Demonstrating the Validity of a Wildfire DDDAS

Craig C. Douglas[1,2], Jonathan D. Beezley[4], Janice Coen[3], Deng Li[1], Wei Li[1], Alan K. Mandel, Jan Mandel[4], Guan Qin[5], and Anthony Vodacek[6]

[1] University of Kentucky, Department of Computer Science, 773 Anderson Hall, Lexington, KY 40506-0046, USA
{deng.li, wli4}@uky.edu
[2] Yale University, Department of Computer Science, P.O. Box 208285 New Haven, CT 06520-8285, USA
douglas-craig@cs.yale.edu
[3] National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307-3000, USA
janicec@ucar.edu
[4] University of Colorado at Denver and Health Sciences Center, Department of Mathematical Sciences, P.O. Box 173364, Denver, CO 80217-3364, USA
{jbeezley, jmandel}@math.cudenver.edu
[5] Texas A&M University, Institute for Scientific Computation, 612 Blocker, 3404 TAMU, College Station, TX, 77843-3404, USA
guan.qin@tamu.edu.
[6] Rochester Institute of Technology, Center for Imaging Science, Rochester, NY 14623 USA
vodacek@cis.rit.edu.

Abstract. We report on an ongoing effort to build a Dynamic Data Driven Application System (DDDAS) for short-range forecast of weather and wildfire behavior from real-time weather data, images, and sensor streams. The system changes the forecast as new data is received. We encapsulate the model code and apply an ensemble Kalman filter in time-space with a highly parallel implementation. In this paper, we discuss how we will demonstrate that our system works using a DDDAS testbed approach and data collected from an earlier fire.

1 Introduction

We describe the current state of a dynamic data driven application system (DDDAS) for simulating wildfires. The motivation for this work is the obvious societal value of an accurate forecast compounded with the inherent challenge in modeling this nonlinear, rapidly-changing phenomena, the difficulty in obtaining remote or in situ data about the fire itself, and the challenges of communicating the on-site, out-of-order data of unknown quality to supercomputers and using it to steer the model simulations. The work necessarily extends beyond data assimilation work in progress in atmospheric science due to the specific application

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 522–529, 2006.

challenges: the model is strongly nonlinear and irreversible, and the data arrives out-of-order from disparate data sources.

The 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.

2 Summary of the Atmosphere-Fire Model

The original modeling system is composed of two parts: a numerical weather prediction model and a fire behavior model that models the growth of a wildfire in response to weather, fuel conditions, and terrain [1, 2]. These 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 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 domain 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 in 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 [3]. The heat release rate is based on [4] 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 [5]. Four tracers, assigned to each fuel cell, 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. Then a simple radiation treatment distributes the sensible and latent heat into the lowest atmospheric grid levels.

The empirical fire model is using a submesh representation of the fire region. Within each cell on the fire model grid, the fireline is approximated by a straight line. The fire area in each grid cell is encoded into the position of four points, called tracers. Two of the tracers are generally the intersection of the fireline with the edges of the grid, but not always. This representation makes the fire area hard to adjust in data assimilation (Sec. 3). For this reason, 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.

Our current experiments are with a simple fire model [6], which uses the reaction-convection-diffusion equation for the temperature T and fuel supply S,

$$c\frac{\partial T}{\partial t} = -\nabla d\nabla T - av\cdot\nabla T + e\frac{\partial S_k}{\partial t} - b\,(T - T_a)\,, \tag{1}$$

$$\frac{\partial S}{\partial t} = -f(T)S. \tag{2}$$

Eq. (1) is the balance of heat. The term $-\nabla d\nabla T$ models the heat diffusion, $-av\cdot\nabla T$ is the convection by wind with speed v, $e\frac{\partial S_k}{\partial t}$ is the heat generated by burning the fuel, and $-b\,(T - T_a)$ is the heat lost by flux to to the ambient environment with temperature T_a. Eq. (2) is the balance of fuel. The equations are discretized by simple central finite differences in space and the second order trapezoidal method time. The nonlinear system in each timestep is solved by a matrix-free Newton-GMRES method with preconditioning by FFT inversion of the diffusion term. This numerical approach is well suited for a diffusion dominated problem. Methods for convection dominated problems (strong winds relative to the mesh scale) involving upwinding and nonsymmetric preconditioning are in development. This simple model is capable of producing a reasonable fire behavior with an advancing fire front. A more realistic model is under development, which will include several species of fuel, radiative heat transfer between, and evaporation of moisture. It is anticipated that this model will replace the empirical fire model and it will be coupled to the atmospheric model. For related physics based fire models in the literature, see, e.g., [7, 8].

3 How the Ensemble Data Assimilation Works

Ensemble filters work by advancing in time a collection of simulations started from randomly perturbed initial conditions. When the data is injected, the ensemble (called *forecast*) is updated to get a new ensemble (called *analysis*) 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,

$$h(u) \approx d, \tag{3}$$

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 EnKF techniques, see [9, 10, 11]. In general, *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 take 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 and frequent small updates, and has the potential for breakdown.

We are using filters based on the EnKF with data perturbation [12]. But, even with the simple wildfire model (1)-(2), the data assimilation always produces an ensemble with nonphysical solutions and then the simulations breaks down numerically. Therefore, we have proposed a regularization by adding a term involving the change in the spatial gradient of ensemble members to the least squares [13]. Existing ensemble filter formulas assume that the observation function is linear, $h(u) = Hu$, 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 large dense matrices. Currently we are using the SCALAPACK parallel linear algebra engine. Future developments include the treatment of nonlinear observation functions, and a hybrid deterministic/Monte Carlo filter that can overcome very strong nonlinearity by modifying the ensemble members to attract them to the truth rather than relying on linear combinations, while maintaining correct ensemble statistics. An approximate one-sided inverse of the observation function is needed for this (Sec. 6). For assimilation of out-of-order data, we will use system states that combine states at several times [6]. The parallel computing framework we have developed was designed with this in mind.

4 Data Transmission Types

Data comes from fixed sensors which measure temperature, radiation, and local weather conditions [14]. 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-over 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.

Data also comes from images taken by sensors on either satellites or airplanes. The primary source of image data is the Wildfire Airborne Sensor Project (WASP) [15]. This three wavelength digital infrared camera system is carried on an airplane that is flown over the fire area. 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 [15, 16] to extract which pixels contain a signal from fire and to determine the energy radiated by the fire [17, 18]. The original pixel values, the derived probability of fire in each pixel, and the latitude and longitude information are stored as GeoTIFF.

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 (or only briefly) 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 needs to be informed of the existence of new data.

5 How the Data Moves

In order to run a demonstration simulation, we create a script that lists the sensors that will be included in a run. Multiple runs are handled with a script each.

We are using an Apple XGrid in a University of Kentucky computer science student laboratory to introduce sensor uncertainty into the simulations, independent of the actual scripts we create. The sensors provide data to the running simulation, which is currently in Colorado. The methodology is portable to any Grid environment, not just Apple's.

When a computer in the sensor Grid is idle, data streams regularly based on the scripts we created. We have to do time zone translation on the time stamps when running a simulation in order to be robust. When someone logs into a computer in the sensor Grid and uses the computer, our sensor code is put to sleep automatically by the XGrid controller and the sensors therein are offline until the computer has been idle for some period of time that we cannot control. Using only cycles on idle computers is typical of many Grid environments.

The actual network streaming tool uses a standard TCP client-server scheme. The TCP protocol is a reliable byte stream protocol and underlying network instabilities are accommodated reliably. The programs are small Java codes using built in networking (e.g., write once, write all capabilities) and data encryption methods. The programs are fast and run on almost anything without recompiling.

The sensor network operates either actively (sending data to a specific set of locations) or passively (sending data to requestors). In an active mode, each sensor is represented by a single client (more than one sensor can be on the client, however). When it is time to send data, three way handshaking is used to create a TCP connection to the remote server and send data through it. In a passive mode, each sensor is a server and clients receive data by creating a connection to it and reading data.

The data used by the sensor network is normally kept in a remote data center and only data to be transmitted soon is prefetched. The actual computer load is quite small since much of the time the sensor program is asleep and only periodically does it wake up to do data transfers. What is important is network bandwidth and reliability. Having an acceptable latency time is less important.

6 How the Data Is Injected into the Ensemble Process

The data is related to the model by the observation equation (3). 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, $g\,(h(u) - d)$ is the direction in which the system state can change to decrease a norm of the data residual $h\,(u) - d$. 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. The data injection itself is done by updating the ensemble to minimize a weighted sum of the data residual and the change in the ensemble (as described in Sec. 3).

The data items enter in a pool maintained by the data acquisition module. The assimilation code can inquire the data acquisition module 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 needed. 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 finally 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.

7 How We Deliver Computational Results

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. A PDF file of the output as a map is generated for potential output as hardcopy view of the fire at a set point in time. For computer based mapping, manipulation, and visualization of the model output, file formats compatible with the geographic information system (GIS) products are generated.

The time varying output for both 2-D and 3-D is also used to generate a movie playable in any of the media formats, e.g., MPEG. The user may select movie duration up to the maximum extent of the model forecast.

An intuitive and easy method for map visualization is to use a web-based mapping server, e.g., GIS software, Google Maps, or Google Earth. These web-based programs simplify access to map and image data. They let us display model output movies on top of a relevant map background. Within Google Earth, for example, this allows User control of the viewing perspective, zooming into specific sites, and selecting the time frame of the visualization within the parameters of the current available simulation. These web-based programs also allow switching between background types, for example, USGS topographic maps or high resolution satellite images with a road layer or other pertinent layers such water sources added.

8 Conclusions

We are progressing towards a full blown computational test. Data will move, possibly unreliably, from remote sensors to a remote computational machine. Our simulations will be data-driven in terms of the models and scales we use. The choices of models and scales (as in multiscales or resolution) will be made in part based on the data streaming in. We are now in a position to develop the final piece of our DDDAS strategy: having the simulation control how much data is needed and from where in order to improve the quality of the flame wavefront predictions. Only then will we have a truly symbiotic relationship between the running computations and data collection. Our current test should have the right ingredients to predict how our DDDAS will work in a planned future field test with a real wildland fire.

References

1. Clark, T.L., Coen, J., Latham, D.: Description of a coupled atmosphere-fire model. Intl. J. Wildland Fire **13** (2004) 49–64
2. Coen, J.L.: Simulation of the Big Elk Fire using using coupled atmosphere-fire modeling. International J. of Wildland Fire **14**(1) (2005) 49–59
3. Rothermel, R.C.: A mathematical model for predicting fire spread in wildland fires. USDA Forest Service Research Paper INT-115 (1972)
4. Albini, F.A.: 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 Univ., Mechanical Engineering Dept. (1994)
5. Anderson, H.: Aids to determining fuel models for estimating fire behavior. USDA Forest Service, Intermountain Forest and Range Experiment Station, INT-122 (1982)
6. Mandel, J., Chen, M., Franca, L.P., Johns, C., Puhalskii, A., Coen, J.L., Douglas, C.C., Kremens, R., Vodacek, A., Zhao, W.: A note on dynamic data driven wildfire modeling. In Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J., eds.: Computational Science - ICCS 2004. Volume 3038 of Lecture Notes in Computer Science. Springer (2004) 725–731
7. Linn, R., Reisner, J., Colman, J.J., Winterkamp, J.: Studying wildfire behavior using FIRETEC. Int. J. of Wildland Fire **11** (2002) 233–246
8. Serón, F.J., Gutiérrez, D., Magallón, J., Ferragut, L., Asensio, M.I.: The evolution of a WILDLAND forest FIRE FRONT. Visual Computer **21** (2005) 152–169
9. Evensen, G.: The ensemble Kalman filter: Theoretical formulation and practical implementation. Ocean Dynamics **53** (2003) 343–367
10. Evensen, G.: Sampling strategies and square root analysis schemes for the EnKF. Ocean Dynamics (2004) 539–560
11. Tippett, M.K., Anderson, J.L., Bishop, C.H., Hamill, T.M., Whitaker, J.S.: Ensemble square root filters. Monthly Weather Review **131** (2003) 1485–1490
12. Burgers, G., van Leeuwen, P.J., Evensen, G.: Analysis scheme in the ensemble Kalman filter. Monthly Weather Review **126** (1998) 1719–1724
13. Johns, C.J., Mandel, J.: A two-stage ensemble Kalman filter for smooth data assimilation. Environmental and Ecological Statistics. Conference on New Developments of Statistical Analysis in Wildlife, Fisheries, and Ecological Research, Oct 13-16, 2004, Columbia, MI (in print)
14. Kremens, R., Faulring, J., Gallagher, A., Seema, A., Vodacek, A.: Autonomous field-deployable wildland fire sensors. International J. of Wildland Fire **12** (2003) 237–244
15. Li, Y., Vodacek, A., Kremens, R.L., Ononye, A., Tang, C.: A hybrid contextual approach to wildland fire detection using multispectral imagery. IEEE Trans. Geosci. Remote Sens. **43** (2005) 2115–2126
16. Dozier, J.: A method for satellite identification of surface temperature fields of subpixel resolution. Remote Sens. Environ. **11** (1981) 221–229
17. Wooster, M.J., Zhukov, B., Oertel, D.: Fire radiative energy for quantitative study of biomass burning: derivation from the BIRD experimental satellite and comparison to MODIS fire products. Remote Sensing of Environment **86** (2003) 83–107
18. Smith, A.M.S., Wooster, M., Drake, N., Perry, G., Dipotso, F., Falkowski, M., Hudak, A.: Testing the potential of multi-spectral remote sensing for retrospectively estimating fire severity in African savanna environments. Remote Sens. Environ. **97** (2005) 92–115