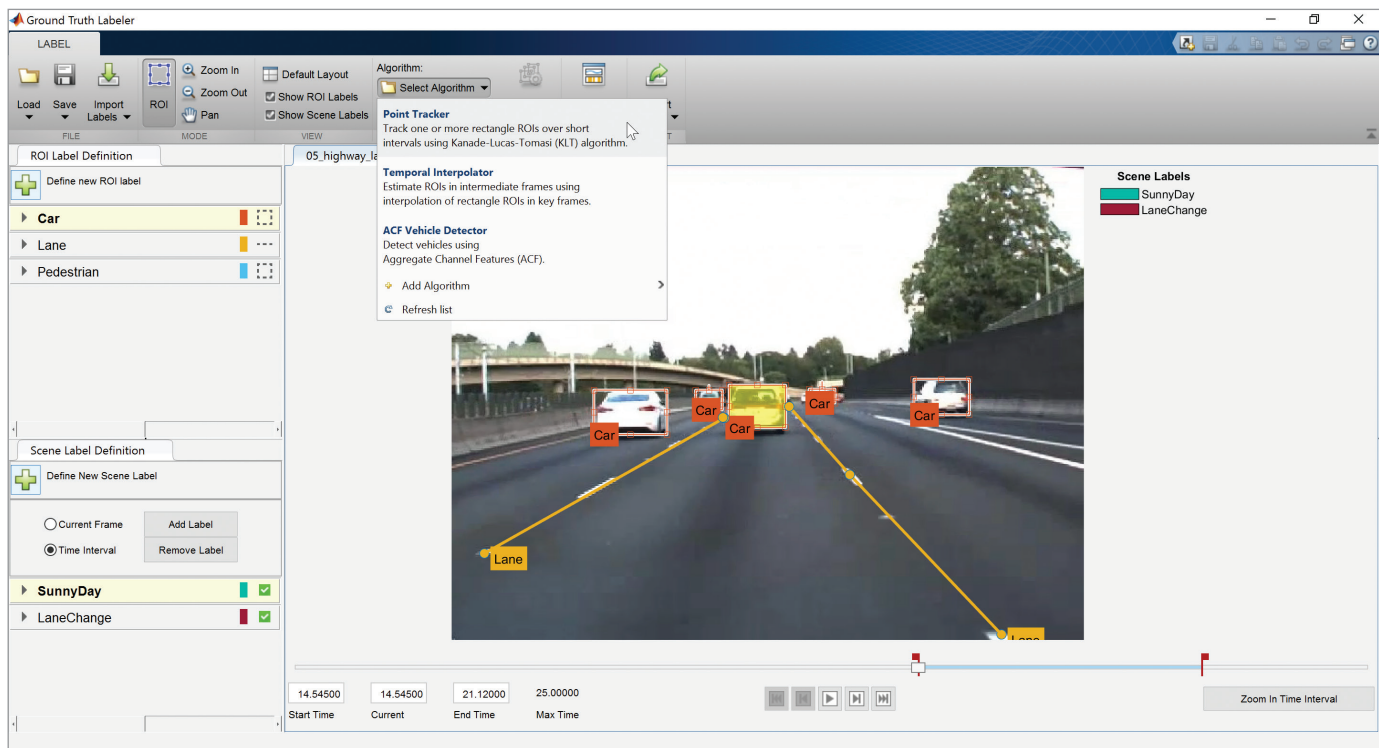


FIGURE 3. The Ground Truth Labeler app.

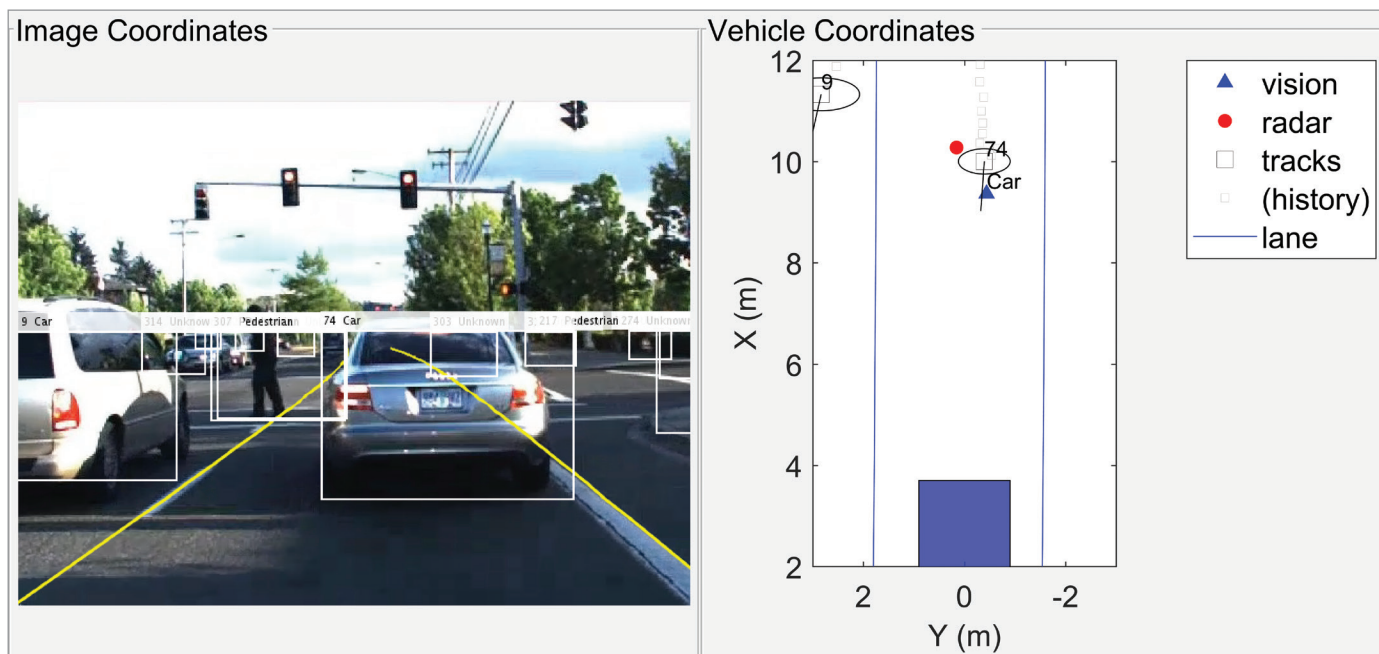


FIGURE 4. The multi-object tracker, used here to fuse radar data (red circle) and vision detection (blue triangle) data to produce a more accurate estimate of the vehicle's location (black oval).

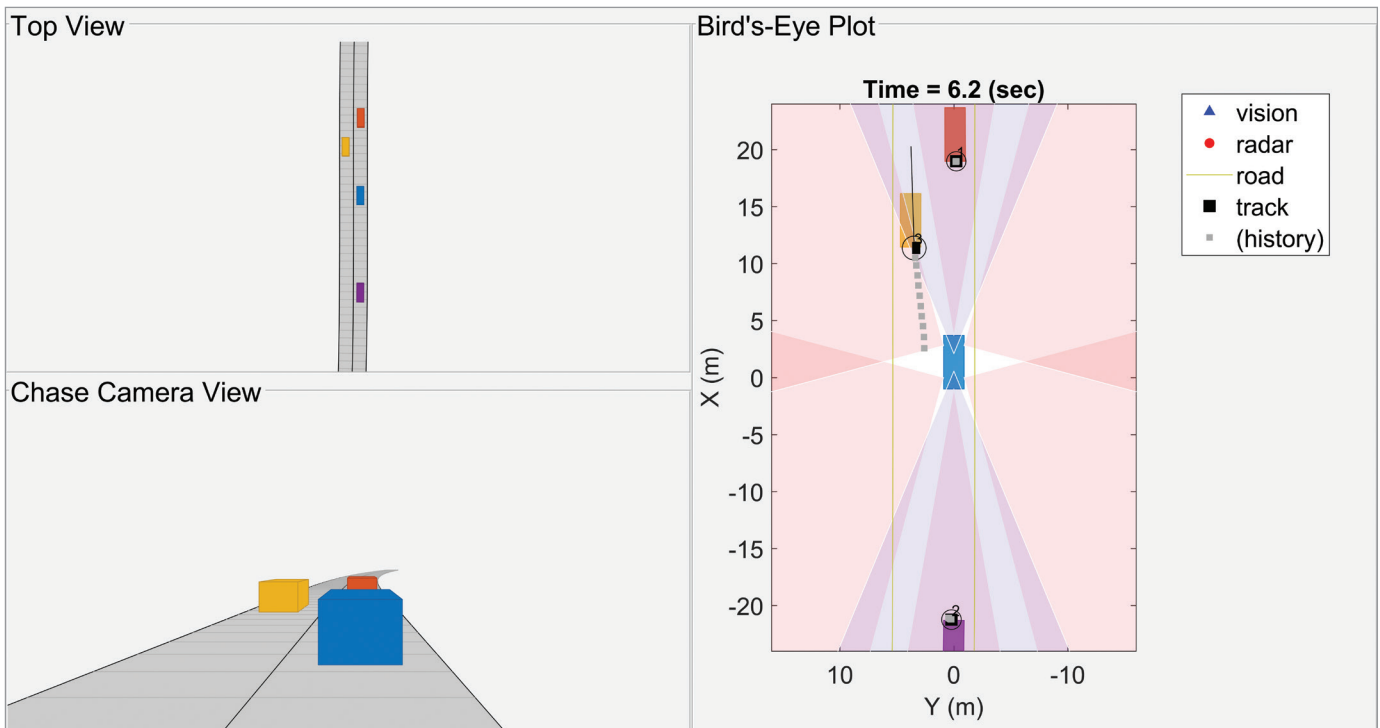


FIGURE 5. Top view, chase camera view, and birds-eye plot of a synthesized test scenario.

because each one is likely to give a slightly different result—for example, the vision detector might report that a vehicle is in one location, while the radar detector shows that same vehicle in a nearby but distinctly different location.

The `multiObjectTracker` in Automated Driving System Toolbox tracks and fuses detections. A common application is to fuse radar and vision detections and improve the estimated position of surrounding vehicles (Figure 4).

Synthesizing Sensor Data to Generate Test Scenarios

Some test scenarios, such as imminent collisions, are too dangerous to execute in a real vehicle, while others may require overcast skies or other specific weather conditions. We can address this challenge by synthesizing object-level sensor data to generate scenarios that include roads, vehicles, and pedestrians as virtual objects. We can use this synthetic data to test our tracking and sensor fusion algorithm. (Figure 5).

Using Vehicle Data in Perception Systems

Visualizing, fusing, and synthesizing vehicle data lays the groundwork for developing object detection algorithms. When we are ready to de-

ploy the MATLAB algorithms, we can use MATLAB Coder™ to generate portable, ANSI/ISO compliant C/C++ code for integration into our embedded environment. ■

LEARN MORE

Developing Advanced Emergency Braking Systems at Scania
mathworks.com/advanced-braking

Traffic Sign Recognition for Driver Assistance Systems (21:46)
mathworks.com/video-108102

Automated Driving Code Examples
mathworks.com/adas-code

Forward Collision Warning Using Sensor Fusion
mathworks.com/collision-warning



Accelerating Drone Research with a Ready-to-Fly Hexacopter and Flight Control Software

By Greg Rose, Tyler Leman, and Bryant Mairs, IntelinAir, and Xiaofeng Wang, University of South Carolina College of Engineering and Computing



Aerial robotics researchers are using drones to develop

ground-breaking flight control algorithms and novel solutions to problems in many fields, including emergency response, home healthcare, and precision agriculture. However, projects are often slowed down by the time researchers must spend implementing basic capabilities such as sensor data processing, orientation and altitude calculation, and in-flight navigation. We developed the IntelinAir RD-100 hexacopter to address this issue.

The RD-100 is a ready-to-fly drone with pre-built autopilot flight software developed in Simulink® (Figure 1). Capable of maintaining stability in flight as well as automating take-offs, landings, and waypoint navigation, the flight software can be customized to meet specific research goals.

By using Model-Based Design with the RD-100 development environment, research teams can rapidly prototype, simulate, and deploy control software and jumpstart new drone research programs. They have full access to the Simulink models and can add

features, implement cooperative flight algorithms and other advanced applications, and verify their designs and algorithms in hardware-in-the-loop (HIL) simulations. Researchers can move from simulation directly to actual flight tests, and it is this capability that sets the RD-100 apart from other drones in its class.

Developing RD-100 Flight Software with Model-Based Design

The RD-100 is designed to simplify flight software development. Model-Based Design

is a natural fit for our development approach because it makes it easy for researchers to use the models we provide to run simulations, run HIL tests, and generate flight code. We could have written the code by hand, but we believe that Model-Based Design is not only the fastest way to develop flight software, it is also preferred by members of the academic research community, who are already comfortable using MATLAB® and Simulink.

IntelinAir engineered the ground station used to communicate with the drone, the hexacopter hardware, and the flight software. One



FIGURE 1. The IntelinAir RD-100 drone.

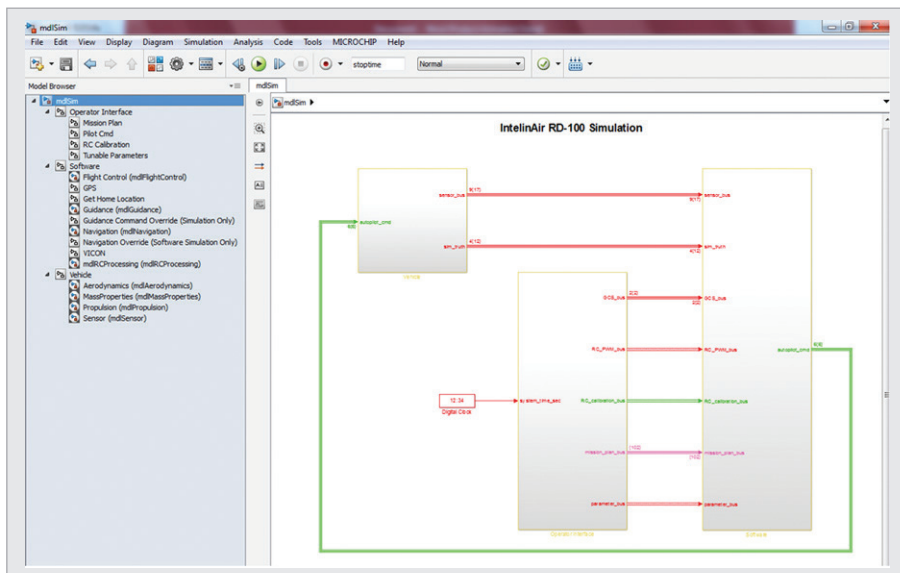


FIGURE 2. Top-level Simulink model showing blocks for the vehicle model, operator interface, and flight control software (created with IntelinAir’s SafeSmart Toolbox).

group developed a six-degrees-of-freedom (6 DOF) Simulink model of the drone, incorporating the airframe, motors, and electronic speed controls. This model has blocks for the drone’s mass properties, propulsion (thrust), aerodynamics, equations of motion, and sensors, including the accelerometer, gyroscope, magnetometer, barometer, and GPS receiver (Figure 2).

Another group modeled the flight control system. This model, developed with Simulink and Control System Toolbox™, includes a navigation block that uses sensor input to calculate the drone’s velocity, orientation, and position. It also includes a guidance block for autonomous flight modes as well as a motor control block that generates commands based on input from the navigation block and the operator interface. To simplify the design and facilitate reuse, the flight software group organized the models hierarchically using Simulink model referencing and created a custom Simulink library of components frequently used in the design.

A major challenge in drone control design is dealing with changes in system dynamics caused by high winds and changing payloads. RD-100 addresses this challenge with L1

adaptive control, an advanced technique developed by IntelinAir co-founder Dr. Naira Hovakimyan at the University of Illinois at Urbana-Champaign. By decoupling the adaptation loop from the control loop, L1 adaptive control enables the RD-100 to rapidly compensate for undesirable effects. This allows for stable, precise flight in even the most challenging conditions and is critical to accurate data collection.

From Desktop Simulation to HIL and Flight Testing

When using the RD-100 software, researchers follow an iterative process of control design, software simulation, HIL simulation, and flight tests (Figure 3).

This process starts with closed-loop software simulations in Simulink. These produce plots of the performance of the aircraft during simulation (Figure 4).

After verifying the basic functionality of the control algorithms on the desktop, the researcher can test the algorithms on hardware in HIL simulations without leaving the desktop. C code is generated from the controller model using Embedded Coder® and deployed to the RD-100 autopilot hardware. C code

from the 6 DOF vehicle model, generated using Simulink Coder™, is deployed to target hardware running Simulink Real-Time™. During HIL simulations, the flight software running on the drone autopilot processor receives sensor input generated by the vehicle model running in Simulink Real-Time. Telemetry data captured during the simulations is logged and analyzed offline in MATLAB for verification and validation. One of the many benefits of HIL testing is that we catch bugs due to incompatibilities with hardware prior to actual flight.

Following HIL tests, the autopilot hardware running the flight control software can simply be unplugged from the HIL test setup on the desktop and plugged into the RD-100 air frame for real-world flight tests. During flight tests, the flight software receives input directly from the onboard sensors and sends commands directly to the motors.

Advantages of Model-Based Design

One of the principal advantages of Model-Based Design is that it enables small teams to tackle projects that typically require a much larger group to complete. That holds true for our development team at IntelinAir and for the numerous academic groups already using the RD-100 in their research. Because no coding expertise is required, controls engineers and researchers can test novel control ideas without involving an embedded software engineer. New algorithms can be tested and debugged on the desktop and in real time. As a result, the engineer can be certain that the small unmanned aircraft system will work.

Example Research Project: Precision Agriculture

IntelinAir and Dr. Wang are partnering to push the envelope in remote sensing for precision agriculture. The impetus for Dr. Wang’s research is the idea that drones will one day be as ubiquitous and commonplace as smartphones.

Today, thanks to the Internet of Things, we have access to vast amounts of useful data. We

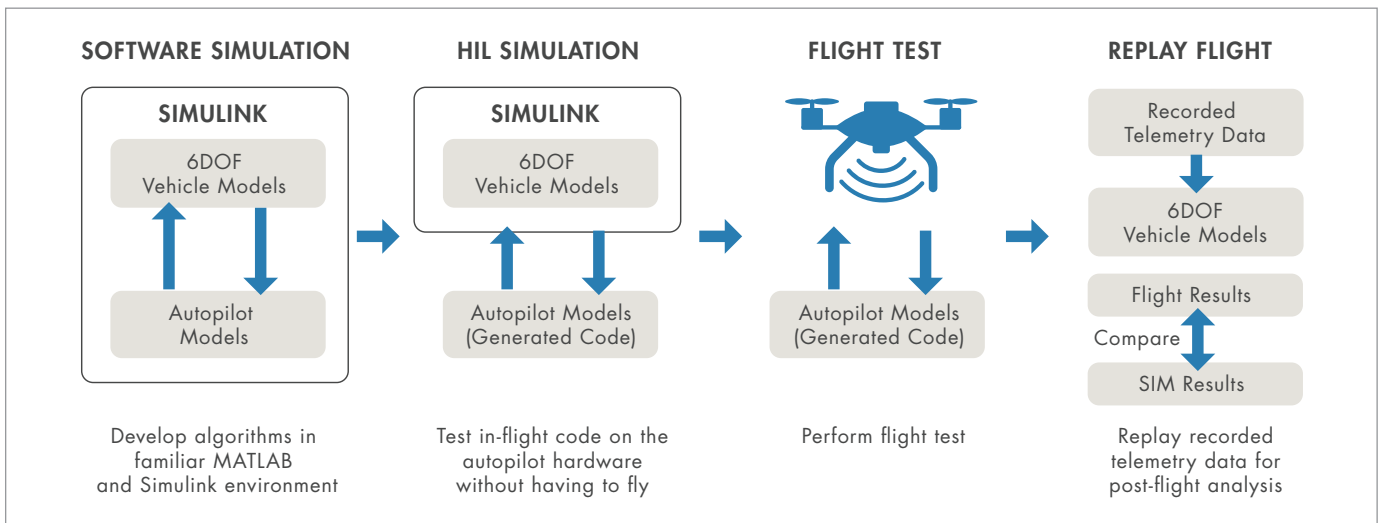


FIGURE 3. Workflow for moving from simulation to HIL testing and flight tests.

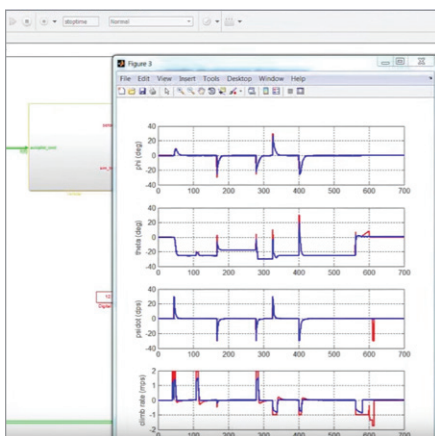


FIGURE 4. Plot showing commands for thrust, yaw moment, roll moment, and pitch moment during simulation.

train computer programs to make decisions based on this data, but we still rely on humans to act on those decisions—for example, in agriculture we often rely on human crop advisors to scout fields where they can only cover a small percentage of the total acreage. Aerial robots will automate this final step, completing a feedback loop that will include Internet-connected devices, decision-making software, and autonomous drones.

The most promising applications for drone technology exist in industries where vast geographies, redundant tasks, and complicated

data analysis obstruct efficiency. One such industry is agriculture, where drone technologies offer significant opportunities to improve efficiency and productivity.

Dr. Wang’s team is working with IntelinAir to explore agricultural applications of drones equipped with miniature soil moisture sensors and high-resolution multispectral cameras. Though not widely used in agriculture today, this technology has the potential to increase productivity and lower the costs associated with labor, nutrients, and irrigation. New sensor technology offers highly accurate measurements for humidity and nutrients. This data can be used to detect yield-robbing anomalies in the field in the middle of the growing season, when there’s still time to intervene.

Once data is captured, it is synthesized by a well-trained convolutional neural network using IntelinAir’s proprietary algorithms and transmitted to the farmer, who then has a detailed overview of crop health to guide decision-making.

This research will encompass farms in Illinois, California, and South Carolina, with the ultimate goal of developing a sensor-equipped unmanned aircraft system that is affordable and convenient for farmers.

Dr. Wang’s team of three Ph.D. students, one master’s student, and three undergradu-

ates develops flight software using Model-Based Design and the RD-100 drone. In the past, when the team used other drones, students had to write C or C++ manually. With the Simulink control models provided with the RD-100, students don’t waste time programming or debugging C code. Instead, they can implement their designs and ideas by simply modifying the models. They can run simulations and HIL tests on the desktop to verify the designs, and then move directly to flight tests on the RD-100. ■

LEARN MORE

Designing a Nonlinear Feedback Controller for the DARPA Robotics Challenge
mathworks.com/darpa

Drones That Can Help Fight Wildfires
blogs.mathworks.com/headlines/?p=417

Quadcopter Simulation and Control Made Easy (37:56)
mathworks.com/video-93365

Introduction to Simulink Quadcopter Simulation and Control (36:39)
mathworks.com/video-100476

MATLAB Licensing for Campus-Wide Use
mathworks.com/matlab-campus

Modeling the Emerging 5G Standard with the 5G Library

By Marc Barberis, MathWorks

The 5G wireless communication standard will provide significantly higher mobile broadband throughput with its enhanced Mobile Broadband (eMBB) mode. While the details are still under discussion, several techniques and features have already emerged as candidates for 5G Release 15, the first version of the 5G standard. Among the key elements of the new standard are:

- New waveforms with improved out-of-band emissions (OOBE)
- Shorter slot durations, corresponding to increased subcarrier spacing, for increased signal bandwidth and shorter latency
- New coding schemes such as LDPC for data and polar codes for control information
- The use of OFDMA as well as SC-FDMA on the uplink

These elements have the potential to improve system efficiency, but they add complexity to your design. Due to the increased bandwidth and operating frequencies, RF components represent a significant challenge, and it is crucial to assess the impact of RF imperfections on the whole system. An additional complication is the high number of antennas required for massive MIMO.

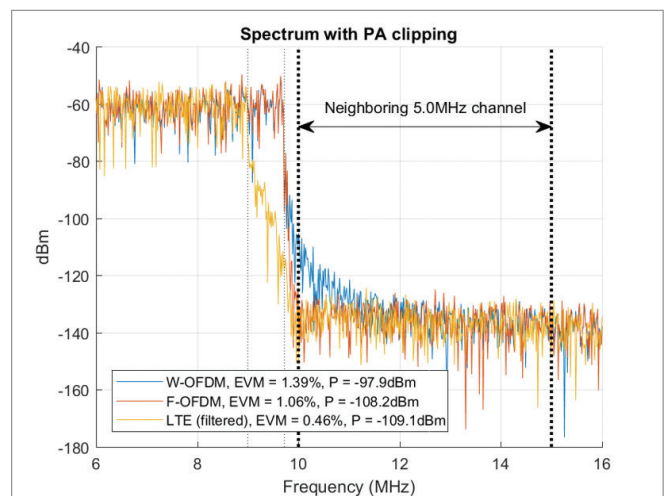
How do you know whether the techniques proposed for 5G will be suitable for your design? Should you implement them all, or just a subset?

Using the 5G library, you can measure the impact of different algorithms and design choices by simulating end-to-end system performance over realistic 5G propagation channels. When used with RF Blockset™, Antenna Toolbox™, and Phased Array System Toolbox™, the 5G library lets you investigate design tradeoffs at an early stage.

What's in the 5G Library

The 5G library is a downloadable add-on for LTE System Toolbox™ that provides executable versions of features defined by the new 5G standard, including:

- New waveforms for more efficient spectrum use
- Propagation channel models for the 0.5 to 100 GHz frequency range
- Coding schemes, including LDPC for data channels and polar codes for control channels
- A customizable end-to-end link simulation example with configurable subcarrier spacing



Spectrum of 5G and LTE waveforms, accounting for power amplifier (PA) clipping, generated with LTE System Toolbox and the 5G library.

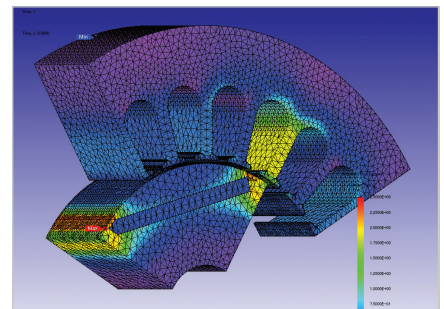
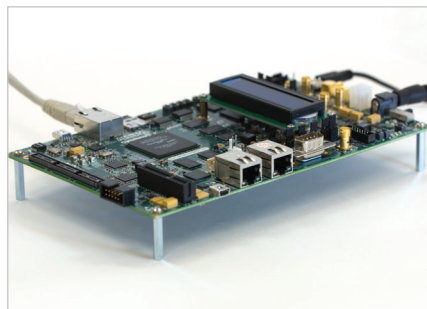
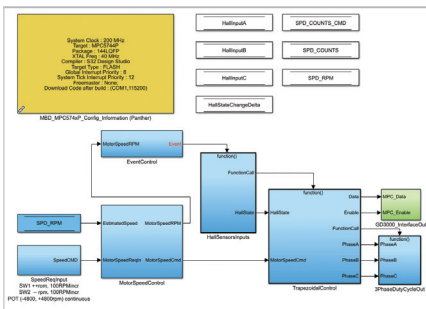
LEARN MORE

5G Library (5:54)
mathworks.com/video-5g-library

Evaluating 5G Waveforms Over 3D Propagation Channels with the 5G Library
mathworks.com/5g-waveforms

Solutions for Developing Custom Motor Control Applications

By combining MATLAB® and Simulink® with third-party products®, engineers can develop and deploy motor control applications using Model-Based Design. They can design control algorithms graphically in Simulink and then simulate them alongside high-fidelity models of motors and blocks representing controller hardware peripherals. They can generate C or HDL code from the algorithms to run on MCU or DSP controller hardware, synthesize to FPGAs, or deploy on SoC architectures.



NXP: Model-Based Design Toolbox

NXP's Model-Based Design Toolbox is a toolchain for configuring and generating software to execute motor control algorithms on NXP MCUs. The toolbox provides a Simulink blockset for peripheral devices such as PWM, A/D, and CAN, as well as an optimized motor control blockset that includes functions such as Park/Clarke transforms and digital filters. The blocksets are integrated with an Embedded Coder® target for generating and deploying code onto NXP controllers and performing software-in-the-loop and processor-in-the-loop testing.

nxp.com/mctoolbox

Microchip: Motor and Controller Model Libraries, MPLAB® Device Blocks

Microchip provides blocksets that allow the simulation of motor control algorithms running on dsPIC® digital signal controllers. The Motor Control Library Blockset contains Simulink blocks for motor control applications, including reference frame transforms, a proportional-integral controller, and trigonometric functions. The Motor Model Library adds a Simulink model for simulating perma-

nent-magnet synchronous motors (PMSMs). For deploying control algorithms onto dsPIC hardware, Microchip's MPLAB® Device Blocks for Simulink provide peripheral blocks for digital/analog I/O, counters and timers, pulse width modulation (PWM) motor control, and more. You can add and configure these blocks in Simulink models and then generate C/C++ code to run on dsPIC/PIC devices.

microchip.com/simplified

Intel: FPGAs, SoC FPGAs, and DSP Builder

Intel provides motor control tools to target both conventional FPGAs and SoC FPGAs that combine programmable logic with an ARM® hard processor. You can design control algorithms in Simulink and then use either HDL Coder™ or Intel's DSP Builder for Intel® FPGAs to generate HDL code for Intel FPGAs. Using Embedded Coder and related support packages you can generate C/C++ code for the ARM cores on FPGA SoC platforms. Intel's Drive-on-a-Chip reference design includes Simulink models of motor control algorithms and physical models of motors for system simulation and VHDL code generation. The reference design supports Intel MAX 10 and

Cyclone® V FPGAs as well as Cyclone V SoC FPGAs, with built-in support for motor control development kits.

altera.com/simulink-motor-control

JSOL: JMAG

JMAG finite element analysis software is used for developing electromechanical equipment such as motors, power converters, and actuators. JMAG can simulate magnetic flux density and electromagnetic forces in a range of motors, including permanent magnet, induction, and stepper motors. JMAG-RT extracts motor features as a precise reduced-order model provided as a Simulink block for motor control development. High-fidelity JMAG-RT models capture device performance, including nonlinear effects, saturation, and space harmonics.

jmag-international.com

LEARN MORE

Motor Control Design with Simulink
mathworks.com/motor-control-design

Third-Party Products and Services
mathworks.com/connections



Teaching Mechatronics with MATLAB, Simulink, and Arduino Hardware

By Joshua Hurst, Rensselaer Polytechnic Institute



Mechatronics is such a broad discipline that teaching

it at the undergraduate level presents a significant challenge. We must give our students a foundation in theoretical analysis that spans mechanical, electrical, and computing domains. At the same time, we must provide hands-on projects that integrate sensors, actuators, and embedded real-time control systems.

At Rensselaer Polytechnic Institute (RPI), my colleagues and I have addressed this challenge with a mechatronics curriculum that incorporates a take-home kit for lab assignments and course projects. The Rensselaer Mechatronics Kit includes an Arduino® based microcontroller, a DC motor, a set of sensors, and MATLAB® and Simulink® software (Figure 1).

In my *Mechatronics* course, fourth-year undergraduates use the kit to design and implement a real-time control system for MinSeg, a two-wheeled, self-balancing robot (Figure 2).

MATLAB, Simulink, and Simulink Support Package for Arduino Hardware enable the students to focus on high-level system design for

inverted pendulum control even as they learn to deal with real-world effects such as saturation, discretization, and measurement delays.

What Inspired the Inverted Pendulum Project and the Mechatronics Kit

As a student at RPI, I took a controls course taught by Dr. Kevin Craig, a mechatronics pioneer. Dr. Craig demonstrated an inverted pendulum system that balanced a broomstick on one end. I remember thinking it was one of the most amazing things that I had ever seen, and I was inspired to learn what it would take to make a system like that work. When I

began teaching at RPI, I wanted to design lab exercises that sparked this same enthusiasm in my students.

Many of the existing lab assignments in the mechatronics course required students to write C code by hand. Maintaining the labs required a great deal of instructor effort, and students spent much of their time debugging code instead of learning to apply the concepts learned in class. All too often, students reverted to trial-and-error rather than theory-based analysis to achieve the desired experimental outcomes.

I could have addressed this issue by simplifying the lab assignments—minimizing real-world effects in order to demonstrate

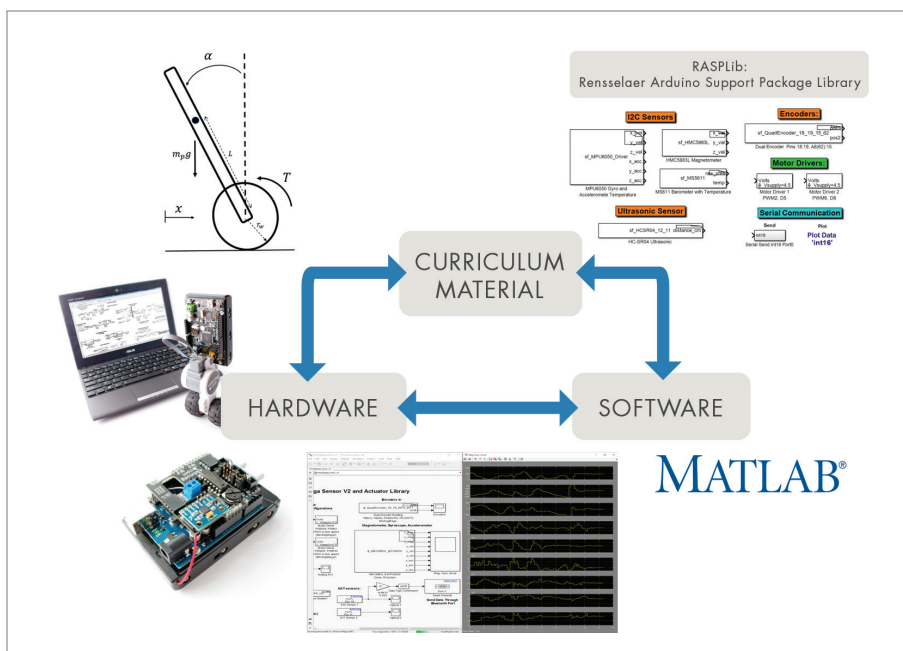


FIGURE 1. Rensselaer Mechatronics Kit components.



FIGURE 2. The MinSeg inverted pendulum robot.

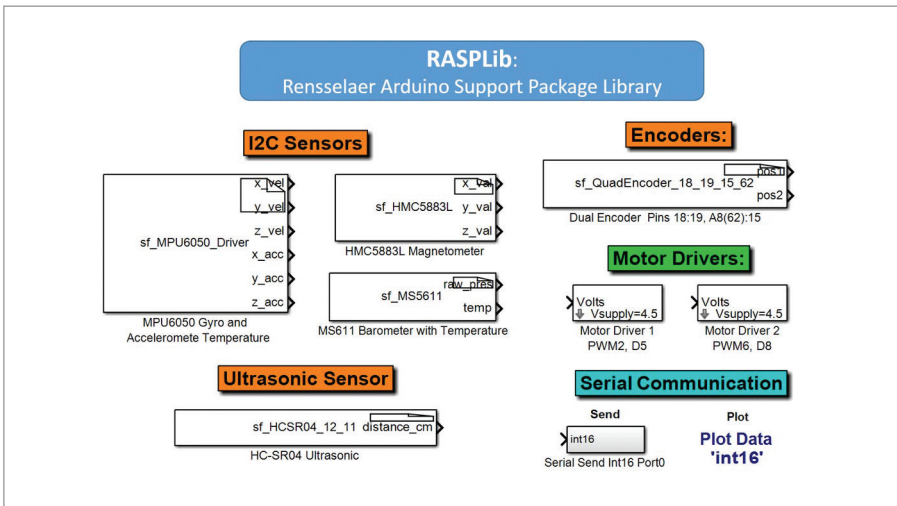


FIGURE 3. A subset of sensor and actuator blocks available in the Rensselaer Arduino Support Package Library.

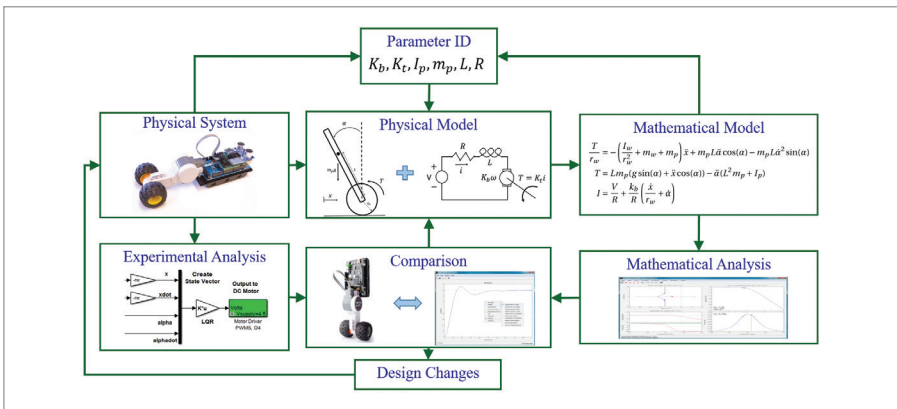


FIGURE 4. Dynamic system investigation process for the inverted pendulum project.

Using the Mechatronics Kit to Complete Lab Assignments

The first few lab assignments in *Mechatronics* introduce the Rensselaer Mechatronics Kit. For example, in an early lab the students use Simulink to program the Arduino micro-processor to make an LED blink. Once the students are comfortable working with the hardware, they begin tackling more complex problems, including motor control and later, the inverted pendulum robot.

For the robot project, I ask the students to design a controller that uses the sensors and motor in the kit to balance the Arduino board on two wheels. To help them appreciate the complexity of this challenge, I first let them come up with a controller design on their own. I then lead them through a dynamic system investigation process with MATLAB and Simulink (Figure 4).

In this process, the students first describe the physical system and a set of simplifying assumptions. They then develop an electrical schematic and a free body diagram of the system's electrical and mechanical components. Applying Kirchoff's voltage law and Newton's laws, the students then generate a mathematical model of the system comprising two coupled nonlinear differential equations.

Since the mathematical model alone does not show the students how the system behaves, I have them analyze the differential equations in MATLAB, solving for the step response and the free response. They run experiments using the actual hardware, and compare the results with results produced by the mathematical model. If the simulated response doesn't match the experimental response, the students either adjust system parameters in the model or re-evaluate their assumptions—perhaps by incorporating friction or backlash effects into the model.

The next step in the process is to develop a controller. The students begin by linearizing the nonlinear equations of motion to obtain a linear state-space model. They then develop a linear quadratic regulator (LQR) full-state feedback controller model in Simulink

the theory. My view, however, is that students need to see that real life doesn't always match the theory.

I decided to revamp the lab curriculum with a focus on DC motor control. DC motors are ideal for mechatronics labs because they are inexpensive, are widely used in industry, and neatly couple electrical and mechanical systems. Low-cost motors and microprocessors were both readily available, but I needed to enable students to use them in labs without spending most of their time coding and debugging.

When I learned how easy it was to program an Arduino with MATLAB and

Simulink, I knew I had a path forward: I simply connected the Arduino hardware to my laptop with a USB cable, and after a few clicks, I was running a Simulink model on the hardware. At this point, all I lacked was a way for students to build Simulink models that could access the various sensors and actuators needed to complete the labs. To address this, I developed Rensselaer Arduino Support Package Library (RASPLib), which contains custom Simulink blocks for all the sensors and actuators in the Rensselaer Mechatronics Kit, including a three-axis accelerometer, gyroscope, and magnetometer, as well as a DC motor and amplifier (Figure 3).

MATLAB and Simulink at RPI

Most students enter my fourth-year mechatronics course already familiar with MATLAB and Simulink because the tools are used throughout the RPI engineering curriculum. For example, engineering students are required to take an introductory programming course that is offered in both C and MATLAB. RPI also offers *Introduction to Mechatronics Hardware and Software*, in which first- and second-year students use the Rensselaer Mechatronics Kit with MATLAB and Simulink to complete lab assignments similar to but simpler than those in my fourth-year course. I teach other fourth-year courses: *Design Optimization: Theory and Practice*, and *Machine Dynamics*, which rely heavily on

MATLAB, and *Multidisciplinary Capstone Design*, in which many students use MATLAB and Simulink to solve design problems spanning multiple engineering domains. Next semester, we plan to incorporate MATLAB, Simulink, and the Rensselaer Mechatronics Kit into *Modeling and Control of Dynamic Systems*, a core course required for all mechanical, aerospace, and nuclear engineers at RPI.

Because RPI has a Total Academic Headcount license, my colleagues and I can integrate assignments to be completed in MATLAB or Simulink into any course, knowing that all students have easy access to the tools.

using standard Simulink blocks as well as gyroscope, encoder, and motor blocks from the RASPLib library (Figure 5).

After running simulations in Simulink with the mathematical plant model, the students use Simulink external mode to run their controller model on the Arduino hardware. In this mode they can interact with their controller design as it is running, which makes debugging and tuning the controller easy. When they are ready to conduct final tests, the students deploy their Simulink model to the Arduino hardware model, enabling the robot to operate independently.

Frustration Gives Way to Inspiration

Early in the *Mechatronics* course, I tell the students that they will not be learning many new concepts. Instead, they'll be recombining much of what they learned about hardware, software, and control theory in their

earlier coursework. Nevertheless, most students find aspects of the course—including learning to model and mitigate real-world nonlinear system effects—very challenging. Inevitably, they are frustrated when they can't get their robots to balance, but when they do finally make the robot work, their eyes light up and the frustration melts away. As my colleague Johan Löfberg commented, “Labs are done, students very happy. ... One student told me that when she saw the robot on the first lecture, she could not imagine that she would be able to balance it, but then on the final lab she realized she had built most of the controller, and here it was, standing up.”

The growing popularity of the *Mechatronics* course is one indication that my students recognize the value in this approach and that I've managed to inspire at least some of them, just as I was inspired as an under-

graduate. The course now has a waiting list, and instead of offering only one session in the fall, we will soon be offering two sessions every semester. ■

LEARN MORE

The Miniature Balancing Robot Education Kit
minseg.com

Rensselaer Arduino Support Package Library (RASPLib)
mathworks.com/fx-62702

Arduino Support from Simulink
mathworks.com/arduino-simulink

Top 5 MATLAB and Simulink Arduino Projects
mathworks.com/arduino-projects

HS Bochum Students Design and Build a Motor Controller for an E-Longboard with Model-Based Design
mathworks.com/hs-bochum

MATLAB Licensing for Campus-Wide Use
mathworks.com/matlab-campus

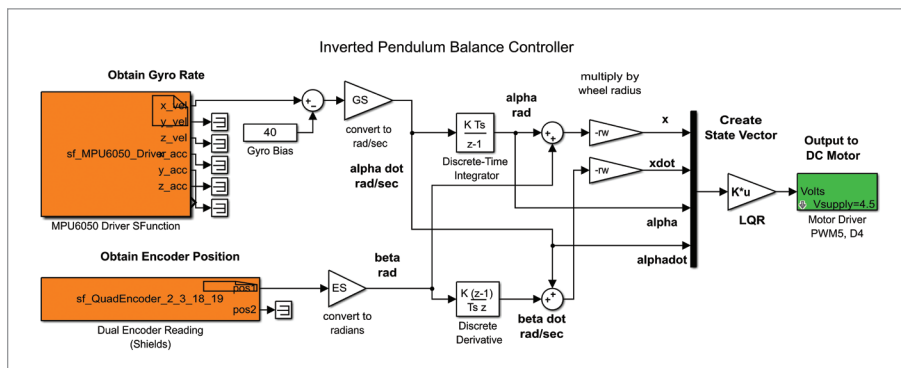


FIGURE 5. Simulink model of an inverted pendulum balance controller.



LIGO/Axel Mellinger

Confirming the First-Ever Detection of Gravitational Waves by Analyzing Laser Interferometer Data

By Matthew Evans, MIT



At about 7 a.m. on September 14, 2015, I received an email from my colleagues in Europe notifying me of an extraordinary event: Two detectors in the Laser Interferometer Gravitational-Wave Observatory (LIGO) had simultaneously identified what appeared to be a transient gravitational-wave signal—a “ripple” in space-time.

As a member of the team that developed the LIGO instrumentation, I was excited but also slightly apprehensive. I was excited because if the signal was authentic, it would mark the first time that a gravitational wave had ever been directly observed, confirming Albert Einstein’s prediction of their existence, made a century ago. It would also mark the first observation of a pair of black holes merging to form a single black hole. My colleagues and I were apprehensive, however, because we did not yet know whether the signal came from a genuine gravitational wave or was merely the result of some error in the LIGO

control system and instrumentation.

I immediately downloaded the LIGO data onto my laptop, opened MATLAB®, and began analyzing the recorded signals and visualizing the data. In the months that followed, my LIGO colleagues and I confirmed that we had, in fact, detected a gravitational wave, and identified its source: the cataclysmic collision of two black holes with a combined mass 60 times greater than our sun in a galaxy more than one billion light years away (Figure 1).

MATLAB is the language used by virtually every team in the world that designs gravitational wave detectors. It is the first tool I go

to when I want to design an instrument or to look at the data coming out of an instrument. For LIGO, we used it to analyze the fundamental noises that limit gravitational wave detector performance, calculate the optical response of our interferometers, and verify the entire control chain to ensure the validity of the results we observed.

Building the World’s Most Accurate Instrument for Measuring Displacement

During the black hole collision that we observed, over 1030 kilograms of mass was



FIGURE 1. Computer simulation of the collision of two black holes. Image courtesy SXS, the Simulating eXtreme Spacetimes (SXS) project.

converted into gravitational wave energy, most of it emitted in less than a second. The gravitational wave power radiated by this event was more than 10 times greater than the combined light power of every star and galaxy in the observable universe. To detect it, however, we had to design and build the most sensitive instruments ever created for measuring displacement: two laser interferometers capable of measuring changes in distance on the order of 10^{-18} meters.

LIGO is the world's largest gravitational wave observatory. Its two interferometers are located more than 3000 kilometers apart, one in Louisiana and the other in Washington. Interferometers like those in LIGO have two arms oriented at right angles to each other. At the LIGO sites, each arm is 4 km long. We shine a laser beam through a beam splitter, directing the two output beams to the ends of the two arms, where they are reflected back by

precisely oriented mirrors (Figure 2). In steady state, the beams, having traveled the same distance through each of the arms, meet back at a terminal photodetector in phase with one another. When a gravitational wave passes through this structure, it distorts the arms of the interferometer, briefly shortening the first and lengthening the second and then reversing this change. The difference in length between the two arms is minuscule—about 1/1000th of the diameter of a proton—but it is enough to affect the time the two beams take to travel through the arms. As a result, they arrive at the photodetector out of phase (at different times from one another), producing an interference pattern that we can measure and observe.

Since gravitational waves travel at the speed of light, if one passed by Earth, it would produce similar interference patterns at the Louisiana and Washington detectors within about 10 milliseconds (the time it takes light

to travel between the two locations). That is exactly what our systems detected on the morning of September 14.

Noise Analysis and Optical Modeling Tools for Interferometer Design

Numerous noise sources on Earth are capable of causing tiny length changes in the arms of LIGO. To investigate the fundamental noises associated with a gravitational wave detector design, we use a Gravitational Wave Interferometer Noise Calculator (GWINC). Developed entirely in MATLAB, this tool is used by physicists around the world to calculate the seismic, thermal, quantum, and other noises that limit gravitational wave detector performance (Figure 3).

The design and continued enhancement of the LIGO interferometers depended on a thorough understanding of noise effects. It also depended on understanding how the optics that make up the system—including beam splitters, lenses, and mirrors—work together in the complete system. To help researchers with this aspect of the interferometer design, I developed Optickle, a MATLAB based tool for frequency-domain modeling and simulation of interferometers. Optickle generates transfer functions for elements of the optical plant, enabling physicists to calculate and visualize the opto-mechanical frequency response of virtually any interferometer, including those in LIGO (Figure 4).

Updating Filters on the Fly

LIGO has thousands of digital filter modules for noise suppression and signal processing. One advantage of the current LIGO setup over its predecessors is the ability to update these filters on the fly. In the past, we had to power down and then restart the system, a process that took up to an hour. Today, we can change the infinite impulse response (IIR) filter coefficients in the modules in about a minute without shutting the system down.

This capability is made possible by a model of the interferometer real-time control system that we created in MATLAB and Simulink®.

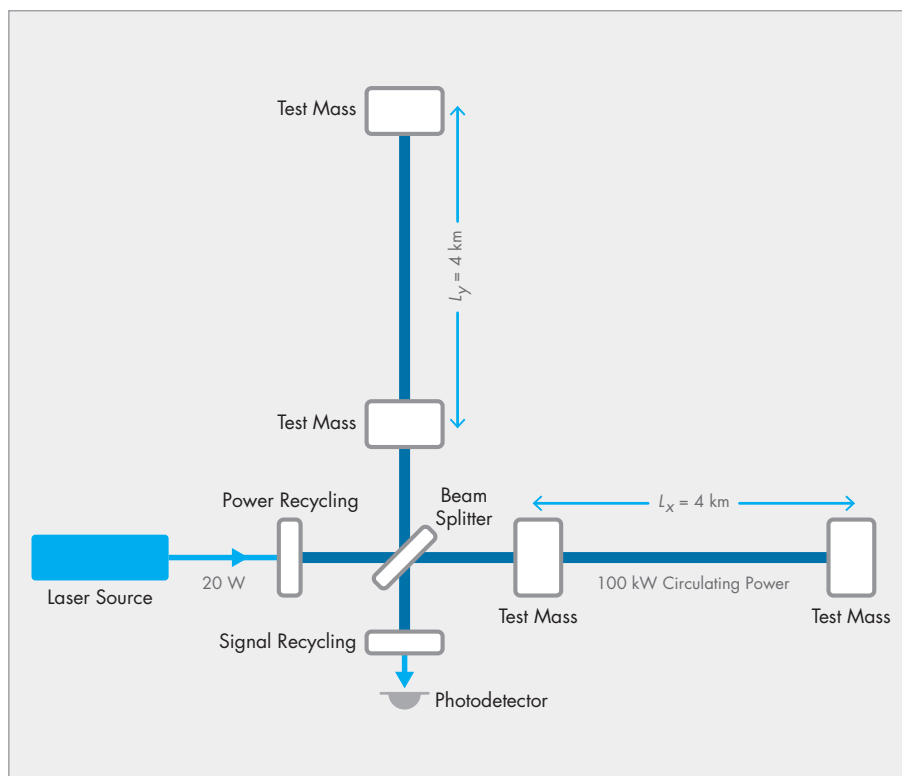


FIGURE 2. Diagram of a simplified LIGO interferometer. Image courtesy B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), "Observation of Gravitational Waves from a Binary Black Hole Merger," *Phys. Rev. Lett.* 116, 061102 – Published 11 February 2016.

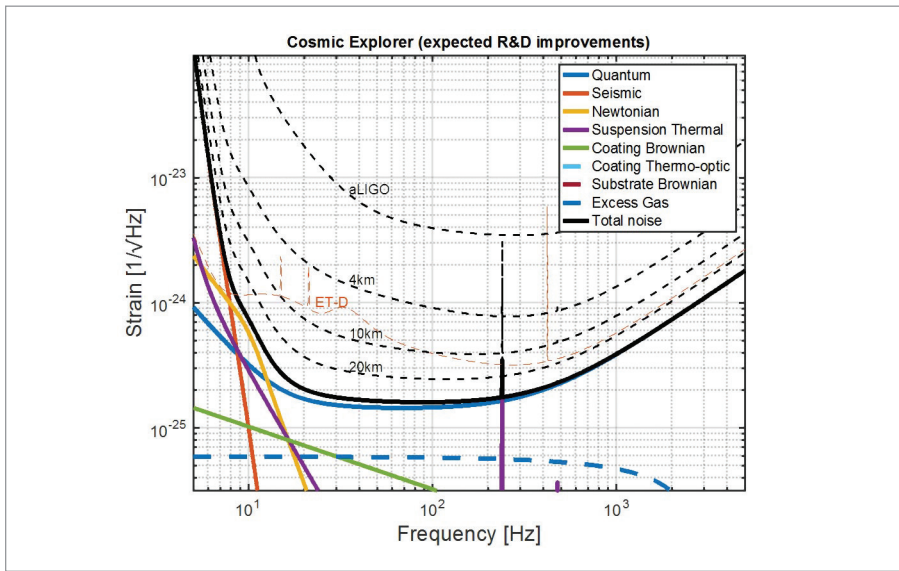


FIGURE 3. GWINC plot showing the dominant fundamental noise sources that limit detector performance.

For example, to suppress motion of the mirrors below 5 Hz, caused by microseismic peaks from ocean waves hitting the shore, we needed to change the filters to add more gain in the control loop. In MATLAB we analyzed the feedback loop of the control system and examined the phase margin of that loop. After plotting the necessary transfer functions and determining how much gain we could add, we generated new filter coefficients and loaded them into the filter modules—all while the interferometer was running.

Verifying the First Detection and Anticipating the Next

After the initial detection, I spent days in MATLAB analyzing all the LIGO data from the event to make sure that the signal wasn't caused by a bug in the system. To do this I reproduced the entire control chain and gravitational wave signal path in MATLAB, including all the filter parameters in use at the time, along with measurements of the detected signal as it moved through the system. This analysis gave us confidence that the

signals captured by both interferometers were from a genuine gravitational wave.

About three months after the first event, on Christmas Eve, a second gravitational wave was detected. This one was smaller, the result of a merger of two black holes that were 8 and 14 times greater than the Sun's mass. The second detection provided further evidence that the first had been no fluke, and it confirmed that we have entered a new era of gravitational wave astronomy.

Gravitational waves give us an entirely new perspective on our universe, enabling us to study events previously invisible to us. I look forward to exploring the data from each new detection in MATLAB and to using the tools we've developed to further improve gravitational wave detector instrumentation. ■

LEARN MORE

LIGO
ligo.org

Dark Energy Gravitational Waves
blogs.mathworks.com/cleve/?p=1367

Accelerate the Design and Prototyping of Signal Processing Algorithms (25:14)
mathworks.com/video-119981

Designing Radio Astronomy Instruments in South Africa and Worldwide with Simulink
mathworks.com/astronomy

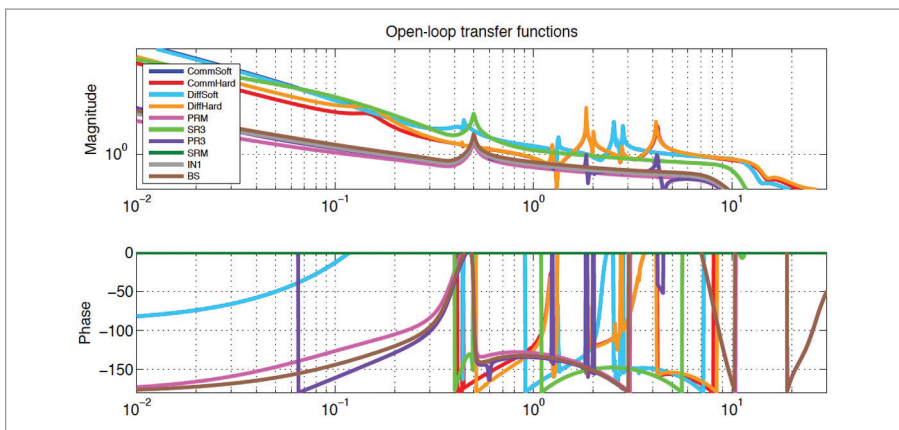
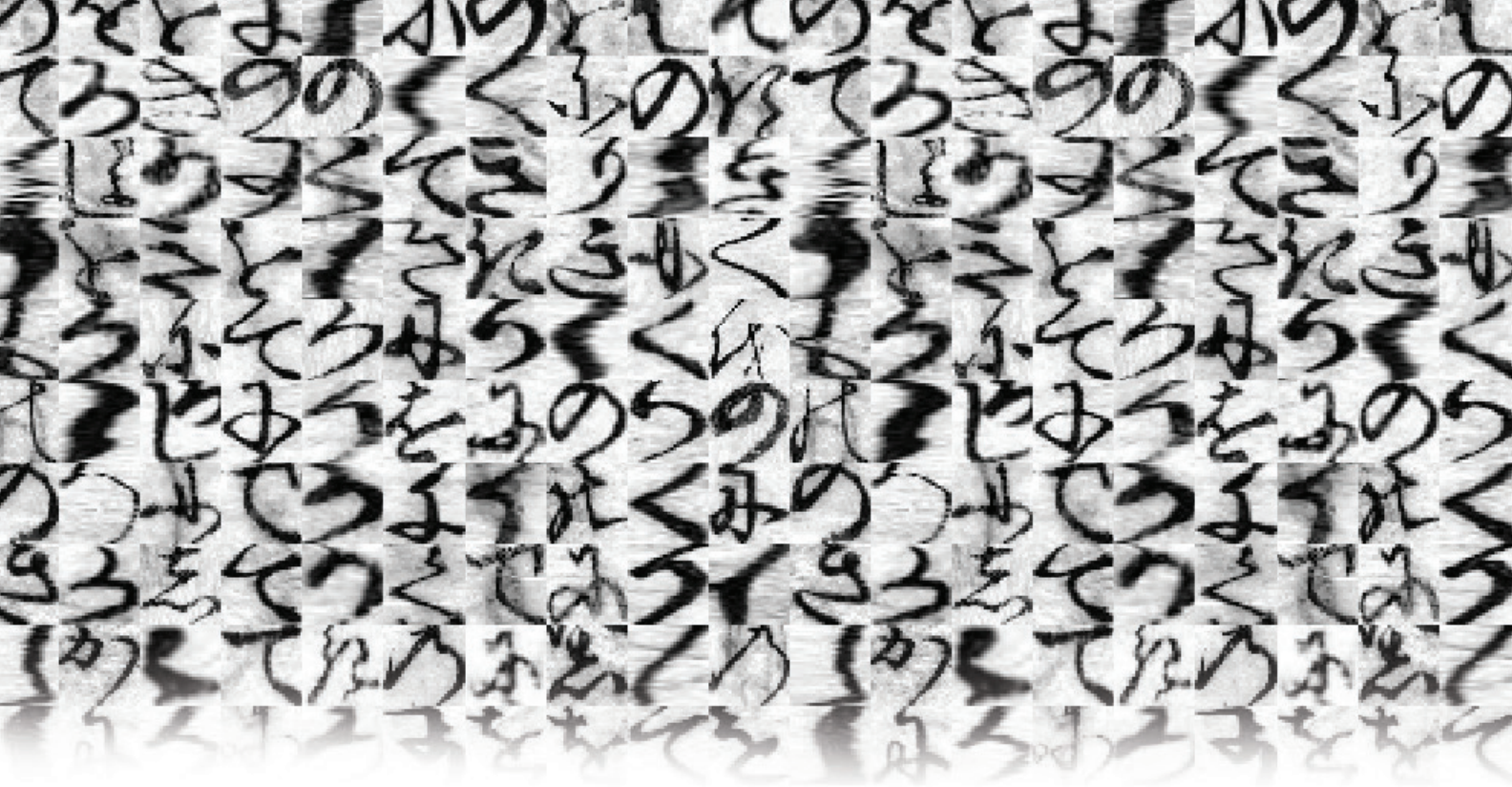


FIGURE 4. Open-loop transfer functions generated by Optickle (a MATLAB based tool for frequency-domain modeling).



Classifying Handwritten Japanese Characters with Deep Learning

The image shows 100 characters from Japanese manuscripts written in the Edo period (1603–1868). At a first glance, these characters just look like scribbles. Perhaps if they were in sentences, it would be possible to identify them through context. But could a deep learning network identify the characters purely by themselves? MathWorks consultant Akira Agata trained a convolutional neural network (CNN) to do just that. Akira tested the CNN against a character set it hadn't encountered before. His network achieved over 90% accuracy, with just a few lines of code.

Find out how Akira did this: blogs.mathworks.com/pick/?p=8671

LEARN MORE

Japanese Classics Character Dataset
codh.rois.ac.jp/char-shape

Deep Learning: Three Things You Need to Know
mathworks.com/deep-learning

*Japanese classics character data set (Kokugaku Kenkyu other collection/CODH processing)
Provided by: Humanities Open Data Shared Use Center*

Fitting and Extrapolating U.S. Census Data

By Cleve Moler, MathWorks

"Growth of U.S. Population Is at Slowest Pace Since 1937." This *New York Times* headline prompted me to revisit an old chestnut: fitting and extrapolating census data. In the process, I have added a couple of nonlinear fits, namely, the logistic curve and the double exponential Gompertz model.

The experiment is older than MATLAB®. It started as an exercise in *Computer Methods for Mathematical Computations*, by Forsythe, Malcolm, and Moler, published 40 years ago. We were using Fortran back then. The data set has been updated every ten years since the book was published. Today, MATLAB graphics make it easier to vary the parameters and see the results, but the underlying mathematical principle remains unchanged:

* Using polynomials of even modest degree to predict the future by extrapolating data is a risky business.

The famous New York Yankees catcher and budding computational scientist Yogi Berra once said, "It's tough to make predictions, especially about the future." We're going to use polynomials, splines, exponentials, and the census app in Cleve's Laboratory to do just that.

The Data

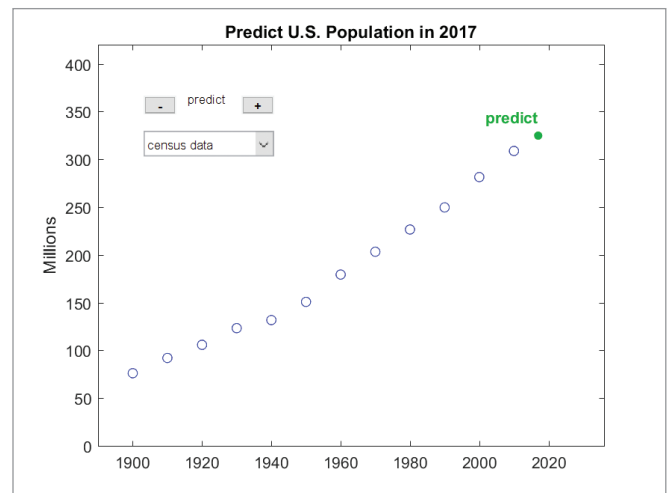
The data for our experiment comes from the decennial census of the U.S. for the years 1900 to 2010 (numbers are in millions):

1900	75.995
1910	91.972
1920	105.711
1930	123.203
1940	131.669
1950	150.697
1960	179.323
1970	203.212
1980	226.505
1990	249.633
2000	281.422
2010	308.746

The task is to extrapolate the size of the population beyond 2010. Using the [censusapp](#), let's see how an extrapolation of just seven years, from 2010 to 2017, matches the Census Bureau model. Before you read any further, pause and make your own guess.

The Census App

Here's the opening screen of the [censusapp](#). The plus and minus buttons change the extrapolation year in the title. If you go beyond 2030, the plot zooms out.



Twelve decades of census data. Let's extrapolate seven more years.

The app's pull-down menu offers seven models:

- census data
- polynomial
- pchip
- spline
- exponential
- logistic
- gompertz

Forty years ago, we only had polynomials.

The Target Value

The United States Census Bureau website provides a dynamic population clock that operates continuously. Here is the snapshot taken at noon, EDT, on Census Day, April 1, 2017. This is the designated time to capture the census value for the year.

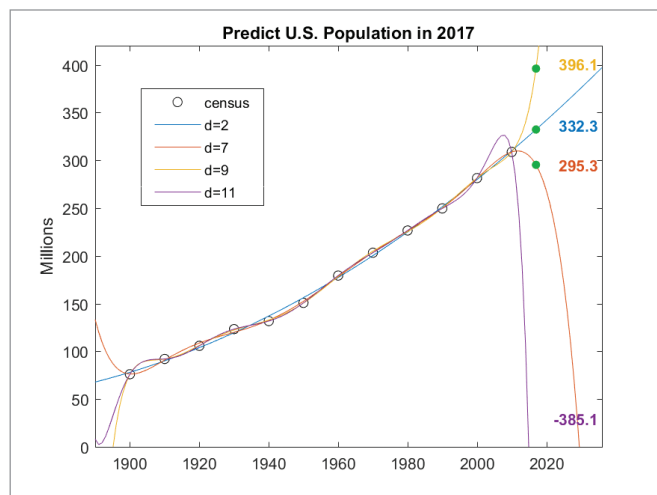


The population clock on April 1, 2017.

So, the target value for extrapolating the census data to 2017 is 324.790 million.

Using Polynomials

Polynomials like to wiggle. Constrained to match data within a particular interval, they go crazy outside that interval. In this experiment, there are 12 data points. The `censusapp` lets you vary the polynomial degree between 0 and 11. Polynomials with degree less than 11 approximate the data in a least squares sense. The polynomial of degree 11 interpolates the data exactly. As the degree increases, the approximation of the data becomes more accurate, but the behavior beyond 2010 (or before 1900) becomes more violent. Here are degrees 2 and 7, 9, 11, superimposed on one plot.



Polynomials of degree greater than two are not suitable for extrapolation.

The quadratic fit is the best behaved. When evaluated at year 2017, it misses the target by 7.5 million, but it fails to predict the decreasing rate of growth observed by the Census Bureau. (Of course, there

is no reason to believe that the U.S. population grows in time like a second-degree polynomial.)

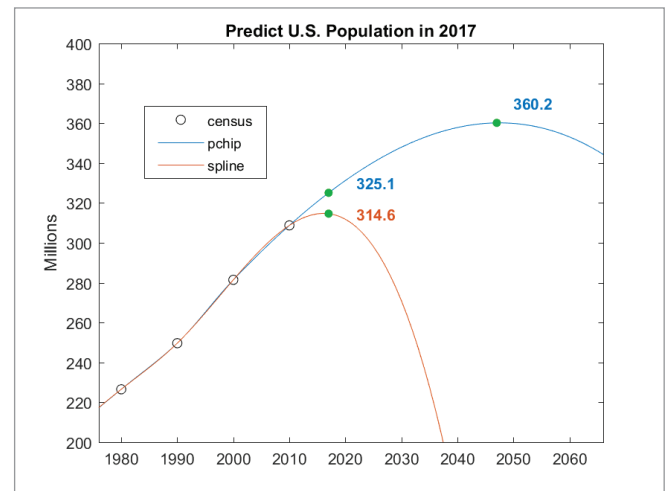
The interpolating polynomial of degree 11 tries to escape even before it gets to 2010, and it goes negative late in 2014.

Extrapolating with Splines

MATLAB has two piecewise cubic interpolating polynomials: `spline` and `pchip`. The classic `spline` is smooth because it has two continuous derivatives. Its competitor, `pchip`, sacrifices a continuous second derivative to preserve shape and avoid overshoots. Neither of these polynomials is intended for extrapolation, but we will use them anyway.

Their behavior beyond the interval is determined by their end conditions. The classic `spline` uses the so-called *not-a-knot* condition. It is actually a single cubic in the last two subintervals. That cubic is also used for extrapolation beyond the endpoint. `pchip` uses just the last three data points to create a different shape-preserving cubic for use in the last subinterval and beyond.

Let's zoom in on `spline` and `pchip`.



`pchip` happens to produce a reasonable extrapolation; `spline` does not.

Both are predicting a decreasing rate of growth beyond 2010, just like the Census Bureau. But `spline` is painting a gloomy picture. The value of 314.6 million for 2017 is 10 million below the population clock value, and is near the maximum. On the other hand, `pchip` gets lucky, and its 2017 value of 325.1 million comes within 0.3 million of the population clock value. Looking into the future, `pchip` reaches a maximum of 360.2 million in 2047. That's a prediction to ponder.

Three Exponentials

As I said, there is no good reason to model population growth by a polynomial, piecewise or not. But because we can expect a growth rate proportional to the size of the population, there is good reason to use an exponential.

$$p(t) \approx \alpha e^{bt}$$

Many authors have proposed ways to modify this model to avoid its unbounded growth. I have added two of these to the [censusapp](#). One is the logistic model.

$$p(t) \approx \alpha / (1 + b e^{-ct})$$

The other is the Gompertz double exponential model, named after Benjamin Gompertz, a 19th-century, self-educated British mathematician and astronomer.

$$p(t) \approx \alpha e^{-b} e^{-ct}$$

In both models the growth is limited because the approximating term approaches α as t approaches infinity.

In all three of the exponential models, the parameters α , b , and possibly c , appear nonlinearly. In principle, I could use `lsqcurvefit` to search in two or three dimensions for a least squares fit to the census data. But I have an alternative: By taking one or two logarithms, I obtain a *separable least squares* model where, at most, one parameter, α , appears nonlinearly.

For the exponential model, take one logarithm.

$$\log p \approx \log \alpha + bt$$

Fit the logarithm of the data by a straight line and then exponentiate the result. No search is required.

For the logistic model, take one logarithm.

$$\log(\alpha/p - 1) \approx \log b - ct$$

For any value of α , the parameters $\log b$ and c appear linearly and can be found without a search. So, use a one-dimensional minimizer to search for α . I could use `fminbnd` or its textbook version, `fminbx`, from *Numerical Methods with MATLAB*.

For the Gompertz model, take two logarithms.

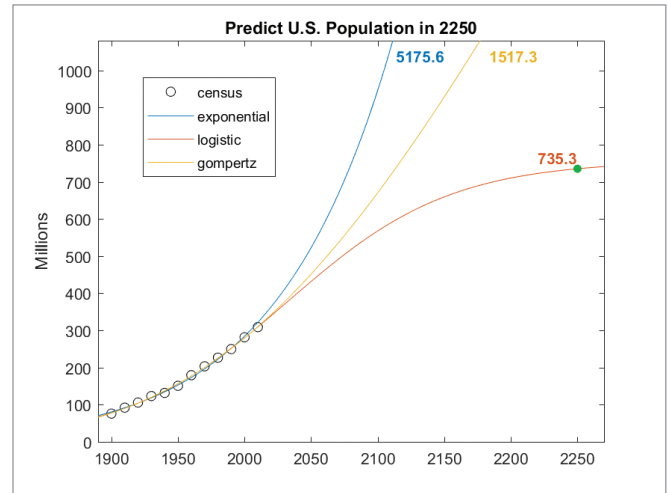
$$\log \log \alpha/p \approx \log b - ct$$

Again, do a one-dimensional search for the minimizing α , solving for $\log b$ and c at each step.

I should point out that taking logarithms changes the fit criterion. I am doing a least squares fit to the log of the data, not to the actual data.

Results

Here are the three resulting fits, extrapolated over more than 200 years to the year 2250.



Exponential models extrapolating over 200 years.

The pure exponential model reaches 5 billion by that time, and is growing ever faster. I think that's unreasonable.

The value of a in the Gompertz fit turns out to be 4309.6, so the population will be capped at 4.3 billion. But it has only reached 1.5 billion two hundred years from now. Again, unlikely.

The value of α in the logistic fit turns out to be 756.4, so the predicted U.S. population will slightly more than double over the next two hundred years. Despite the Census Bureau's observation that our rate of growth has slowed, we are not yet even halfway to our ultimate population limit.

I'll let you be the judge of that prediction. ■

LEARN MORE

Cleve's Laboratory
mathworks.com/cleves-lab

Cleve's Corner Blog: Splines and Pchips
blogs.mathworks.com/cleve/?p=196

U.S. Census Bureau Population Clock
census.gov/popclock



Robot Prints 3D Building in Half a Day

Researchers from MIT's Mediated Matter Group designed a robotic platform that can 3D-print architecture. The Digital Construction Platform (DCP) consists of a large industrial robot arm with a smaller KUKA robot attached to it. The large arm gives the robot its reach. The KUKA robot provides precise control and can be fitted with a variety of tools such as foam and concrete sprayers.

DCP recently printed a building. The entire 167-square-meter structure—the largest ever printed by a mobile robot—was created in just 13.5 hours.

Find out how it was done: blogs.mathworks.com/headlines/?p=937

LEARN MORE

Identifying and Simulating a Dynamic Model
of the KUKA LWR4+ Robot
blogs.mathworks.com/kuka

Blog: MATLAB and Simulink Behind the Headlines
blogs.mathworks.com/headlines

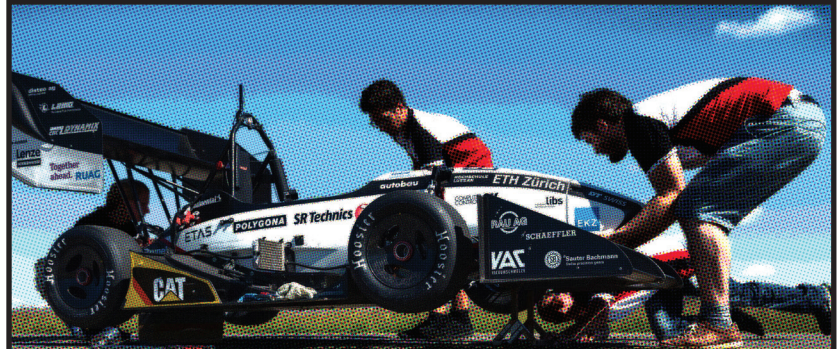
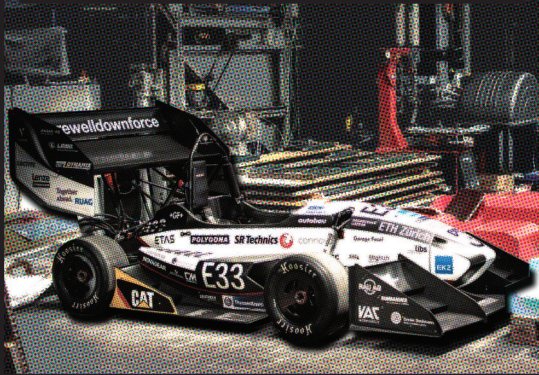
Image Credit: Steven Keating, Julian Leland, Lev Cai, and Neri Oxman/Mediated Matter Group, MIT

grimsel Sets Electric Vehicle Acceleration Record

Formula Student is the world's biggest competition for student engineers, with over 600 teams competing in events including vehicle design, endurance, and acceleration. When the Academic Motorsports Club Zurich (AMZ) entered Formula Student last year, they decided to attempt the acceleration record as well. AMZ team leader Leiv Andresen tells their story.

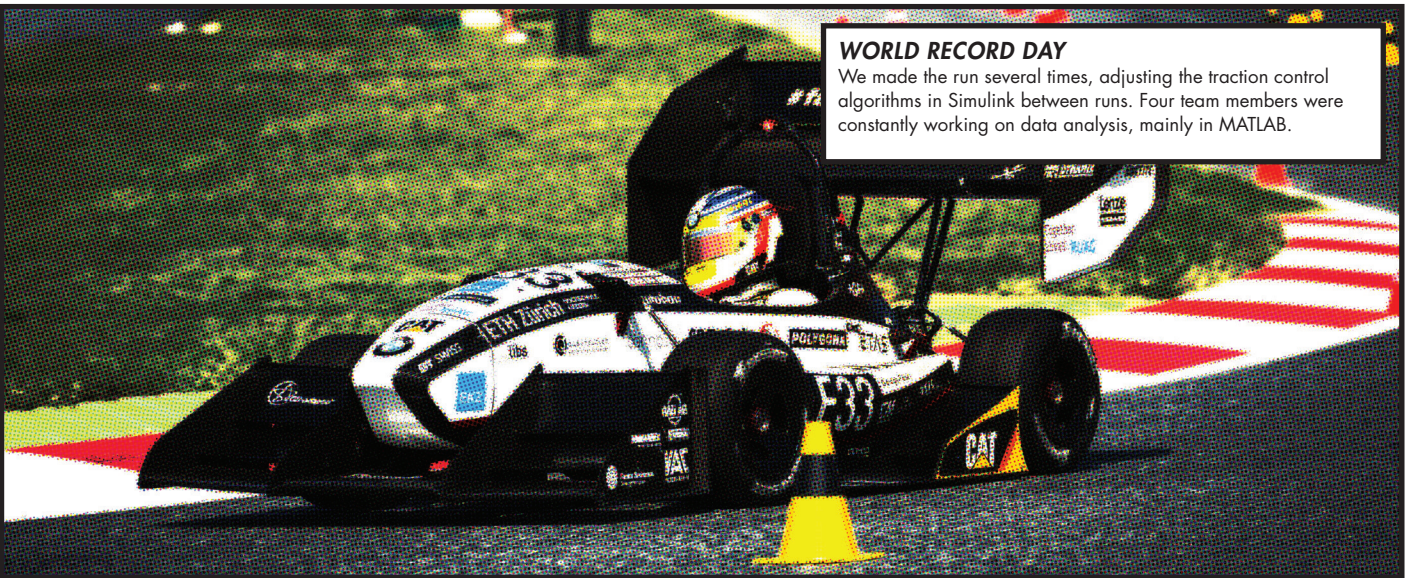
MEET grimsel

grimsel features a carbon fiber chassis, 4 electrical wheelhub motors yielding a peak power of 200 HP, and individual traction control on each wheel.



PREPPING grimsel FOR THE TRACK

To save weight, we removed parts of the cooling system and the rear wing. Simulink® simulations helped us decide which components to keep. We analyzed lots of data in MATLAB® to validate our setup changes.



WORLD RECORD DAY

We made the run several times, adjusting the traction control algorithms in Simulink between runs. Four team members were constantly working on data analysis, mainly in MATLAB.

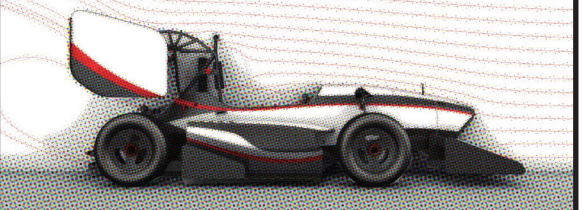
MISSION ACCOMPLISHED!

0–60 miles per hour in 1.513 seconds! Once we'd grasped the significance of that result, the awesome feeling that we'd just broken the world record set in.



BUT THE STORY DOESN'T END HERE

Since that record-breaking run we've continued to compete in races all across Europe. ... And meet flüela, our first fully autonomous racecar.



Watch grimsel (3:40): youtube.com/watch?v=I-NCH8ct24U

A robot that sees, acts, and learns, programmed in an afternoon. That's Model-Based Design.



To create an advanced humanoid robot that can perceive, throw, and catch a ball, engineers at DLR used Model-Based Design with MATLAB and Simulink.

Result: The team could integrate control and vision for catching, and optimize the throwing trajectory, generate embedded software, and verify it worked—in one afternoon.

Discover Model-Based Design with MATLAB and Simulink at mathworks.com/mbd