

The comparison of four dynamic systems-based software packages: Translation and sensitivity analysis

Donna M. Rizzo^{a,1}, Paula J. Mouser^{a,*}, David H. Whitney^{a,1}, Charles D. Mark^{a,1},
Roger D. Magarey^{b,2}, Alexey A. Voinov^{c,3}

^a University of Vermont, Department of Civil & Environmental Engineering, 213 Votey Building, Burlington, VT 05405, USA

^b United States Department of Agriculture, 1017 Main Campus Drive, Suite 1550, Raleigh, NC 27606, USA

^c University of Vermont, Gund Institute for Ecological Economics & Computer Science Department, 590 Main Street,
Burlington, VT 05405, USA

Received 3 May 2004; received in revised form 12 July 2005; accepted 22 July 2005
Available online 19 October 2005

Abstract

Dynamic model development for describing complex ecological systems continues to grow in popularity. For both academic research and project management, understanding the benefits and limitations of systems-based software could improve the accuracy of results and enlarge the user audience. A Surface Wetness Energy Balance (SWEB) model for canopy surface wetness has been translated into four software packages and their strengths and weaknesses were compared based on ‘novice’ user interpretations. We found expression-based models such as Simulink and GoldSim with Expressions were able to model the SWEB more accurately; however, stock and flow-based models such as STELLA, Madonna, and GoldSim with Flows provided the user a better conceptual understanding of the ecologic system. Although the original objective of this study was to identify an ‘appropriate’ software package for predicting canopy surface wetness using SWEB, our outcomes suggest that many factors must be considered by the stakeholders when selecting a model because the modeling software becomes part of the model and of the calibration process. These constraints may include user demographics, budget limitations, built-in sensitivity and optimization tools, and the preference of user friendliness vs. computational power. Furthermore, the multitude of closed proprietary software may present a disservice to the modeling community, creating model artifacts that originate somewhere deep inside the undocumented features of the software, and masking the underlying properties of the model.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Model comparison; Dynamic simulation; System-based models; Canopy surface energy balance

1. Introduction

Understanding complex ecological systems and the management of associated ecosystem resources requires an interdisciplinary approach by a variety of researchers, policy makers, resource managers, and stakeholders. The development of systems-based models for forecasting and decision making is far too complex for any one individual or isolated group of researchers, requiring collaborative model development, validation and assessment. Recently, a suite of general-purpose, systems-based, software tools have been introduced, specifically

* Corresponding author. Fax: +1 802 656 8446.

E-mail addresses: donna.rizzo@uvm.edu (D.M. Rizzo), paula.mouser@uvm.edu (P.J. Mouser), dwhit1885@yahoo.com (D.H. Whitney), charles.mark@uvm.edu (C.D. Mark), roger.d.magarey@aphis.usda.gov (R.D. Magarey), alexey.voinov@uvm.edu (A.A. Voinov).

¹ Fax: +1 802 656 8446.

² Fax: +1 919 513 7044.

³ Fax: +1 802 656 2995.

designed to support the development of dynamic ecological/economic/social systems models. These modeling environments (i.e., STELLA, Madonna, GoldSim, Simulink, etc.) are becoming increasingly popular, primarily because they provide an easy-to-use, graphical icon-based interface that can be understood by novice modelers (Costanza and Voinov, 2001), while model development using more traditional low- and high-level programming languages may require years of technical training. With the plethora of software packages available, each developed for different purposes and with individual sets of strengths and weaknesses, choosing a software package for a particular application can be a frustrating and time-consuming process, as there is little literature to rely upon when choosing an appropriate software package for a particular ecosystem modeling objective. Voinov (1999) offers an overview of several modeling packages with a limited description of their features. Seppelt and Richter (2005) have compared a number of packages with a focus on the mathematical rigor and precision of non-linear models. These sources do not highlight the general benefits or drawbacks of one software package over another for a particular application of interest. The situation is further complicated because we are dealing with graphical, icon-based commercial software intended for scientists with little programming experience, which by design specifically masks details and programming scripts from the user. From a research or management standpoint, understanding the advantages and limitations of these dynamic systems-based software packages may improve the accuracy of research results and increase the user audience. Unfortunately, the software is often pre-selected, either by the training of the user (e.g., users tend to use familiar software), or because the software has already been purchased. As a result, it is rare to find comparisons of these systems-based software tools applied to the same model or system.

As part of a simulation modeling course at University of Vermont, we translated a Surface Wetness Energy Balance (SWEB) model for grape canopy surface wetness (Magarey, 1999; Magarey et al., 2005a), into four dynamic systems software packages and compared the strengths and weaknesses based on 'novice' user interpretation. We deliberately selected a fairly simple model, expecting a simple and accurate implementation in each of the analyzed software packages. The task proved more difficult than anticipated, illuminating some inherent differences in the software packages and modeling results. This raises questions regarding the whole process of modeling with these various systems-based software tools. It appears that switching from one software package to another is not that straightforward. The software actually becomes part of the model and the embedded algorithms and methods become an important part of the calibration process. Therefore, when changing between software, developers must be conscious that additional adjustments and calibration may be necessary.

The overall goal of this study is to compare the benefits and limitations between four dynamic, systems-based software packages using a fairly simple Surface Wetness Energy Balance (SWEB) model to forecast relative canopy surface wetness. Three specific objectives associated with this goal

include: (1) translate the original SWEB model into four software packages: STELLA, Madonna, GoldSim, and Simulink and identify the most appropriate for representing surface wetness in crop canopies; (2) compare the benefits and limitations of the software packages to the original SWEB canopy model; and (3) perform a sensitivity analysis on five of the six input parameters.

2. Background

Crop management studies have shown that forecasting the risk of fungal and bacterial diseases in a crop canopy using disease risk models can reduce disease incidence and severity (Campbell and Madden, 1990; Funt et al., 1990), as well as decrease excess application of fungicides, which alters the soil ecosystem and accelerates resistance. Although a number of specific weather agricultural variables (i.e., temperature, relative humidity, net solar radiation, wind speed, and canopy surface wetness duration) are needed to accurately predict plant diseases, surface wetness duration (SWD) has long been acknowledged as a key parameter (Yarwood, 1978; Huber and Gillespie, 1992); and despite considerable attention from a diverse group of scientists, it continues to be the most difficult of these variables to quantify and forecast. SWD is defined as the sum of the duration over which the observed fraction of plant parts or organs that are wet in a canopy, exceeds some specified (plant specific) threshold (Magarey, 1999; Magarey et al., 2005a).

Experts in quantifying canopy surface wetness rely on visual observations, tactile observations, and/or a variety of sensor measurements on some number of individual leaves to reflect the spatial aspect of the proportion of leaves wet in an entire canopy. Despite meticulous measurements and various protocols, the sensors represent only a portion of the physics underlying the duration of the canopy wetness. As a result, the accuracy of these indirect sensor measurements should be validated with visual observations; however, this is often omitted because data collection is labor intensive. While other climatic variables have become standard measurements in weather station networks, the inconvenience and lack of a measurement standard for monitoring SWD at the local crop scale prevent existing disease risk models from being used with reliability.

An alternative to sensor measurements is the computational simulation (both statistical and physical) of surface wetness (Huber and Gillespie, 1992; Weiss, 1990) calibrated to a given crop via visual observations or sensor measurements. For excellent reviews of surface wetness models that have been validated under controlled or field conditions, see Huber and Gillespie (1992) and Magarey et al. (2005a). Despite the complexity of testing at the leaf scale and the lack of standards for defining and measuring surface wetness, it is anticipated that the next 10–20 years will see a shift in the estimation of surface wetness from systems based on in situ sensors to those based on simulation models and remote sensing (Magarey et al., 2005b).

2.1. SWEB model Version 0.28

Recently, a Surface Wetness Energy Balance (SWEB, Version 0.28) model has been developed as a potential theoretical standard for surface wetness measurement (Magarey, 1999; Magarey et al., 2005a). Fig. 1 provides a schematic of the

overall model structure. The model uses a physical approach to predict surface wetness using a canopy water budget and surface energy balance. The canopy water budget (Fig. 1a) is a function of the water storage (S) which, in turn, is a function of the volume of water condensing on the leaf as dew (D), the volume of water that is evaporating (E), and the intercepted

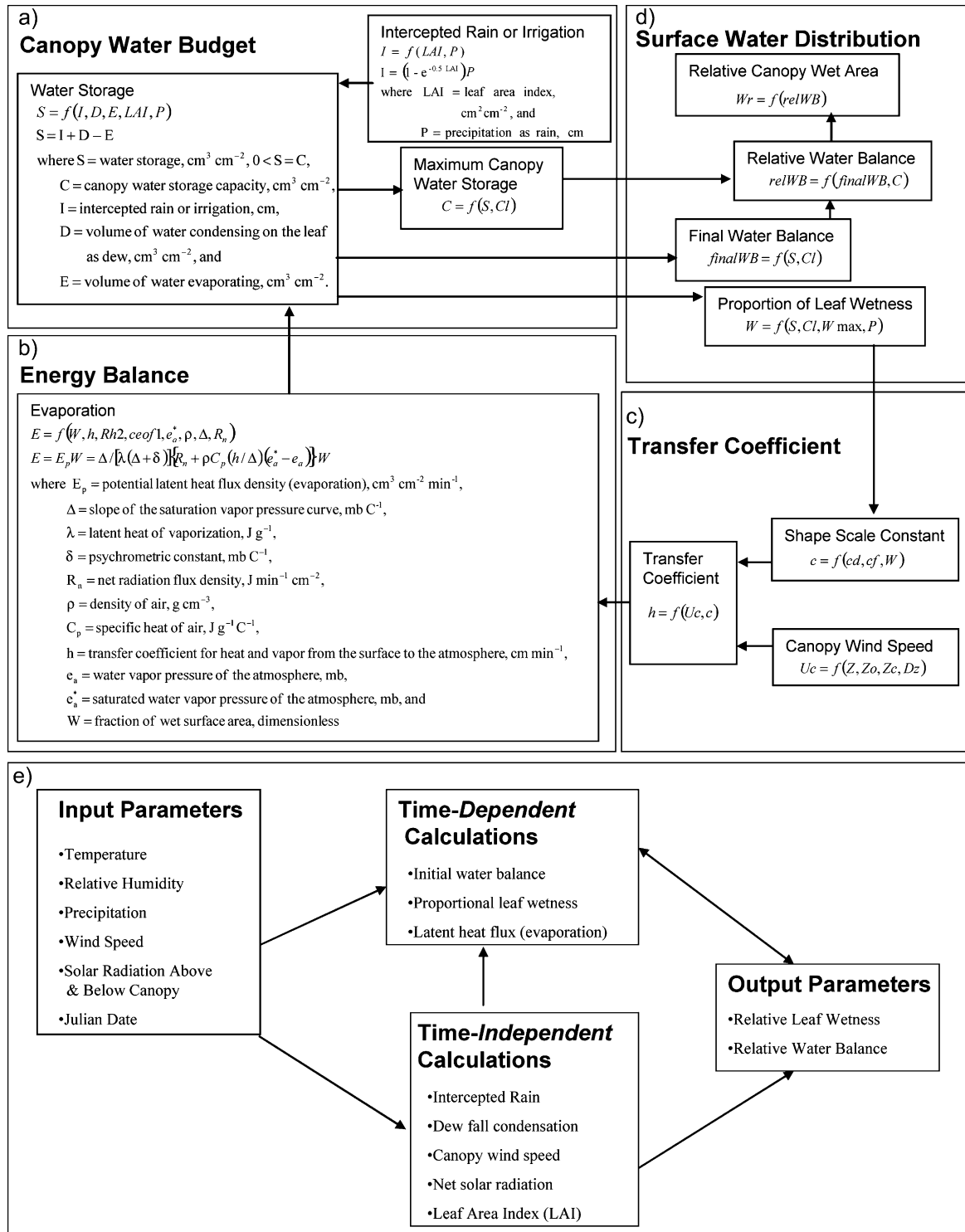


Fig. 1. Equations and model structure for the SWEB canopy model.

rain or irrigation (I). The interception term, I , selected for this application (see Fig. 1a) is the simple exponential formulation provided in Norman and Campbell (1983). It is a function of precipitation (P) and leaf area index (LAI); the latter being defined in broadleaf canopies as the one-sided green leaf area per unit ground area. The equation presented in Fig. 1b for the volume of water evaporating from the canopy surface is derived from the combination of surface energy balance and aerodynamic transport equations (Tanner and Fuchs, 1968). The equations and other details of the SWEB model are provided in the MATLAB code in Appendix A. More information regarding the derivation of equations within the SWEB model can be found in Magarey (1999) and Magarey et al. (2005a).

The SWEB model calculates a relative canopy leaf surface wetness (fraction of the canopy that is wet represented by a number ranging between 0 and 1) and a relative water balance from six main input parameters: temperature, relative humidity, precipitation, wind speed, net canopy radiation, and Julian date (Fig. 1e). It has a combination of both time-independent calculations and time-dependent differential equations. For example, the volume of intercepted rain is calculated using the date and the volume of precipitation associated with the current time step (t) without considering the volume of precipitation during the previous time step, while the volume evaporated from the leaf surface is based on the net radiation and relative humidity at the current time step (t) and the quantity present at the previous time step ($t - 1$). This combination of time-independent and time-dependent calculations presented challenges during model translation.

The SWEB model was calibrated and tested on an extensive data set from four grape cultivars growing at the Climatological Reference Station (NWS) in Geneva, New York (Magarey, 1999; *The Leaf Surface Wetness Duration*). Only one of the four field sites (Geneva_LO98) was used for the model comparisons and analyses presented in this work. This particular field site consisted of a 173-day growing season and included observations of surface wetness over six rain and eight dew events. For each moisture event, electronic sensors collected data (leaf wetness, temperature, relative humidity, wind speed and direction, precipitation, net radiation, soil heat flux, soil moisture and Julian date) along three vines within the canopy at five canopy positions and at 10-min intervals. Visual surface wetness measurements were collected in triplicate along the same 15 canopy positions at approximately 1-h intervals (totaling 45 visual measurements at any given collection time). These visual measurements provide validation for the physics-based SWEB model forecasts.

Fig. 2 is an example of the visual, sensor, and predicted SWEB canopy wet surface area, for a single moisture event (dew in this case). (Note: each point is the average over 15 canopy positions; and although the visual data collection is quite extensive for this particular moisture event, there is still a significant gap in the frequency of the resulting surface wetness measurements.) For this example moisture event, the shape of the SWEB model matches the sensor measurements

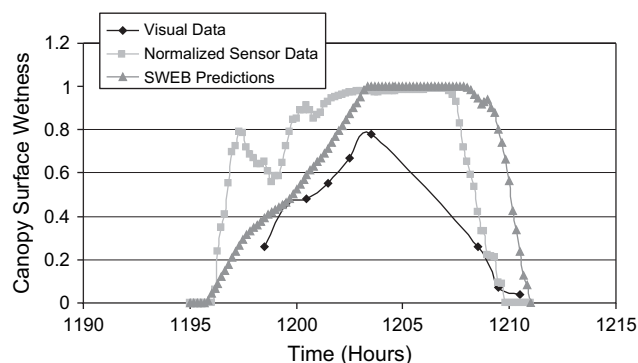


Fig. 2. Visual observations, sensor measurements, and SWEB model predictions of canopy surface wetness averaged over all vines and elevations within the canopy for one moisture event.

well. The timing of the SWEB model and sensor data are off-set. The SWEB model appears to lag behind both the sensor and visual data; the correction for this time offset will be discussed in the next section.

3. Methods

3.1. Model translation: SWEB Version 0.28

The original SWEB model was developed in Microsoft Excel (Microsoft Corporation, Redmond, WA) by leading experts in the area of leaf wetness who have worked on determining suitable protocols for estimating surface wetness in physical terms, as well as solutions to the lack of surface wetness standardization from both a measurement and a simulation perspective (Magarey, 1999; Magarey et al., 2001, 2005a,b). Excel was selected primarily because the experts had access to this software and were not familiar with a more traditional programming language. To gain an understanding of the SWEB model prior to translation into dynamic systems models, we first coded it into a high-level programming language (MATLAB, V6.1, R12.1). This conversion was relatively straightforward; equations could be translated directly from excel to MATLAB, resulting in leaf wetness estimates identical between excel and MATLAB to five decimal places (Table 1 and Fig. 4).

MATLAB (Mathworks Inc., Natick, MA) is both a low- and high-level programming language that markets to the science

Table 1

Root mean square (RMS) error values between software packages and the Excel SWEB model

Software package	RMS (Excel)
MATLAB	0.00006
STELLA	0.1307
Madonna	0.1297
GoldSim (Flows)	0.1613
GoldSim (Expressions)	0.0908
Simulink	0.0891

and engineering disciplines. It is unique when compared to computer languages such as FORTRAN, BASIC, C, C++, Pascal, and Java, as it uses a combination of the two major types of programming language translators – compilers and interpreters. This interactive programming environment does not require the formal compilation, linking/loading, and execution associated with other high-level computer languages. MATLAB models are scripted in m-files that are then compiled (part-way) into p-code (pseudo code) the first time they are executed in a given session. The p-code is then interpreted, resulting in a much faster development environment because most of the compilation has already been performed. MATLAB is coupled with extensive visualization capabilities for generating graphs and the ability to access and process external files with large quantities of data. While MATLAB is computationally very powerful and flexible, it is a programming language and, as a result, requires a reasonable investment of time and training.

The systems-based software packages chosen to represent the SWEB model, including STELLA, Madonna, GoldSim, and Simulink, spanned a range of dynamic systems model development tools. While programming languages such as MATLAB often require more mathematical and programming experience, dynamic systems models are developed by creating a symbolic representation of the system with graphical icons and imbedded equations. The intricacies of complex systems are exposed through the use of distinctly different and descriptive icons, with flow lines that indicate interactions between parameters. For stock and flow models (STELLA, Madonna, GoldSim with Flows), we first translated the SWEB model to STELLA and used these equations as the basis for the Madonna and GoldSim. Madonna has the capability to directly import STELLA models, which was used as one method of minimizing translation errors between the dynamic software packages. The equation-based models (Simulink, GoldSim with Expressions) were developed directly from the SWEB model equations. Brief descriptions of the four software packages are provided in Section 3.2.

3.2. Simulation modeling software packages

3.2.1. STELLA Version 5.0

The STELLA (High Performance Systems, Inc., Lebanon, NH) dynamic systems software package is an icon-based simulation tool that uses differential equations represented as stocks and flows. This software has been highly used for understanding population dynamics and economic fluxes. Stocks represent a balance unit that changes with each time step; flows represent a positive or negative change of flux; converters represent input parameters; and arrows represent mathematical relationships between the elements. Simple graphing and table features allow the user an easy visual or quantitative method for checking output values. Three numerical integration methods are available in STELLA: Euler, Runge-Kutta 2 and Runge-Kutta 4. Equations are easily accessed by clicking on the icons or by manually altering mathematical

relationships in the equation editor. Tables and equations can be quickly exported as text files to be used in other software programs.

3.2.2. Berkeley Madonna Version 7.0.2

Berkeley Madonna (Macey and Oster, Berkeley, CA) is a dynamic systems software package developed under National Science Foundation (NSF) and National Institute of Health (NIH) sponsorship. Equations may be drawn graphically using the flowchart editor or formulated manually in the equation editor. Although it can be applied to a variety of ecological and systems-based problems, it would be particularly useful in mechanical engineering and chemistry applications, as illustrated by the accompanying tutorials, batch runs, modules, and sensitivity options. Madonna allows users to have direct access to all equations, time steps, and constants. The user may select an integration method from one of four built-in functions (Euler, Runge-Kutta 2, Runge-Kutta 4, Rosenbrock) or design a custom time integration method within the software. Madonna also allows for quick data input and output through text files loading as vectors or matrices. The graphical user interface is easy to use and allows easy comparison of model results to external data. Optimization and sensitivity analyses are built-in options in the Madonna software and are extremely useful when evaluating the relative importance of model parameters.

3.2.3. GoldSim Pro Version 7.51.100

GoldSim (GoldSim Technology Group, Redmond, WA) is a flexible software modeling package with applications ranging from simple static and deterministic systems, to complex systems with unpredictable behavior and high degrees of uncertainty. There are many built-in features that quantitatively address uncertainty associated with modeled processes, parameters and future events such as the discrete and triggered event tools. The Euler numerical integration method is used for solving differential equations in the GoldSim simulator. Models are created in GoldSim by constructing an influence diagram using built-in elements that are represented by graphical icons or by programming equations. The programming capabilities of GoldSim are much more flexible and/or advanced than STELLA and Madonna. More complex models are assembled by arranging the system in a hierarchical, modular manner. GoldSim may be dynamically linked to external programs or spreadsheets, a useful feature for importing large data sets. In addition, GoldSim is ‘dimensionally aware’, internally converting output to user-specified units, and thereby minimizing error. For example, if the dimensional units for time, distance, and velocity have been specified as minutes, kilometers, and miles-per-hour, respectively, the user can specify output units for travel time in minutes, and GoldSim will complete all necessary unit conversions.

3.2.4. Simulink Version 4.1.2.4

Simulink (Mathworks Inc., Natick, MA) is a block diagram visual modeling tool that is an add-on package to MATLAB. It

uses the same MATLAB language and functions, but rather than writing lines of code, models are developed in an icon-based user interface that enables a conceptual diagram of the modeled system, similar to electrical circuits. Icons are added to the model using a library of available graphics. Groups of variables and operations are built as sub-modules that transform complicated models into a comprehensible diagram where relationships between system components can be readily observed. Many integration methods are available for Simulink including Euler, Runge-Kutta, Gear, Rosenbrock, as well as the ability to manually program alternative methods. When data are not available for a particular time step, Simulink automatically interpolates a value for the missing time step. Results are easily graphed using the graph block feature connected to a variable flow path.

4. Results

4.1. Model translation

When translating the SWEB model into the four dynamic systems software packages, two interesting challenges surfaced. First, the translation of the SWEB model into each software package required a slightly different model formulation due to the individual program structure and syntaxes. Less room for interpretation error was encountered using the equation-based models (e.g., MATLAB, Simulink, and GoldSim with Expressions), in contrast, the stock and flow-based models relied heavily on the programmer's conceptual understanding of the system (e.g., STELLA, Madonna, GoldSim with Flows). The combination of time-independent and time-dependent calculations in the SWEB model (Fig. 1e) resulted in large accumulation errors for uncalibrated stock and flow models. In general, software packages that enabled the use of programming language to code expressions rather than stocks and flows to describe temporal difference equations produced results that more closely resembled the original canopy surface wetness model.

Secondly, we observed that translating the SWEB model into a dynamic systems-based model (STELLA, Madonna, GoldSim with Flows) provided a better conceptual understanding of the ecological system than the language-based programs (MATLAB, Simulink, GoldSim with Expressions). If an objective of model development is to gain a better understanding of a system, the capability of a software package to graphically represent the model structure should not be overlooked. Scientists with little programming experience can begin developing, calibrating and testing models almost immediately. This is an important consideration if the model is for educational purposes or for fostering a collaborative model development rather than for accuracy or prediction.

Additionally, other interesting discussion points surfaced between the four different software packages. Although many of the packages had functions and capabilities that overlapped, we found a broad range of user friendliness and mathematical power between the systems-based software

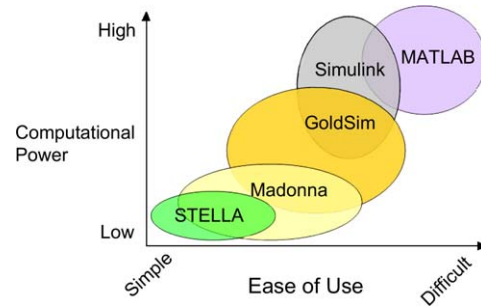


Fig. 3. Conceptual relationship between software packages for ease of use and computational power.

(Fig. 3). These comments are summarized in tabular form (Table 2).

STELLA was by far the most user-friendly piece of software with cartoon-like icons and animated stocks and flows. However, we found it to be somewhat limited when interacting with external data sets and equations that are not strictly calculating a mass-balance. For example, one of the most tedious tasks associated with the SWEB model development in STELLA was data input/output. STELLA allows a maximum of 1500 data points to be manually imported or copied and pasted from an external file. Additionally, circular references were often encountered in STELLA when converters involved parameters calculated from previous time steps. The construction of additional stocks and flows in an attempt to circumvent errors resulted in compounded time-step errors.

The Madonna software is, in general, very user-friendly and similar to STELLA in its presentation as an icon-based editor. Madonna is capable of importing equations directly from STELLA; this is an attractive feature, as many models have been historically developed in STELLA. In addition, Madonna performed quick and easy sensitivity analyses and optimization of parameters with manual sliders. Graphical displays were updated immediately with the response of the model to parameter changes.

The GoldSim software has a less user-friendly interface than STELLA or Madonna, but includes the capability to develop models using built-in icons or an equation editor. In addition, the GoldSim software has expansion capabilities (i.e., a contaminant transport package, dashboard authoring, distributed processing) and may be linked directly to external files such as Microsoft Excel for easy data access. To circumvent circular references, GoldSim allows the user to create a new parameter using the built-in delay function. The GoldSim software had slightly slower run-times when the model was linked to an external data set and equations could not be created in this software without clicking on individual icons.

In the stock and flow software packages (STELLA, Madonna, and GoldSim with Flows), one major limitation is the user's lack of control over computations taking place behind the scenes, particularly, discretizations in time. For example, when a model developer wants to calculate parameters using values that should be lagged in time, circular dependencies often appear in dynamic software packages. This issue can

Table 2
Comparison of five modeling software packages used to simulate the SWEB canopy witness model

Software package	Cost ^a	Data input formats	Operating system (Interface)	Available integration method	Sensitivity analysis	Optimization
MATLAB V. 7		Excel, Text, Database, Realtime	Windows/Macintosh (No Graphical Interface)	Euler, Gear, Runge-Kutta, Rosenbrock, Manual	Manual	Manual through specialized functions
Student version	\$99.00					
Academic price	\$500.00					
Commercial version	\$1900.00					
STELLA V. 8.1		Excel, Text	Windows/Macintosh (Graphical Interface)	Euler, Runge-Kutta	Built-in	Manual
Student version	\$129.00					
Academic price	\$649.00					
Commercial version	\$1899.00					
Berkeley Madonna V. 8.0		Excel, Text	Windows (Graphical Interface)	Euler, Runge-Kutta, Rosenbrock, Manual	Built-in	Built-in
Student version	\$99.00					
Academic price	N/A					
Commercial version	\$299.00					
GoldSim Pro V. 9		Excel, Text	Windows (Graphical Interface)	Euler	Manual	Manual
Student version	Free					
Academic price	\$950.00					
Commercial version	\$3950.00					
Simulink V. 6.1		Excel, Text, Database, Realtime	Windows/Macintosh (Graphical Interface)	Euler, Gear, Runge-Kutta, Rosenbrock, Manual	Manual	Manual through specialized functions
Student version	\$99.00					
Academic price	\$500.00					
Commercial version	\$2800.00					

^a Listed prices are for most recently available versions as of July 2005.

be addressed through model refinement in the form of additional variables or delays that introduce redundancy and error.

Simulink was not as easy to learn as the systems-based software packages; it is icon-based, but relies on a programmer to provide additional model structure that is automatically created in dynamic systems software such as STELLA, Madonna, and GoldSim. For example, the differential equations developed in one step using dynamic systems-based models may take up to three user steps to discretize time, overlay the function, and apply the time sequence in Simulink. The additional steps give the user more control over specific aspects in the modeling process, which may explain why errors between Simulink and the SWEB model are lower than other software packages (Table 1). Circular references were not encountered in Simulink and the model may be directly linked to external files for quick and easy importation of data.

4.2. Ability to translate the SWEB model

The ability to translate the SWEB model into the four modeling packages was assessed using two measures: absolute and root mean square error. Absolute error was used as a measure of precision between the SWEB model and the systems software packages. It was calculated as the square root of the difference between the estimates of canopy surface witness produced by the SWEB model, X_i , and the systems-based software model of interest, Y_i , for each of the 3970 hourly time steps, i , using $\text{AbsError}_i = \sqrt{(X_i - Y_i)^2}$. We found the software packages produced statistically significant absolute errors when compared to the SWEB model (Kruskal–Wallis Test, $p < 0.0001$) (Fig. 4). GoldSim with Expressions had the lowest absolute error; while

GoldSim with Flows had the highest absolute error. These absolute errors from GoldSim with Expressions and GoldSim with Flows were significantly different from each other and all other models (Tukey's Pairwise Comparison, $p = 0.01$). STELLA, Madonna, and Simulink had similar absolute errors that were not significantly different from each other (Tukey's Pairwise Comparison, $p = 0.01$).

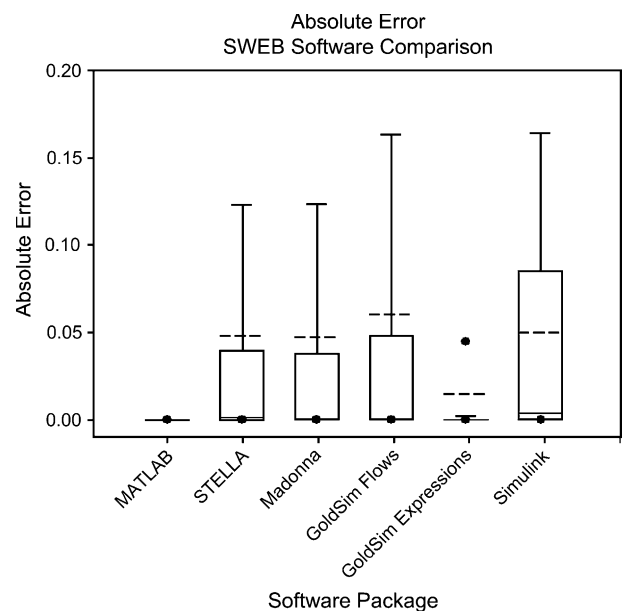


Fig. 4. Box plot of individual absolute errors between the SWEB Excel model and software package. Boxes represent the 25th, 50th, and 75th percentile, Whiskers represent the 5th and 95th percentiles, circles represent outliers, and dotted lines represent the means.

The second assessment measure, the root-mean-square (RMS) error, is an overall measure of accuracy between the SWEB model and the systems-based software packages. It was calculated for each of the models over all 3970 hourly time steps as $RMS = \sqrt{(\sum_{i=1}^n (x_i - y_i)^2) / n}$ where, n = number of time steps. The RMS errors between the SWEB model and the software packages are shown in Table 1. Simulink and GoldSim with Expressions had the lowest RMS error values. These two models were expression based yet used icons to represent relationships that may have resulted in slight translation errors. The stock and flow-based software packages, STELLA, Madonna, and GoldSim with Flows had the highest RMS values, which may be a result of interpretation and translation error from the SWEB model. It is interesting to note the slight difference in RMS error between STELLA and Madonna. The STELLA equations were imported directly into Madonna. As a result, we expected exact results between the two models using the same equations; however, this was not the case with Madonna having slightly larger RMS error than STELLA (Table 1).

One explanation for the difference in RMS errors between icon- and expression-based models is the user's control of the timing over which calculations are made. Errors produced by the numerical approximations and the timing of these errors could have resulted in higher RMS error values for the stock and flow-based software packages. A second reason for the difference in RMS errors between models is related to the developer's choice in addressing circular dependencies in each of the software packages and the selection of time steps.

An overall RMS error does not describe all aspects of model performance. In addition to errors between overall model performance, we were interested in time-dependent error or performance (i.e., within a particular wetting or drying event). To assess this, a cumulative RMS error and an RMS error value averaged over a moving window of 200 time intervals (h) were calculated.

Fig. 5 shows the two RMS error statistics versus time for each software package (cumulative RMS error values are shown under the title). The time period between 500 and 800 hours yielded a low time averaged RMS error value. This time period corresponds with a period of stable and sustained relative leaf wetness values at or near full saturation. It should be noted that MATLAB (Fig. 5a) is plotted on a much smaller scale than the other software packages (Fig. 5b–f). In general, the shape of all RMS error values over time is similar. The expression models (Simulink and GoldSim with Expressions) have RMS values that are approximately half as large, on average, as the stock and flow models (STELLA, Madonna, GoldSim with Flows). The change in the cumulative RMS error can indicate the stability of the model over time whereas a cumulative RMS error value that remains constant or decreases over time indicates a model in which the performance does not deteriorate over time. All translated models had cumulative RMS errors that remained constant or decreased over time (Fig. 5a–f). This is especially significant for the stock and flow-based models since they are more susceptible to accumulating errors than expression-based models. Note the largest variance in error for the stock and flow models occurs in the first 1000 hourly time

steps, which are approximately three times that of the expression-based models. After the first 1000 time steps, the cumulative and moving window average of the stock and flow models approaches that of the expression-based models (an RMS error of approximately 0.1).

4.3. Sensitivity analysis

Sensitivity analysis is performed to better understand how changes in model parameters affect output values and stability. In this project, we were interested in how noise from the microclimate sensor data may affect canopy surface wetness predictions. One measure of sensitivity is to create systematic errors in one or more input parameters and observe how the system responds. In this work, five input variables: temperature, wind speed, precipitation, relative humidity, and net canopy radiation were each subjected to random noise of $\pm 10\%$. Random noise was added to one of the five input parameters while each of the other four parameters were held constant. The sixth sensitivity run incorporated $\pm 10\%$ random noise to all five input parameters. Canopy surface wetness was the response (output) variable used for comparison. Performance for the sensitivity analysis was measured by comparing each model's original performance against its performance with $\pm 10\%$ random error, rather than a comparison between software packages. As a result, the correlation parameter R^2 was used as an internal performance for relative leaf wetness measurements with and without noise, respectively.

Table 3 compares the resulting R^2 values for the various software packages with lower R^2 values indicating greater sensitivity to the random errors and worse performance. MATLAB, Simulink and Madonna packages performed well. It is interesting to note that Madonna performed the best of the dynamic systems models, including STELLA, the software package from which it was originally translated. Both Madonna and STELLA models used the same equations and structure, yet Madonna was much less sensitive to noise. The GoldSim models in general were much more sensitive to noise than the other modeling software packages.

The sensitivity analysis also helped quantify the influence of each input parameter in the model. Net canopy radiation and relative humidity were the two most sensitive variables for all software packages, as indicated by the lowest average R^2 values. From an application standpoint, this indicates that noise in these two variables causes the largest model error, thus more resources (time, quantity, and quality of sensors) should be allocated to improving the accuracy of the data collection, reducing the measurement error and insuring correct sensor calibration. Another interesting comparison occurs between the R^2 values for MATLAB and Madonna. MATLAB is the least sensitive to changes in net canopy radiation, but the reverse is true for the relative humidity parameter. While the difference in ranked sensitivity between each model's parameters is slight in this study, it may be a significant consideration in software selection when the user has some prior knowledge of the expense (labor or monetary)

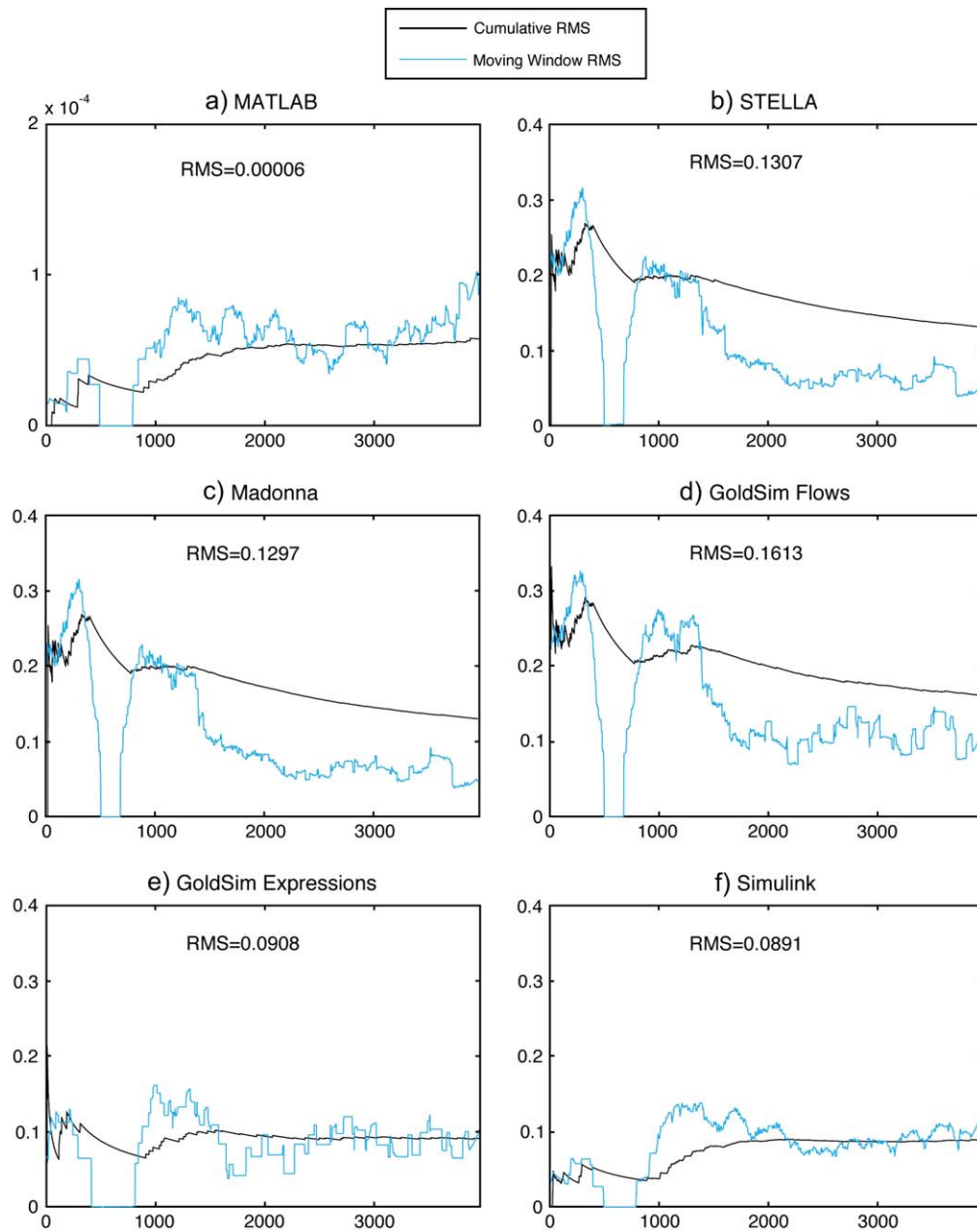


Fig. 5. Variations of the root-mean-square (RMS) errors over 3970 hourly time steps. The dark line represents the cumulative RMS error while the lighter gray line represents the 200 time step moving interval.

associated with obtaining input parameter data (e.g., net canopy radiation is a more costly and labor intensive expense than relative humidity and temperature, which is important for this particular application).

5. Discussion

Two of the primary objectives of this study were to (1) identify an ‘appropriate’ software package for representing the conceptual energy balance model of surface wetness in crop canopies and (2) compare the benefits and limitations between four dynamic, systems-based software packages. As is

frequently the case, the choice of software depends on the application, since many factors must be considered by the stakeholders when selecting a model. These constraints may include user demographics, phase of development, budget limitations, built-in sensitivity and optimization tools, and the preference of user friendliness versus computational power. The results of this study suggest that dynamic systems tools provide a good conceptual understanding of the SWEB model, without necessarily sacrificing accuracy. However, understanding the underlying mathematics behind stock and flow models may be difficult at best for novice modelers. We recommend that models like SWEB be developed initially in a dynamic software package until researchers are confident with

Table 3
 R^2 values resulting from a sensitivity analysis conducted by changing five input parameters (wind speed, temperature, radiation, precipitation, and relative humidity) by $\pm 10\%$

Model	Wind speed	Temperature	Radiation	Precipitation	Relative humidity	All 5	Average
MATLAB	1.0000	1.0000	0.9939	0.9999	0.9666	0.9617	0.987
STELLA	0.9999	0.9999	0.9355	0.9989	0.9562	0.8956	0.964
Madonna	1.0000	1.0000	0.9772	0.9999	0.9767	0.9547	0.985
GoldSim flows	0.8221	0.8226	0.8016	0.8223	0.7690	0.7494	0.798
GoldSim expressions	0.8519	0.8522	0.8435	0.8519	0.7744	0.7671	0.824
Simulink	1.0000	1.0000	0.9933	1.0000	0.9712	0.9649	0.988
Average	0.946	0.946	0.924	0.945	0.902	0.882	

the selection of input parameters and variable ranges. The dynamic model can then be translated into an expression-based model to improve accuracy of results.

The error associated with a given solution is directly related to the integration method used. The Euler method was used in all software packages for this comparison, however, a ‘basic’ and ‘modified’ Euler method exists and may not have been consistent between packages. We noted that changing the time step size significantly affected output results, particularly in the dynamic software packages. Thus, it is possible that the dynamic packages were (1) not consistent with the power order and number of significant figures retained during integration or (2) used a variation of the Euler integration method. A good example is the difference in model output and sensitivity results between STELLA and Madonna. The same syntax, input files, and parameter values were copied directly between the packages, yet output results differed slightly.

For this study, models were selected that spanned a range between dynamic system simulators and expression-based computational tools (see Table 2). The SWEB model developed in Excel was translated to MATLAB with minimal error (RMS = 0.00006), yet translation to dynamic software packages ultimately resulted in different types of translation errors. The dynamic system software SWEB models were not as accurate, although similar performance trends were observed between model pairings such as GoldSim with Expressions and Simulink, as well as STELLA, Madonna, and GoldSim with Flows. A sensitivity analysis indicated that all models were most sensitive to net canopy solar radiation and relative humidity. Simulink and Madonna exhibited the least amount of sensitivity to input parameter variability. STELLA and GoldSim models were significantly affected when all five of the input parameters were varied, while the GoldSim models were the most sensitive to noise in individual input parameters.

Our major conclusion from this study is that switching from one systems-based software package to another can be challenging at best, even for very simple models. The modeling software becomes part of the model and of the calibration process. The performance becomes intimately connected with the choice of software package used to implement and analyze the model. Therefore, the reimplementations in different software is not similar to switching from one programming

language to another. Each package has its own conventions and restrictions that become an inherent part of the model implementation and are best taken into account in the early stage of model development. A mechanistic transition from one package to another results in surprises, discrepancies, errors and behavior that may be difficult to explain and mitigate. The multitude of closed proprietary software that has hit the market lately may actually present a disservice to the modeling community, creating a multitude of model artifacts that originate somewhere deep inside the undocumented features of the software, and may hide or distort the actual properties of the methods and formalizations of the model that are reported and published. When programming languages are used to implement models, the models are separate from the computer implementation, and it matters less what language is selected for model development. Programming languages do not, however, provide a conceptual understanding of the system, or visual interpretation of the interactions between parameters without studying the code and individual equations.

Acknowledgements

Special thanks goes to Alaina Dickenson for development of Fig. 1 and Appendix A. Work was funded in part by a Vermont NSF EPSCoR Graduate Research Assistantship #524474.

Appendix A

The program is implemented for MATLAB Version 6.1 Release 12.0 and requires no toolboxes. This program is copyrighted and permission is granted for the use of this program for individual study. Commercial use is explicitly not allowed. The authors and the University of Vermont provide this code on an “as is” basis and accept no responsibility for errors, mistakes, or misrepresentations that may occur as a result of its use. Our use of MATLAB is for product identification purposes and does not represent an endorsement of the product.

Constants

```

Zc=1.8;
Dz=1.2;
Zo=0.18;
Z=3;
Cp=0.286;
pair=0.0012;
Wmax=0.5;
cf=4.1;
cd=18.2;
LL=4.5;
radE=0;
finalWB=zeros(length(Uz),1);
Cl=0.02;

```

Model

```

LAI = 0.161 + 2.0929. / (1 + exp (-0.1951.* (day-175.18)));
C = LAI.*Cl;
Uc = Uz.*(log ((Zc-Dz)/Zo)/log ((Z-Dz)/Zo));
Ucl = Uc.*(1 + 1.3*(1-1.05/1.8))^2;
CNR = (ANR-BNR).*0.00002388.*60.*0.5;
(ea*) = 6.11.*exp (5327*(1/273-1./ (T+273)));
Δ = (ea*).*(6790.4985./(T+273).^2-5.02808./ (T+273));
ΔP = Δ + 0.66;
Rn = (CNR.* Δ). / ΔP;
coef1 = (Cp.* ρ). / ΔP;
Rh2 = (1-Rh/100);
I = (P. / 10).*(1-exp (-0.5.*LAI));

for loop = 1:length(Rn);
  if Rn (loop,1) < 0;
    D(loop,1) = -Rn (loop,1).*(60/583);
  else
    D(loop,1) = 0;
  end
end

for loop = 1:length(Rn);
  if loop == 1;
    initialWB(loop,1) = max(0,min((I(loop,1)+ D(loop,1)+ 0),C(loop,1)));
    W(loop,1) = Wmax .* ((0./C(loop,1)).^(.6667));
  else
    initialWB(loop,1) = max(0,min((I(loop,1)+ D(loop,1)+ finalWB(loop-1,1)), C(loop,1)));
    W(loop,1) = Wmax .* ((initialWB(loop-1,1)./C(loop,1)).^(.6667));
  end

  c(loop,1) = ((1-W(loop,1)).*cd + (W(loop,1).*cf));
  h(loop,1) = c(loop,1).*sqrt((Ucl(loop,1).*6000)./(LL));
  E(loop,1) = (Rh2(loop,1).*coef1(loop,1).*(ea*)(loop,1)/583).*h(loop,1).*W(loop,1);
  volE(loop,1) = E(loop,1).*60;
  minusWB(loop,1) = radE + volE(loop,1);
  if loop == 1;
    finalWB(loop,1) = max(0,min(0+minusWB(loop,1),C(loop,1)));
  else
    finalWB(loop,1) = max(0,min(initialWB(loop-1,1)+ minusWB(loop,1),C(loop,1)));
  end
end

relWB = finalWB./C;
Wr = relWB.^(0.6667);

```

References

- Campbell, C.L., Madden, L.V., 1990. Introduction to Plant Disease Epidemiology. John Wiley & Sons, New York.
- Costanza, R., Voinov, A., 2001. Modeling ecological and economic systems with STELLA: Part III. Ecol. Model. 143 (1–2), 1–7.
- Funt, R.C., Ellis, M.A., Madden, L.V., 1990. Economic analysis of protectant and disease-forecast-based fungicide spray programs for control of apple scab and grape black rot in Ohio. Plant Dis. 74 (9).
- Huber, L., Gillespie, T.J., 1992. Modeling leaf wetness in relation to plant disease epidemiology. Annu. Rev. Phytopathol. 30, 553–577.

- Magarey, R.D., Seem, R.C., Weiss, A., Gillespie, T.J., Huber, L., 2005a. Estimating surface wetness on plants. In: Hatfield, J.L., Baker, J.M., Viney, M.K. (Eds.), *Micrometeorology in Agricultural Systems*, Agronomy Monograph No. 47, in a series of Agronomy. American Society of Agronomy, Crop Science Society of America, Soil Science Society of America: Madison, WI.
- Magarey, R.D., Russo, J.M., Seem, R.C., Gadoury, D.M., 2005b. Surface wetness duration under controlled environmental conditions. *Agric. For. Meteorol.*, 128 (1–2), 111–122.
- Magarey, R.D., 1999. A theoretical standard for estimation of surface wetness duration in grape. PhD Dissertation, Cornell University, Ithaca, NY.
- Magarey, R.D., Seem, R.C., Russo, J.M., Zack, J.W., Waight, K.T., Travis, J.W., Oudemans, P.V., 2001. Site-specific weather information without on-site sensors. *Plant Dis.* 85, 1216–1226.
- Norman, J.M., Campbell, G., 1983. Application of a plant environment model to problems in the environment. *Adv. Irrig.* 2, 155–188.
- Seppelt, R., Richter, O., 2005. “It was an artifact not the result”: a note on systems dynamic model development tools. *Environ. Model. Softw.*, 20 (20), 1543–1548.
- Tanner, C.B., Fuchs, M., 1968. Evaporation from unsaturated surfaces: a generalized combination method. *J. Geophys. Res.* 73, 1299–1304.
- Voinov, A., 1999. Simulation Modeling. <<http://www.likbez.com/AV/Simmod.html>>.
- Weiss, A., 1990. Leaf wetness: measurements and models. *Remote Sens. Rev.* 5, 215–224.
- Yarwood, C.E., 1978. Water and the infection process. In: Kozlowski, T.T. (Ed.), *Water and Plant Disease. Water Deficits and Plant Growth*, vol. V. Academic Press, New York, NY, pp. 141–173.
- The Leaf Surface Wetness Duration. <<http://www.nysaes.cornell.edu/pp/faculty/seem/magarey/leafwet/lwtitle2.html>>.