Dynamic Data-Driven Application Systems for Empty Houses, Contaminat Tracking, and Wildland Fireline Prediction

Craig C. Douglas[1,2], Divya Bansal[1], Jonathan D. Beezley[3], Lynn S. Bennethum[3], Soham Chakraborty[1], Janice L. Coen[4], Yalchin Efendiev[5], Richard E. Ewing[5], Jay Hatcher[1], Mohamed Iskandarani[6], Christopher R. Johnson[7], Deng Li[1], Minjeong Kim[3], Robert A. Lodder[8], Jan Mandel[3], Guan Qin[5], and Anthony Vodacek[9]

1 University of Kentucky, Department of Computer Science, 773 Anderson Hall, Lexington, KY 40506-0046, USA.

2 Yale University, Department of Computer Science, P.O. Box 208285, New Haven, CT 06520-8285, USA.

3 University of Colorado at Denver and Health Sciences Center, Department of Mathematical Sciences, P.O. Box 173364, Denver, CO 80217-3364, USA.

4 National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307-3000, USA.

5 Texas A&M University, Institute for Scientific Computation, 612 Blocker, 3404 TAMU, College Station, TX, 77843-3404, USA.

6 University of Miami, Rosenstiel School, of Marine and Atmospheric Science, 4600 Rickenbacker Causeway, Miami, FL 33149-1098, USA.

7 University of Utah, Scientific Computing and Imaging Institute, Salt Lake City, UT 84112, USA.

8 University of Kentucky, Department of Chemistry, Lexington, KY, 40506 USA.

9 Rochester Institute of Technology, Center for Imaging Science, Rochester, NY 14623 USA.

Abstract. We describe three different dynamic data-driven applications systems (DDDAS): an empty house, a contaminant identification and tracking, and a wildland fire. Each has something in common with all of the rest and can use some common tools. Each DDDAS is quite complicated in comparison to a traditional static input simulation that is run with large numbers of inputs instead of one longer run that is self-correcting.

1 Introduction

We quote from the 2005 dynamic data-driven application systems (DDDAS) National Science Foundation solicitation [1], "DDDAS is a paradigm whereby application (or simulations) and measurements become a symbiotic feedback control system. DDDAS entails the ability to dynamically incorporate additional data into an executing application, and in reverse, the ability of an application to dynamically steer the measurement process. Such capabilities promise more accurate analysis and prediction, more precise controls, and more reliable outcomes. The ability of an application to control and guide the measurement process and determine when, where, and how it is best to gather additional data has itself the potential of enabling more effective measurement methodologies. Furthermore, the incorporation of dynamic inputs into an executing application invokes new system modalities and helps create application software systems that can more accurately describe real world, complex systems. This enables the development of applications that intelligently adapt to evolving conditions and that infer new knowledge in ways that are not predetermined by the initialization parameters and initial static data. The need for such dynamic applications is already emerging in business, engineering and scientific processes, analysis, and design. Manufacturing process controls, resource management, weather and climate prediction, traffic management, systems engineering, civil engineering, geological exploration, social and behavioral modeling, cognitive measurement, and bio-sensing are examples of areas likely to benefit from DDDAS." See also [2] for numerous examples and clear definitions of what makes a system a DDDAS.

As small groups, we are working on three kinds of DDDAS critical infrastructure projects funded by the NSF:

- *ITR/NGS: Collaborative Research: DDDAS: Data Dynamic Simulation for Disaster Management*. The emphasis is on wildland fire modeling, simulation, prediction, and a major milestone is to provide real-time information to people fighting actual fires. The final test of the project will be to do a full scale test with a prescribed burn of a mountainside in 2008-2009.
- *ITR: Collaborative Research: Predictive Contaminant Tracking Using Dynamic Data Driven Application Simulation (DDDAS) Techniques*. Multiscale data-driven algorithms and software to easily move data from sensors to computers potentially far away has been developed.
- *DDDAS-TMRP: Collaborative Research: Adaptive Data-Driven Sensor Configuration, Modeling, and Deployment for Oil, Chemical, and Biological Contamination near Coastal Facilities*. Consider a networked drone operating off a coast that recognizes oil in water. Upon detection and alerting the simulation, by dynamically loading into the drone sensor a chemical library specific to hydrocarbon pollution, the sensor can search for chemicals that will identify the source of the hydrocarbons. For example, a diesel-driven ship may have sunk nearby, or a fishing boat may simply be

leaking fuel. 100LL would indicate a small downed aircraft. Depending on the sensor result, very different computations can be done: trace where the ship or aircraft sank and alert rescue, or trace where the boat sailed and what its travel route was to identify the boat and mitigate the problem.

The remainder of this paper is organized as follows. In §2, we describe features that are common in DDDAS. In §3, we describe a whimsical DDDAS that would make a good commercial product for the modern American home. In §4, we describe a contaminant tracking DDDAS based on a set of movable drones in water bodies. In §5, we describe a wildland fire DDDAS. Finally, in §6, we offer some concluding remarks.

2 What Is in a Typical DDDAS

DDDAS environments require new software capabilities for application modeling and composition, dynamic runtime, resource management, data management, and measurement control aspects, as well software architecture drilling across all layers and end-to-end software infrastructure. The DDDAS program solicitation includes a comprehensive list of challenges and has inspired the scientific community, as exemplified by DDDAS projects that have started to address these and other related challenges. In our own DDDAS projects, we have identified several relatively diverse areas that have common issues that must be addressed by DDDAS: computer science, informational, and computational sciences that lead to significant impact for addressing important problems. These include:

1. Effectively *assimilating* continuous streams of data into running simulations. These data streams most often will be…
   a. Noisy but with known statistics, and must be incorporated into the model using stochastic methods, such as filters and smoothers.
   b. Received from a large number of scattered remote locations and must therefore be injected into a usable computational grid.
   c. Missing bits or transmission packets, as for example is the case in wireless transmissions.
   d. Injecting dynamic and unexpected data input into the model.
   e. Limited to providing information only at specific scales, specific to each sensor type.
2. *Warm restarting* simulations by incorporation of the new data into parallel or distributed computations, which require the data but are sensitive to communication speeds and data quality.
3. *Tracking and steering* (control of measurements, models, reporting results, and visualization) of remote distributed simulations to efficiently interact with the computations and to collaborate with other researchers.
4. *Translation components* to rectify when simulation output does not directly match observational data.

5. *Interpretation and analysis components* to assist researchers with collections of simulations.
6. *Application program interface and lightweight middleware components* for designing and creating a DDDAS or DDDAS problem solving environment.
7. Better *scheduling of computational and network resources* so that multiple models, possibly running at different locations, can be coordinated and data can be exchanged in a timely manner.
8. *Virtualization* and *sandbox* implementations for testing purposes and security.

DDDAS assumes that application components, resource requirements, application mapping, interfaces and control of the measurement system can be modified during the course of the application simulation. The diagram in Figure 1 shows how a number of elements might dynamically interact with each other: Any of the components may change without resorting to a new simulation as the computation progresses. Many DDDAS applications are multiscale in nature. As the scale changes, models change, which in turn, changes which numerical algorithms must be used and possibly the discretization methods. DDDAS applications involve a complicated time dependent, nonlinear set of coupled partial differential equations, stochastic or agent-based simulation methods, which add to the complexity of dynamically changing models and numeric algorithms. It also causes computational requirements to change, particularly if dynamic adaptive grid refinement or coarsening methods are used, in response to the dynamically streamed data into the executing model.

To support data management needs in our DDDAS projects, data acquisition, data accessing, and data dissemination tools are typically used. Data acquisition tools are responsible for retrieving of the real-time or near real-time data, processing, and storing them into a common internal data store. Data accessing tools provide common data manipulation support, e.g., querying, storing, and searching, to upper level models. Data dissemination tools read data from the data store, format them based on requests from data consumers and deliver the formatted data to the data consumers. Figure 2 illustrates a simplified view of the software framework of the DDAS system we are developing. In our implementation, the data used to drive a DDDAS system are retrieved periodically by a data retrieval service, extracted, converted, quality controlled, and then staged as dynamic inputs to our simulation models. The extraction process reads the retrieved data based on the meta data associated with them and feeds the extracted values to the conversion model whose major purpose is unit conversion, e.g., from inches to millimeters. The converted data are then analyzed for potential errors and missing values by the quality control model. This control process will ensure the correctness of the data, which is of great importance for the model simulation accuracy. The quality controlled data are then fed to the data storage model, which either saves the data to a central file system or loads them to a central database (this depends on project requirements). The data store model may also need to register the data in a metadata database so that other models can query it later.

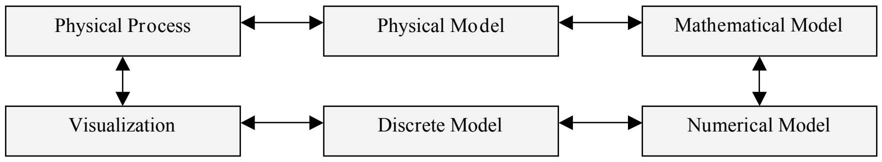


Fig. 1. DDDAS processing [3]

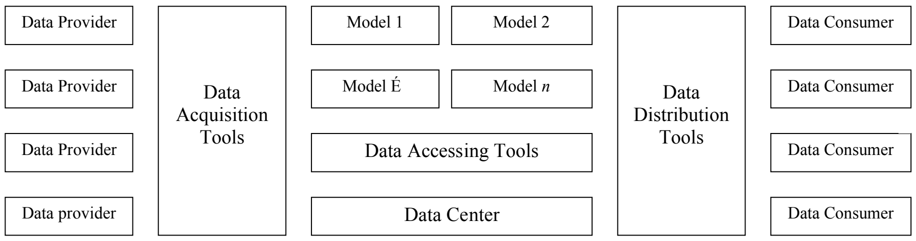


Fig. 2. Data acquisition, accessing, and dissemination software layout in a typical DDDAS project with *n* models

DDDAS research projects have brought together multidisciplinary expertise, involving researchers from a number of fields to synergistically pursue research on creating DDDAS capabilities and environments. There is a learning curve that is nontrivial. DDDAS applications are usually complicated, getting data is usually difficult, and there is already large scale research ongoing using traditional, take initial data and just run a simulation some period of time, and look at the results.

A community web site, http://www.dddas.org [4], has been developed by Prof. Douglas with help from about 50 other DDDAS-related projects. The site currently has a complete funded project list (from 2000 to 2006), virtual proceedings from workshops from 2000 through 2006 [5-8], a number of talks on topics that range from disaster management to transportation to homeland security to how a bat flies, news items, pointers to working DDDAS codes, and the January 2006 NSF DDDAS workshop report [9]. Most of the projects listed are from the United States, though a number of the projects have international partners and interest in DDDAS overseas has been increasing.

3 An Empty House DDDAS

In the United States, it is quite common for homes to be devoid of people for a significant number of hours per day (i.e., “My two cats really own the house.”). However, it is advantageous to have the home appear to be lived in and constantly occupied. A *smart home* is able to communicate with the owners and for sensors to be adjusted to the immediate situation, inside or outside.

We need sensors that can detect motion and identify individuals (pictorially and verbally). Depending on the first results from the DDDAS, the sensors will have to detect much more complicated data. Face recognition and some indication of the emotional state is one of the highly successful DDDAS projects [10].

We need to distinguish between animals, vehicles, regular visitors (wanted and unwanted), and irregular visitors. Each category requires its own computational model. Frequently, more than one model must be used in parallel. In fact, the way to implement an empty house DDDAS is two tiered: (1) object detection and identification, and (2) receiving information about a detected object (recognized friendly, recognized undesirable or unrecognized) and chooses an appropriate response. The two tiered approach reduces the load on the second tier so that resources are available for decision making and communicating with the owner and makes it easy to add extra (or new) models later.

Animals can usually be ignored unless individuals are regular nuisances. Ones that live in the house and need to be let in (or out) are a special case and must be recognized as well as the animal's intent. When the occupants are away for an extended period of time, the animals need to be fed and given water on a regular basis.

Vehicles on a driveway are the first point of identification of individual people. For example, recognizing the license plate or corporate identity (e.g., UPS) leads to running a model for acceptable visitors. A moving van might indicate house robbers and a call to the police might be warranted depending on what the people inside it do after getting out of the van. Smoke detection in or near the house obviously needs a call to the fire department.

People walking up to a door (or window) provides a different recognition problem. Examples of walkers include the mailman or other deliver people, product sellers, house robbers, arsonists, and religious nuts. The former is welcome, but the rest are unwelcome and/or a serious threat to the integrity of the house. Being able to identify unwanted visitors and determine which ones will go away with a polite, but firm, "No," is essential and nontrivial to model. A database has to be developed over time as the DDDAS is trained.

For deliveries, a voice greeting needs to be generated, answers to common polite questions (e.g., how are you?), signing for a box or envelope, and directions given to what to do with a delivery. If the delivery is put into a secure box, the contents need to be transferred into the house either immediately or on a regular basis. The house occupants need to be notified of a delivery and who delivered it (using pictures or audio). If a delivery person cannot be answered by the DDDAS, the occupants should have the option of seeing, hearing, and talking to the delivery person in real time. Hence the DDDAS needs to be able to track the occupants seamlessly. Thus, two way, secure communications is required.

Clearly the Household DDDAS is nontrivial, yet much of what is needed to produce a working one has been developed over the past few years in NSF supported DDDAS grants [4]. To make it work requires a set of sensors (motion detectors, microphones, and webcams), mechanical devices to move or rotate the sensors, face and vehicle recognition software, voice decoding, networking (at the house and with the occupants, wherever they might be), and parallel processing to run multiple models simultaneously. Yet the total cost of such a system is not very high thanks to most of the devices and software being either commonplace or already existing. It is a matter of assembling the pieces correctly and devising the DDDAS. This would make an interesting commercial product for installing in new homes where the cost would be dwarfed by construction and land expenses.

4 Contaminant Tracking DDDAS

The most infamous oil spill was the Exxon Valdez oil spill. It was the largest oil spill ever when it occurred, but is no longer ranks among the top 50 largest oil spills globally. Oil spills remain one of the largest threats to coastal water regions and water supplies. Yet even small oil spills can indicate different things, as noted in the DDDAS-TRMP project summary in §1.

The DDDAS contaminant tracking system consists of sensors, a hydrodynamic and contaminant transport models, a data assimilation system, as well as computers, networks, and software to integrate the capabilities of the various components into a unified system for disaster management and mitigation.

Our sensor is a Solid-State Spectral Imager (SSSI) designed to gather hydrological and geological data and then to perform chemical analyses. The sensor is small and light enough to be mounted on various roving platforms so it can be used in remote-sensing situations and can scan ranges of 10-100 meters in distance. Using a laser-diode array, photodetectors, and on-board processing, the SSSI combines spectroscopic integrated sensing and processing with a hyperspace data analysis algorithm.

The SSSI detects and identifies contaminants in water using near-infrared (IR), visible, and ultraviolet light. Absorption, fluorescence, and even Raman spectrometry can be implemented, but absorption spectrometry is the most common. Virtually every organic compound (e.g., polycyclic aromatic hydrocarbons, paraffins, carboxylic acids, and sulfonic acids) has a near-IR spectrum that can be measured, including two classes of terrestrial biomarkers, lipids, and amino acids. Near-infrared spectra consist of overtones and combinations of fundamental mid-infrared bands, giving near-infrared spectra a powerful ability to identify organic compounds while still permitting some penetration of light into samples.

The SSSI has a modest amount of memory and computing capacity on board. The SSSI is reprogrammable in the field. When an interesting chemical trace is discovered, the reaction from the application overseeing the SSSI is two-fold: (a) invoke an appropriate application, and (b) request that the SSSI look for specific other chemical traces using other specific pulse sequences. There is a symbiotic relationship between the sensor network and the application simulation that is typical in a DDDAS.

The SSSI uses Walsh-Hadamard or Complementary Randomized Integrated Sensing and Processing (CRISP) encoding sequences of light pulses to further increase the signal-to-noise (S/N) ratio. In a Walsh-Hadamard sequence multiple laser diodes illuminate the target at the same time, increasing the number of photons received at the photodetector and the S/N. The Walsh-Hadamard sequence can be demultiplexed to individual wavelength responses with a matrix-vector multiply operation. CRISP encoding uses orthogonal pseudorandom codes with unequal numbers of on and off states. The duty cycle of each code is different and the codes are selected to deliver the highest duty cycles at the wavelengths where the most light is needed and lowest duty cycle where the least light is needed to make the sum of all of the transmitted (or reflected) light from the samples proportional to the analyte concentration of interest.

The hydrodynamic model consists of the Spectral Element Ocean Model (SEOM) in its two dimensional shallow water version. The spatial discretization relies on the spectral element method, an *h-p* type finite element discretization, which relies on relatively high degree (5-8th) polynomials to approximate the solution within each element. The main features of the spectral element method are: geometric flexibility due to its unstructured grids, dual paths to convergence: exponential by increasing polynomial degree or algebraic via increasing the number of elements, dense computational kernels with sparse inter-element synchronization, and excellent scalability on parallel machines. The model can be forced through winds, tides, and lateral injection of mass at inflow boundaries (e.g., river input). The model is supplemented with an advection-diffusion equation to simulate the trajectory of contaminants as they are carried along by the simulated flow.

Using multiple linear regression the Bootstrap Error-adjusted Single-sample Technique (BEST) classification algorithm can be performed in situ, allowing a rover to classify many samples, only notifying the simulation when an interesting substance is found. Once the spectrum of a sample has been collected, it must be classified to determine the substance present. Spectra recorded at n wavelengths are represented as single points in a n-dimensional hyperspace. In this scheme, similar samples produce similar spectra that project as probability orbitals or clusters into similar regions of hyperspace. The BEST metric is a clustering technique for exploring these distributions of spectra in hyperspace.

An initial library can be computed based on substances likely to be found in the target environment. When a substance unknown to the BEST library is found, the sensor can sample nearby points with similar spectra to create a new library entry for the new substance. Scientists can determine the type of substance present by further analyzing raw spectra of the substance provided by SSSI and by using data from their other instruments, apply these data to update the simulation. The SSSI chemical library will comprise substances expected to be in the environment in which the SSSI operates.

The initial deployment of the sensor and model focuses on estuarine regions where water quality monitoring is critical for human health and environmental monitoring. A sample tidal calculation will be performed using a grid that encompasses a bay or set of bays regions and possibly a river region. The model is forced with tidal elevation obtained from tide gauges. Runs without data assimilation have shown good comparison with observation and previous modeling results.

However, for DDDAS the use of data assimilation is imperative to inject observational data in the model while accounting for model and observational errors.

The data assimilation reduces the computational errors associated with initial data, essentially the solution at previous time step, and improves the prediction. Using the first set of measurements, the approximation of the initial data is recovered. As new data are incorporated into the simulator, the initial data are updated using an objective function. We note that the formulated problem is ill posed because there are fewer sensors than the finite dimensional space describing the initial data. The objective function is set up based on both a measurement error as well as a penalization term that depends on the prior knowledge about the solution at previous time steps (or initial data). The prior information is refreshed using the updated initial data. The penalization constants depend on time of update and can be associated with the relative difference between simulated and measured values. In the simulations, both the prior and penalization constants change in time.

To account for the errors (uncertainties) associated with sensor measurements, we consider an initial data update within a Bayesian framework. The posterior distribution is set up based on measurement errors and prior information. This posterior distribution is complicated and involves the solutions of partial differential equations. We could use a Metropolis-Hasting Markov chain Monte Carlo (MCMC) method to generate samples from the posterior distributions. However, a sampling with MCMC is expensive since it requires iterative steps and the acceptance rate is typically low. We developed an approach that combines least squares with a Bayesian approach that gives a high acceptance rate. In particular, we can prove that rigorous sampling can be achieved by sampling the sensor data from the known distribution, thus obtaining various realizations of the initial data. Our approach has similarities with the Ensemble Kalman Filter approach, which can also be adapted to an initial data update.

Consider finding hydrocarbon fuel in a body of water. Gasoline can simply be a sign of pollution from a small boat. Heavier fuel oils could be an indication that a larger boat has a leak or sank recently nearby. Jet fuel could come from a downed aircraft. The SSSI needs to be reprogrammed in the sunken vehicle case and a search and locate application must be invoked to find the accident and rescue any people that may be in danger. Emergency services, the coast guard, and the news media may need to be automatically informed of progress.

Oil droplets can travel nearly anywhere in the ocean. The droplet size exerts a major effect on droplet motion. The rise velocity of oil droplets extends from about 2.5×10^{-7} m/s for a diameter of 2µ m to 4.3×10^{-3} m/s for a diameter of 260µ m. Droplets traveling at 2.5×10^{-7} m/s will ascend only 0.001 m and 0.02 m, over periods of 1 hour and 24 hours, respectively. However, droplets ascending at 4.3×10^{-3} m/s will climb 15 m and 370 m over equivalent periods. A vertical diffusivity of 51 cm^2/s will distribute oil droplets (equally upward and downward) about 6 m and 30 m over the same time. Therefore, the smallest oil droplets act as though they are neutrally buoyant, i.e., transported only by diffusion. However, buoyancy primarily advects the largest droplets.

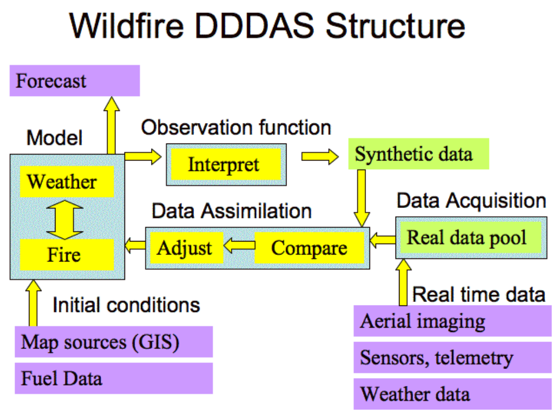


Fig. 3. Schematic diagram of a wildland fire dynamic data-driven application system.: blue blocks are functional units and purple are data inputs and outputs

5 Wildland Fireline Predictive DDDAS

Our wildland fire DDDAS is built upon a previously existing coupled atmosphere-wildfire model. Components have been developed and added which (1) save, modify, and restore the state of the atmosphere-wildfire model, (2) apply ensemble data assimilation algorithms to modify ensemble member states by comparing the data with synthetic data of the same kind created from the simulation state, (3) retrieve, process, and ingest data from both novel ground-based sensors and airborne platforms in the near vicinity of a fire, and (4) provide computational results visualized in several ways adaptable to user needs. Fig. 3 presents the actual software structure. The observation function interprets the model variables in terms of observable quantities and produces synthetic data from the model state. The data assimilation compares the synthetic data and the real data, and adjusts the model state accordingly.

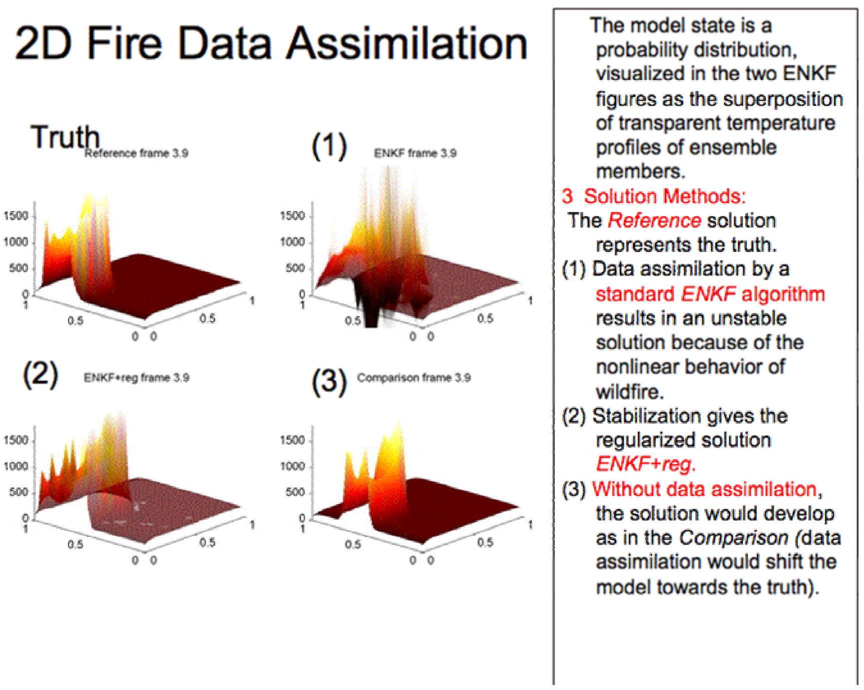
The original modeling system is composed of two parts: (1) a numerical weather prediction model and (2) a fire behavior model that models the growth of a wildfire in response to weather, fuel conditions, and terrain. Both models are two way coupled so that heat and water vapor fluxes from the fire feed back to the atmosphere to produce fire winds, while the atmospheric winds and changes in humidity in turn drive the fire propagation. This wildfire simulation model can thus represent the complex interactions between a fire and the atmosphere.

The meteorological model is a three dimensional non-hydrostatic numerical model based on the Navier-Stokes equations of motion, a thermodynamic equation, and conservation of mass equations using the anelastic approximation. Vertically stretched terrain following coordinates allow the user to simulate in detail the airflow over complex terrain. Forecasted changes in the larger scale atmospheric environment are used to initialize the outer of several nested domains and update lateral boundary conditions. Two way interactive nested grids capture the outer forcing domain scale of the synoptic scale environment while allowing the user to telescope down to tens of meters near the fireline through horizontal and vertical grid refinement. Weather processes such as the production of cloud droplets, rain, and ice are parameterized using standard treatments.

Local fire spread rates depend on the modeled wind components through an application of the Rothermel fire spread formula [11]. The heat release rate is based on [12] which characterizes how the fire consumes fuels of different sizes with time after ignition, distinguishing between rapidly consumed grasses and slowly burned logs. Within each atmospheric grid cell, the land surface is further divided into fuel cells, with fuel characteristics corresponding to the 13 standard fuel types [13]. Each fuel cell has four tracers, which identify burning areas of fuel cells and define the fire front. Fire spread rates are calculated locally along the fire as a function of fuels, wind speed and direction from the atmospheric model (which includes the effects of the fire), and terrain slope while a local contour advection scheme assures consistency along the fireline. The canopy may be dried and ignited by the surface fire, so a simple radiation treatment distributes the sensible and latent heat into the lowest atmospheric grid levels.

The empirical fire model uses a submesh representation of the fire region. Within each cell on the fire model grid, a quadrilateral defines the burning region. The burning area in each grid cell is defined by the position of four moving points, called tracers. This representation makes the fire area hard to adjust in data assimilation. As a result, we have developed a translation of the tracers into a level function. The level function is given by values at nodes of the fire grid. The fire region is where the level function is positive. The absolute value of the level function is approximately equal to the Euclidean distance from the fireline. In data assimilation, the level function can be increased or decreased just like the physical quantities in the model and greatly simplifies the assimilation process.

Ensemble filters work by advancing in time a collection of simulations started from randomly perturbed initial conditions. When the data is injected, the *forecast ensemble* is updated to get a new *analysis ensemble* to achieve a least squares fit using two conditions: change in the ensemble members should be minimized, and the data d should fit the ensemble members state u, $h(u) \approx d$, where h is called the observation function. The weights in the least squares are obtained from the covariances of the ensemble and of the data error. For comprehensive surveys of Ensemble Kalman Filters (*EnKF*) techniques, see [14-16]. In general, an EnKF works by forming the analysis ensemble as linear combinations of the forecast ensemble. This raises two concerns, especially in highly nonlinear models: if the change of state in the update is large there may not be suitable forecast members to make linear combinations of in order to match the data. Hence, a linear combination of realizable states may not itself be a realizable state. This results in the need for large ensembles, frequent small updates, and has the potential to break down due to nonphysical states being introduced.



The model state is a probability distribution, visualized in the two ENKF figures as the superposition of transparent temperature profiles of ensemble members.

3 Solution Methods:

The *Reference* solution represents the truth.

(1) Data assimilation by a standard *ENKF* algorithm results in an unstable solution because of the nonlinear behavior of wildfire.

(2) Stabilization gives the regularized solution *ENKF+reg*.

(3) Without data assimilation, the solution would develop as in the *Comparison* (data assimilation would shift the model towards the truth).

Fig. 4. Comparison of the results of 4 methods of simulating 2-D growth of a fire using an ensemble of solutions where the vertical axis is temperature and the 2 horizontal axes represent x- and y- spatial dimensions

We were using filters based on the EnKF with data perturbation. The data assimilation always produced an ensemble with nonphysical solutions and so that the simulation always broke down numerically. Therefore, we have developed a regularization by adding a term involving the change in the spatial gradient of ensemble members to the least squares procedure [17].

Consider Fig. 4 [18]. The exact solution is shown in the upper left. The ensemble solution with a standard EnkF algorithm is shown in the upper right, which creates unstable and nonphysical solutions. An EnkF solution with stabilization with the Johns and Mandel (2004) method is shown in the lower left, which produces the best, physically realistic solution. The solution of the ensemble without any data assimilation is shown in the lower right, in which the solution of the ensemble drifts away from the solution.

Existing ensemble filter formulas assume that the observation function is linear and then compute with the observation matrix *H*. To simplify the software, we have derived a mathematically equivalent ensemble filter that only needs to evaluate *h(u)* for each ensemble member. The ensemble update involves computation with extremely large, dense matrices.

There is clearly a need to adjust the simulation state by distorting the simulation state in space rather than employing an additive correction to the state. Also, while the position of the feature may have error distribution that is approximately gaussian, this is not necessarily the case for the value of the state at a given point. For this reason, alternative error models including the position of features were considered in the literature [19] and a number of works emerged that achieve more efficient movement of features by using a spatial transformation as the field to which additive corrections are made: a transformation of the space by a global low order polynomial mapping to achieve alignment [20], and two-step models to use alignment as preprocessing to an additive correction [21, 22]. We have proposed [23] a new method, a Morphing Ensemble that combines alignment and additive correction into a single step, using ideas borrowed from registration and morphing in image processing [24].

Data comes from fixed sensors that measure temperature, radiation, and local weather conditions. The fixed sensors, positioned so as to provide weather conditions near a fire, are mounted at various heights above the ground on a pole with a tripod base. The data logging and transmission electronics are buried in the soil in a protective box. Wiring to the sensors and antennae is insulated. This type of system will survive burn overs by low intensity fires. These sensors supplement other sources of weather data derived from permanent and portable automated weather stations. The temperature and radiation measurements provide the direct indication of the fire front passage and the radiation measurement can also be used to determine the intensity of the fire. The raw data is logged and transmitted as comma delimited ASCII text for easy use in spreadsheets.

Data also comes from images taken by sensors on either satellites or airplanes. Camera calibration, an inertial measurement unit, GPS, and digital elevation data are used in a processing system to convert raw images to a map product with a latitude and longitude associated with each pixel. The three wavelength infrared images can then be processed using a variety of algorithm approaches to extract which pixels contain a signal from fire and to determine the energy radiated by the fire. The original pixel values, the derived probability of fire in each pixel, and the latitude and longitude information are stored in a Data Center as GeoTIFF images.

Data from previous fires are stored in a data center in GeoTIFF (images), Excel spreadsheet files, or text files (sensors). The Excel data is made more accessible by converting it to a comma separated value (CSV) format. GPS information is stored about each fixed-location sensor. Each sensor's data is time stamped to identify when the data was collected or received (if it comes without a time stamp). For mobile sensors, both the time stamp and GPS information is available.

Data that comes into the data center must go through a process consisting of up to six steps:

- *Retrieval*: Get the data from sensors. This may mean receiving data directly from a sensor or indirectly through another computer or storage device (e.g., a disk drive).
- *Extraction*: The data may be quite messy in raw form, thus the relevant data may have to be extracted from the transmitted information.
- *Conversion*: The units of the data may not be appropriate for our application.

- *Quality control*: Bad data should be removed or repaired if possible. Missing data (e.g., in a composite satellite photo) must be repaired.
- *Store*: The data must be archived to the right medium (or media). This might mean a disk, tape, or computer memory, or no storage device at all if data is not being archived permanently or only temporarily.
- *Notification*: If a simulation is using the data as it comes into the data center, the application must be informed of the existence of new data.

The data is related to the model by the observation equation $h(u) \approx d$. The observation function h maps the system state u to synthetic data, which are the values the data would be in the absence of modeling and measurement errors. Knowledge of the observation function, the data, and an estimate of the data error covariance is enough to find the correct linear combinations of ensemble members in the ensemble filter. The data assimilation code also requires an approximate inverse g of the observation function. For a system state u and data d, is the direction in which the system state can change to decrease a norm of the data residual. For an observation function that is simply the value of a variable in the system state, the natural choice of approximate inverse can be just the corresponding term of the data residual, embedded in a zero vector.

Building the observation function and its approximate inverse requires conversion of physical units between the model and data, and conversion and interpolation of physical coordinates. In addition, synthetic data at instants of time between the simulation time of ensemble members need to be interpolated to the data time. Data is injected into the ensemble to minimize both a weighted sum of the data residual and the change in the ensemble.

The data items enter in a pool maintained by the data acquisition module. The assimilation code can query the data acquisition module to see if there are any new data items available, request their quantitative and numerical properties, and delete them from the pool after they are no longer of use. The properties of the data items include

- a time stamp,
- encoding of the type and parameter values of the observation function and its approximate inverse,
- estimate of the error of the data, and
- the numerical values of the data itself.

From the point of view of the assimilation code, all information about physical units, etc., is encoded in the observation function.

Visualization of the model output as an image is accomplished by brightness, color encoding, and transparency for a visual indication of the location and intensity of the fire, and of the probability distribution of the forecast. 3-D visualization of the fire is more complex and complexity increases if high spatial resolution of the output is desired. 3-D visualization uses model output from the fire propagation code for the flame region and from the atmospheric code for visualization of smoke. Ensemble statistics are used for visualization of probability.

The geographic output of the fire model in 2-D or 3-D is visualized in a number of ways:

- For computer based mapping, manipulation, and visualization of the model output, file formats compatible with the geographic information system (GIS) products are generated.
- A PDF file: the output is a map generated for potential output as hardcopy view of the fire at a set point in time.
- A MPEG-4 (or similar format) file: the time varying output for both 2-D and 3-D is also used to generate a movie.
- A file appropriate for viewing as a layer on top of Google Earth [25].

Our Google Earth Fire visualization system (see Fig. 5) greatly simplifies map and image visualization. The user can control the viewing perspective, zooming into specific sites, and selecting the time frame of the visualization within the parameters of the current available simulation.

6 Concluding Remarks

DDDAS is an interesting field that is trying to abstract into a science a number of previously treated areas. Data assimilation, control engineering, process control, cyber physical systems, and other buzzwords describe special cases of DDDAS (unless you are a researcher in these fields, in which case DDDAS is a special case of your own field).

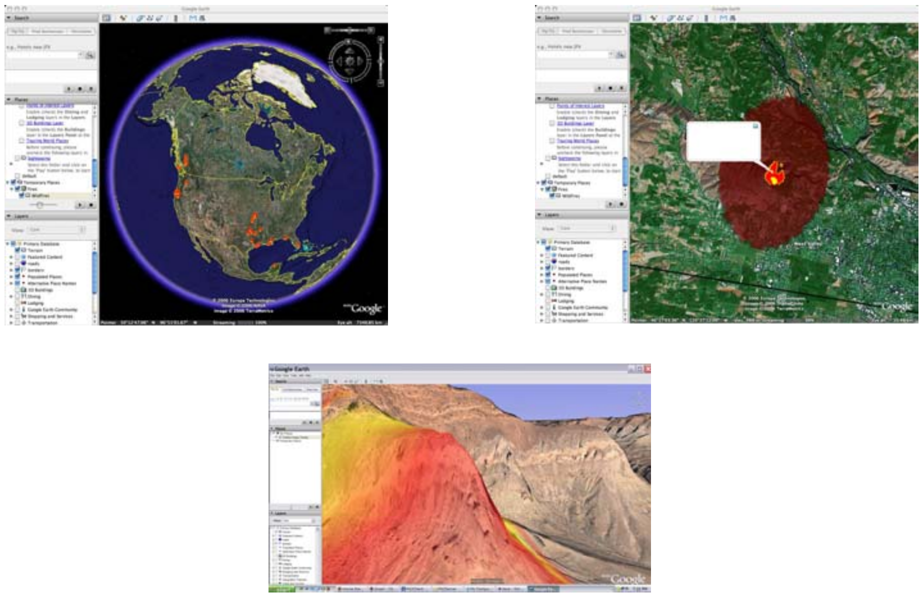


Fig. 5. Google Earth Fire Layering software tool: top left picture is what you get when clicking on Wildfires, top right picture is what you get by clicking on one of the fire symbols on the top left, and the bottom picture shows how the 3D layering appears

There are many application areas in which data can be injected into a running process. Doing it right leads to applications that can run forever instead of simulating short periods of time using static, initial data. While long range predictions can be achieved (e.g., weather prediction) using many runs with different sets of initial data from slightly different initial times, it is not the same as running just one simulation.

Making one traditional application starting from the static, initial data into an application that uses dynamic data to run a long time is good engineering. Abstracting what makes many different applications run as a DDDAS is good science, which is completely different. The purpose of the DDDAS program at the NSF is to do good science that is also good engineering. However, the list of DDDAS projects on http://www.dddas.org goes far beyond traditional engineering topics.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under grants CNS 0325314, CNS-0324988, CNS-0324989, CNS-0324876, CNS-0324910, ACI-0305466, EIA-0219627, CNS-0540178, CNS-0540153, CNS-0540155, CNS-0540136, and OISE-0405349.

See [26-28] for further information.

References

1. F. Darema et al, DDDAS: Dynamic Data Driven Applications Systems, program solicitation 05-570, National Science Foundation, Arlington, VA, 2005; Site http://www.nsf.gov/pubs/2005/nsf05570/nsf05570.htm visited 10/28/2006.
2. F. Darema, Introduction to the ICCS2006 Workshop on Dynamic Data Driven Applications Systems, in Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III, edited by V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, and J.J. Dongarra, Lecture Notes in Computer Science 3993, Springer-Verlag Heidelberg, 2006, pp. 375-383.
3. R.E. Ewing, Interactive Control of Large scale Simulations, Workshop on Dynamic Data-Driven Application Systems, National Science Foundation, Arlington, VA, March 8-10, 2000. Site http://www. dddas.org/NSFworkshop2000.html visited 10/28/2006.
4. C.C. Douglas, DDDAS.org. Includes project descriptions, many DDDAS workshop virtual proceedings, and links to DDDAS software. Site http://www.dddas.org visited 10/28/2006.
5. 2003 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2003: 3rd International Conference, Melbourne, Australia and St. Petersburg, Russia, June 2-4, 2003, Proceedings, Part IV, P.M.A. Sloot, D. Abramson, A.V. Bogdanov, J.J.

Dongarra, A.Y. Zomaya, Y.E. Gorbachev (Eds.), Lecture Notes in Computer Science, Vol. 2660, Springer-Verlag Heidelberg, 2003, pp. 279-384.

6. 2004 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part III, Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and J.J. Dongarra (eds.), Lecture Notes in Computer Science series, vol. 3038, Springer-Verlag Heidelberg, 2004, pp. 662-834.
7. 2005 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science - ICCS 2005: 5th International Conference, Atlanta, Georgia, USA, May 22-25, 2005, Proceedings, Part II, Vaidy S. Sunderam, Geert Dick van Albada, Peter M.A. Sloot, Jack J. Dongarra (eds.), Lecture Notes in Computer Science series, vol. 3515, Springer-Verlag Heidelberg, 2005, pp. 610-745.
8. 2006 Dynamic Data-Driven Application Workshop, F. Darema, ed., in Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III, edited by V.N. Alexandrov, G.D. van Albada, P.M.A. Sloot, and J.J. Dongarra, Lecture Notes in Computer Science 3993, Springer-Verlag Heidelberg, 2006, pp. 375-607.
9. K. Baldridge, G. Biros, A. Chaturvedi, C.C. Douglas, M. Parashar, J. How, J. Saltz, E. Seidel, A. Sussman, January 2006 DDDAS Workshop Report, National Science Foundation, 2006. Site http://www.dddas.org/nsf-workshop-2006/wkshp_report.pdf visited 10/28/2006.
10. D. Metaxas and G. Tsechpenakis, Dynamic Data Driven Coupling of Continuous and Discrete Methods in 3D Tracking, in Computational Science - ICCS 2005: 5th International Conference, Atlanta, Georgia, USA, May 22-25, 2005, Proceedings, Part II, Vaidy S. Sunderam, Geert Dick van Albada, Peter M.A. Sloot, Jack J. Dongarra (eds.), Lecture Notes in Computer Science series, vol. 3515, Springer-Verlag Heidelberg, 2005, pp. 712-720.
11. R.C. Rothermel, A mathematical model for predicting fire spread in wildland fires. USDA Forest Service Research Paper INT-115, 1972.
12. F.A. Albini, PROGRAM BURNUP, A simulation model of the burning of large woody natural fuels, Final Report on Research Grant INT-92754-GR by U.S.F.S. to Montana State University, Mechanical Engineering Dept., 1994.
13. H. Anderson, Aids to determining fuel models for estimating fire behavior. USDA Forest Service, Intermountain Forest and Range Experiment Station, INT-122, 1982.
14. G. Evensen, The ensemble Kalman filter: Theoretical formulation and practical implementation, *Ocean Dynamics*, 53 (2003), pp. 343–367.
15. G. Evensen, Sampling strategies and square root analysis schemes for the EnKF. *Ocean Dynamics*, 54 (2004), pp. 539–560.

16. M.K. Tippett, J. L. Anderson, C.H. Bishop, T.M Hamill, J.S. Whitaker, Ensemble square root filters, *Monthly Weather Review*, 131 (2003), pp. 1485–1490.
17. C.J. Johns and J. Mandel, A two-stage ensemble Kalman filter for smooth data assimilation, to appear in Environmental and Ecological Statistics, Conference on New Developments of Statistical Analysis in Wildlife, Fisheries, and Ecological Research, Oct 13-16, 2004, Columbia, MI, 2006.
18. J. Mandel, L.S. Bennethum, J.D. Beezley, J.L. Coen, C.C. Douglas, L.P. Franca, M. Kim, and A. Vodacek, A wildland fire model with data assimilation, CCM Report 233, 2006. Site http://www.math.cudenver.edu/~jmandel/papers/rep233.pdf visited 10/28/2006.
19. R.N. Hoffman, Z. Liu, J.-F. Louis, and C. Grassoti, Distortion representation of forecast errors, *Monthly Weather Review*, 123 (1995), pp. 2758–2770.
20. G.D. Alexander, J.A. Weinman, and J.L. Schols, The use of digital warping of microwave integrated water vapor imagery to improve forecasts of marine extratropical cyclones, *Monthly Weather Review*, 126 (1998), pp. 1469–1496.
21. W.G. Lawson and J.A. Hansen, Alignment error models and ensemble-based data assimilation, *Monthly Weather Review*, 133 (2005), pp. 1687–1709.
22. S. Ravela, K.A. Emanuel, and D. McLaughlin, Data assimilation by field alignment. *Physica D*, to appear, 2006.
23. J. Mandel and J.D. Beezley, Predictor-corrector and morphing ensemble filters for the assimilation of sparse data into high dimensional nonlinear systems, to appear, 11th Symposium on Integrated Observing and Assimilation Systems for the Atmosphere, Oceans, and Land Surface (IOAS-AOLS), CD-ROM, Paper 3.12, 87th American Meterological Society Annual Meeting, San Antonio, TX, January 2007.
24. L.G. Brown, A survey of image registration techniques, *ACM Computing Surveys*, 24 (1992), pp. 325–376.
25. S. Chakraborty, J. Hatcher, D. Bansal, and C.C. Douglas, Google Earth Fire Layering Tool, in preparation, 2006.
26. C.C. Douglas, ML-DDDAS research group web site and reports, site http://www.mgnet.org/~douglas/ml-dddas.html visited 10/28/2006.
27. C.R. Johnson, Scientific Computing and Imaging Institute at the University of Utah, site http://www.sci.utah.edu visited 10/28/2006.
28. J. Mandel, NSF funded wildfire project public web site, site http://www-math.cudenver.edu/~jmandel/fires visited 10/28/2006.