26th International Congress on Modelling and Simulation, Adelaide, SA, Australia, 30 Nov–4 Dec 2025
mssanz.org.au/modsim2025

A state-variable based approach to model integration in a horticultural modelling platform

ST Bell [a], J Zhu [c], R Cichota [a], H Jenkins [a], D Liu [a], OA Chuang, J Zhang [b], C Van Houtte [d], J Tang [b], C Cerecke [b] and HT Lin [d]

[a] *New Zealand Institute for Bioeconomy Science Limited, Lincoln, Canterbury, New Zealand*
[b] *New Zealand Institute for Bioeconomy Science Limited, Auckland 1142, New Zealand*
[c] *New Zealand Institute for Bioeconomy Science Limited, Blenheim 7240, New Zealand*
[d] *New Zealand Institute for Bioeconomy Science Limited, Palmerston North 4442, New Zealand*
Email: stephen.bell@plantandfood.co.nz

Abstract: Horticultural modelling is an interdisciplinary field that uses biophysical, statistical, and mathematical models to understand and predict plant growth, development and the relationships between plants and their environment. Larger simulations may draw on expertise from different science domains and increasingly rely on diverse models developed in different frameworks, programming languages, and operating at multiple spatial, temporal, and physiological scales. Integrating these models into a coherent simulation is challenging due to differences in data formats, variable naming, and underlying modelling paradigms. We present a state-variable based integration approach that abstracts model state as shared variables, referenced through tuple-based identifiers for consistent grouping and namespace management. A memory-based cache (simulation layer) manages the exchange of these state variables, while a collective simulation framework schedules model execution and handles interdependencies through synchronisation points. Using this method, once models are adapted into a collective simulation, they can be linked with minimal rework and reused across multiple simulation contexts. We demonstrate the approach by integrating FruitCropXL, a functional–structural fruit-crop model, with a soil water balance model, enabling direct communication of water fluxes and shared micro-climate inputs. This integration strategy lays the foundation for digital horticultural twins by providing a reusable and scalable mechanism for interoperability between independently developed biophysical models.

Keywords: *Integrated modelling, state-variable, biophysical models, model-to-model communication*

1. INTRODUCTION

In relation to horticultural modelling, the *crops in silico* initiative (Marshall-Colon and Long et al., 2017) provides an extensive review of the model integration challenges and range of modelling frameworks available. Common approaches to model interoperability include standardised data exchange formats, pair-wise connected models, and frameworks supporting groups of similar models. In many cases these frameworks have been started by building on isolated models and extended to support small groups of interconnected models. The Crop2ML initiative (Midingoyi et al., 2021) describes two model interoperability paradigms, the first based on explicit import-export linkages between small groups of models, and the second where model linkages are mediated via a centralised component (a standardised modelling language).

A central feature of our model integration approach is the use of state variables as high-level abstractions, enabling flexible communication between models developed under diverse biophysical modelling paradigms. To ensure naming consistency and modularity, state variables are referenced using tuple-based identifiers, which provide structured grouping and namespace management. While models currently use native variable names, our platform is designed to incorporate a remapping system (netlist), inspired by electronics and semiconductor design techniques, to align variable names across models within multi-model simulations. The netlist functions as a directed graph that specifies all connections between shared (or public) state variables across models. This mechanism provides a systematic way to align variable names and manage model interactions, forming the basis for the model-to-model communication concepts described in later sections.

2. BACKGROUND

The motivation for this work is the development of a modelling and simulation platform for the Digital Horticulture Systems (DHS) (Mawson et al., 2023; Stanley et al, 2024). This platform is essential for enabling interoperability among biophysical models developed across multiple science domains and research teams. The

existing models in the platform were created in different modelling frameworks, implemented in different languages, and have data storage formats that are often unique to each model. Moreover, these models operate at a wide range of scales, from orchard ecosystems to individual trees and organs, while also representing key physiological processes. These scales are complemented by cellular and molecular models, which help interpret genetic information and explain the material properties of crops. Two examples of models in use within DHS are:

- A soil-water-balance model adapted from APSIMX - a well-established horticultural modelling framework operating on a daily time step (Cichota et al., 2021; Holdsworth et al., 2018), and
- FruitCropXL – a functional structural plant model operating at hourly time steps, based on eXtended L-systems and GroIMP growth grammar techniques (Zhu et al., 2021; Kniemeyer and Kurth, 2008).

Other modelling paradigms are also represented within the DHS platform. These include statistical models applied in genetics, biosecurity, and bio-protection, as well as agent-based models relevant to orchard management, pest dynamics, and pollinator interactions. While these approaches are important components of the broader platform, their use is beyond the scope of this paper, and we therefore focus our demonstrations on the integration of the above two models.

3. MODEL INTEGRATION

3.1. Building the approach

A common aspect of the initial DHS models is that they are biophysical dynamic systems models, describing a real-world system. Although their formal implementations differ, each model fundamentally expresses the state of biophysical properties as variables. In this paper, we use the term state variable to refer to this general concept of a model variable representing a state of a biophysical system variable.

Despite some general conventions, we have found it useful to consider a high level of abstraction for *state-variables* where they may be any type of variable describing the system, such as *quantities*, *concentrations*, *fluxes*, etc. This is because, in a software context, our interoperability layer does not need to mediate in decisions such as how to partition and structure variable use in individual models – it is the role of the model to do this. At this point we can summarise that a dynamic systems model of a biophysical system comprises a set of state variables that describe the system. Such variables are updated in a series of steps using a corresponding set of update equations during the model simulation.

Time evolving Markov-processes can be described as a chain or series of states, where each successive state is determined by a statistical calculation using only the previous state (Brémaud, 1999). It turns out that this structure for updating the *state-variables* in a Markov-process model can also fit within the general structure for updating *state-variables* in a biophysical systems model. This rather interesting insight allows us to propose that the *state-variable* based model-to-model communication interface, is suitable for both mechanistic dynamic systems, and Markov-process based statistical models of biophysical systems. Consequently, our abstraction should also be compatible with a wide range of modelling approaches of biophysical systems and was adopted as a key approach to explore in our modelling platform.

It is readily apparent that there should only be one computing process that calculates the final value of a state-variable at a given step, although it is common for several subprocesses to calculate parts that are then combined. For example, multiple *fluxes* may be aggregated, and the aggregate flux used to update a corresponding *quantity state-variable.*

The time scale is an important aspect for biophysical simulations and may affect many underlying modelling processes. For example, minute or hourly updates are often used when modelling processes that vary over a diurnal cycle (Zhu et al., 2021), and simulations running with daily updates are common in other models that do not need a high level of detail or when datasets at the finer time scales are limited or hard to obtain. In addition to fixed time steps, some simulations may utilise a variable time step. This can happen in horticultural simulations where a coarse time interval is applied to dormant cycles, with finer time steps during periods of active change, such as springtime bud break and flowering. Additionally, sparsely timed events may be needed, for example, to include horticultural management actions and events. We refer to the simulation processes for manipulating regular time-based updates inside the simulation as the *simulation clock.*

Technically, processes running at finer timescales than the main simulation clock, may be included in the simulation process using a custom sub-grid calculation, with updates communicated back to the main simulation processes when the sub-grid calculation finishes. This approach is relatively common in models written as software code. There are also some simulation environments that provide communication interval

or other time synchronisation features (Mitchell and Gauthier, 1977, Fritzson and Pop et al., 2020). Predictor-corrector methods such as the well-known Runge-Kutta methods may be used in conjunction with larger time steps (Butcher, 1987), but the interpolating types may have an additional computational cost.

3.2. Collective simulation

We introduce the term *collective-simulation* as a general term to represent the description, orchestration, and execution of a multi-model simulation. Particularly for simulations where the models are interdependent (Figure 1).

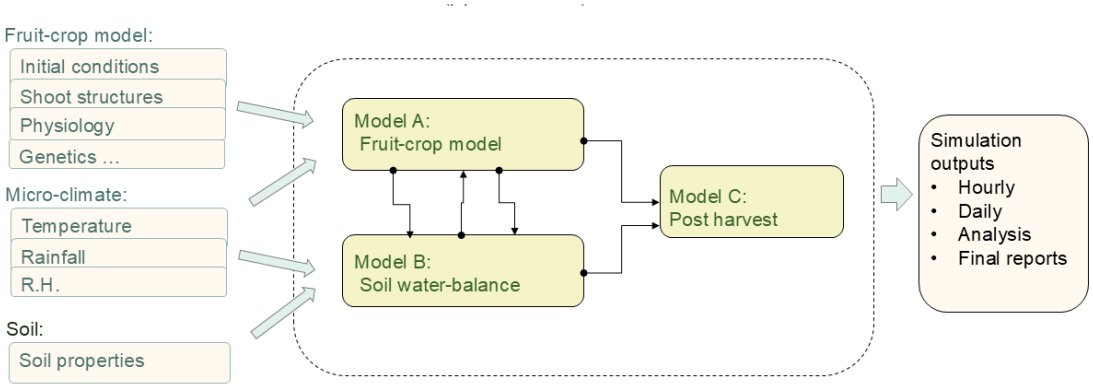

Figure 1 Schematic representation of a collective simulation including (left) selected model input datasets and parameters, (middle) a block representation of model connections, and (right) model outputs, post simulation analysis and reports. The arrows between models A and B represent reading of state-variables (section 3.4).

To orchestrate this type of collective simulation, we first need to describe its structure and the scheduling relationships between the component models. Our execution platform includes a set of virtual queues named slot-1, ..., slot-n (Figure 3b). Parsing the pipeline structure and queuing the model tasks is described as follows, Figure 2 shows the basic building blocks used to create the pipeline structure.

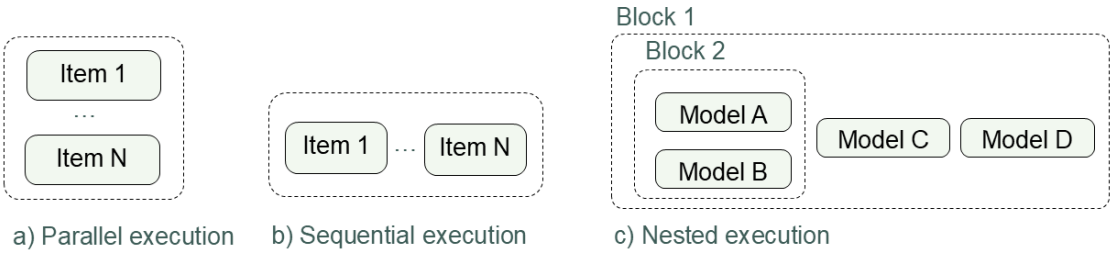

Figure 2 Representation of elements used to construct a simulation pipeline structure. a) and b) are basic building blocks for parallel and sequential execution, items may be blocks or models, and blocks may be nested. c) A nested block example that indicates models A and B are run in parallel, followed by model C then model D.

The structure in Figure 2c can be described by a tree graph (Figure 3a), which is traversed top to bottom, and left to right, blocks are expanded as they are encountered during the traversal and a synchronisation task is added after expanding each block. Models from parallel execution blocks are assigned to adjacent queue slots. The final queued state of the models for the simulation example in Figure 2c is shown in Figure 3b.

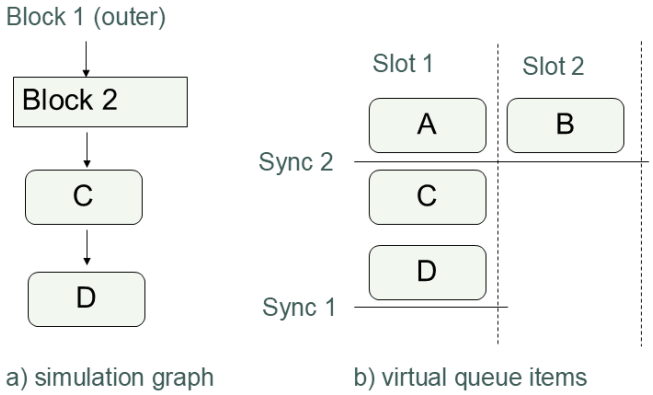

Figure 3 a) graph representation of the nested simulation example (Figure 2c), b) queued models after parsing the example simulation. Models A and B run in parallel, Sync 2 waits for A and B, Sync 1 waits for D.

Execution blocks at the synchronisation barriers and waits until all tasks generated in the block have completed.

The collective simulation pipeline structures are implemented in XML and included in the *simulation scenario* and *simulation instance* data structures described below.

A *simulation scenario* is a complete recipe for running a simulation (Figure 4) and includes descriptive records of the science goals for the simulation, the models used, the pipeline structure of the simulation, and the data classes needed to run the simulation.

When the model is configured to run, the simulation identifier, any user adjustments to simulation settings, and identifiers of the selected input and output datasets are combined with the *simulation scenario* information to create the similarly named *simulation instance* (Figure 4). Once the simulation is run, this *simulation instance* becomes an immutable record of this particular simulation.

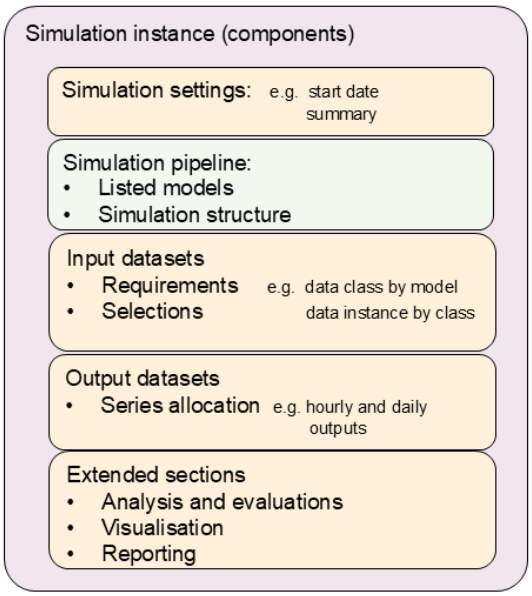

Figure 4 The main components of a simulation instance record, provides a complete description of a simulation. A simulation scenario template is simply a simulation instances with the simulation identifier and dataset selections removed.

3.3. Model integration

It is necessary to briefly describe a few of the main steps that need to be taken to prepare a model to run on our platform when using the model-to-model communication features.

Full integration: Only one model in the simulation can run the top-level control loop that is providing the global simulation clock (*control phase*), but all models will need to execute the code that runs the update steps (*run phase*). Additionally, we require models to include an initialisation phase (*prep phase*) where any read-write state-variables the model intends to use must be initialised. Similarly, a post simulation clean-up phase (*post phase*) should be defined. An *info* phase is the final phase required and is used by the platform to retrieve model-metadata. In summary, the model should implement a calling interface with entry points for: *control*, *run*, *prep*, *post*, and *info*.

Partial integration: For models not intending to use the model-to-model communication interface or where it is not possible to adapt the model code, we provide a wrapper application, with reduced interface requirements of; *run*, *prep*, *post*, and *info*. In this case the model is run in the normal way during the *run* phase. Such models may still be run in parallel with fully integrated models, but it will not be possible to use the model-to-model communication.

3.4. Model-to-model communication

Models running in parallel on our platform may interact indirectly with each other by reading and writing state-variable information to a simulation layer component (cache). This is achieved as depicted in Figure 5

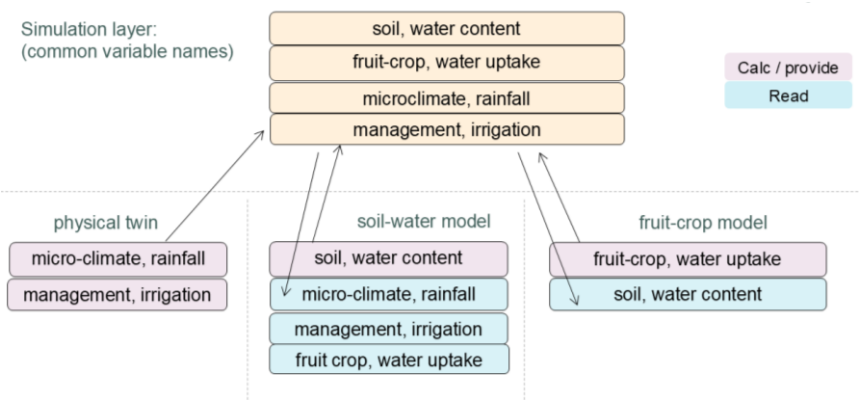

Figure 5 Mapping variables between models, this schematic shows netlist connections between the physical twin, soil water-balance model, and fruit crop model. The simulation layer (cache) provides a simple ontology-like name space defined in the simulation scenario.

The simulation layer is implemented as a memory-based cache structure. Models read and write state-variables only to the simulation layer. We use a tuple-based reference for naming state-variables, using: *owner*, *domain, category, name. Owner* is used by the platform, and *domain*, *category*, and *name* are available to the models, for example, *physical-twin, micro-climate, rainfall.* This provides a convenient ontology-like namespace for variables in the model. In our current prototype, the state-variables

are named consistently between models and the simulation layer (i.e. are the same). However, our design allows for translation of the state-variable names to occur as part of the read/write interface.

The simulation layer also writes the state-variables to persistent storage at each update, allowing the underlying data platform to capture a complete record of the simulated states. Additionally, the read/write interface allows writing of an initial value for state-variables during initialisation. Our Minerva data platform (Bell et al, 2025), developed to support our horticultural modelling and simulation work, is also able to transform datasets from one format to another during the data load (and store) steps. This is useful when two models in the simulation need to use the same dataset but with a different format, and when integrating existing models.

In many simulation models, micro-climate input variables such as rainfall, air temperature, etc, are loaded as table data, either all-at-once, or record-by-record, typically from a file or database. A useful technique in our platform is to adapt the models to treat these as (read only) state-variables, which then provides the capability to switch the micro-climate between different sources, such as historical datasets, forecast data, or model generated perturbations.

4. APPLICATION TO SIMULATION AND MODELLING

Throughout this work we have used a demonstration *simulation scenario* for understanding requirements and testing our prototypes. This demonstration simulation comprises two models configured to run in parallel. The first model is the FruitCropXL functional structural fruit-crop model (Zhu et al., 2021), and the second is a soil water balance model (Cichota et al., 2021). A diagram representing the simulation is shown in Figure 6.

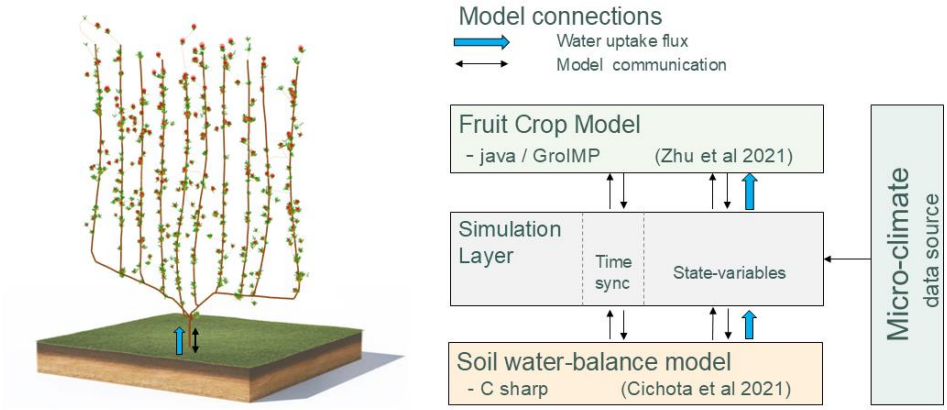

Figure 6 (left) Diagram representing the two-model demonstration simulation. (right) Configured structure of the simulation. The communication of water exchange between the models occurs via the tree water update flux, and soil water content state-variables using the simulation layer. The apple tree visualisation is created by our Sight Crop visualisation tool using data from the FruitCropXL model (Zhu et al, 2021).

The exposed state-variables available in these models are summarised as depicted earlier in Figure 5 and the connections drawn in Figure 6, and relate to the water uptake flux, soil water content and shared micro-climate data. These two models were developed independently and use different file formats for several of the input datasets. We were able to readily resolve this by pre-loading microclimate data into the simulation layer (cache) and applying format transformations, specific to each model, when loading soil parameter datasets from the data platform.

5. DISCUSSION AND CONCLUSION

We have developed an approach for model-to-model communication in biophysical models for horticultural systems based on using the modelled state-variables of the system. This operates by having the models communicate with a memory-based simulation layer (cache) using an ontology-like tuple-reference to name each shared variable. The simplicity and standardisation of the state-variable based data exchange between models provides an opportunity for external tools to integrate with simulations.

The model interfacing work described here integrates well with our data platform tools, and when used together these provide useful data transform features that have allowed us to resolve several data format inconsistencies between the models without having to rework the model code.

The partial model integration process can be carried out without reworking the model code. However, the models do need a small wrapper interface when integrating in this way. Partial integration gives these models

access to the data management, and identity management capabilities of our wider tool system. Fully onboarding models in the platform, requires more work, as the main control loops that run the main simulation clock, and solving processes need to be isolated from the sections of code that run the update equations at each simulation step. Although, this is relatively straight forward, in a large modelling framework it helps to have a good working knowledge of the model to perform this integration efficiently.

Our horticultural test simulation was successful for demonstrating the proposed approach. We have been able to connect two quite distinct models, the FruitCropXL model is a complex and relatively large GroIMP-Java based model, whereas the soil-water balance model is a simpler, but non-trivial C# based model. The relatively clean and lightweight read/write interface of our tools has helped make it straightforward to expose and connect the chosen state-variables in each model. Initially, we have provided support for four languages, C#, Java, Python and R, based on our underlying interface library written in C++. The state-variable references and our read write interface are also readily adaptable to other languages. This is particularly useful when connecting models created in different teams, or organisations.

In our approach the simulation layer (cache), comprised of the system state variables, provides a centralised layer that mediates communication between models. All state variables written to the simulation layer are tracked in output datasets by our underlying data platform. Additionally, the simulation instance is used by our modelling platform to record identifiers for all input datasets and parameters used in the simulation, which can be used to re-run and reproduce the simulation. Ideally, for best future model re-use and interoperability, a relatively high proportion of the model state-variables should be exposed externally by the techniques we have described. However, in practise choosing a smaller group of state-variables has been an effective method for our work. The model state-variables are treated as proxies for the underlying state-variables of the real-world biophysical system, and as such are suitable for describing the connections between models.

The main focus of our work has been to provide tools that improve the interoperability and re-use of models in different situations. In our demonstration the overall performance was dominated by the performance of the fruit crop model simulation, which was the more complex of the two models used. Although there is a computational cost for the model-to-model communication, the use of a memory-based simulation layer helps avoid performance bottlenecks. Another aspect of interoperability is the degree to which models are treated as white-box or black-box in nature. Our approach considers the model inputs, outputs, and the updated states of the models, which are treated as black-box models. However, some semantic knowledge may be needed to ensure the use of connecting state-variables is appropriate.

Standardising model communication interfaces is a common approach in integrative modelling and is most useful and easily applied when connecting relatively small groups of similar models. Similar in this context is considered to imply models operating at the same scale, and paradigm. Integration becomes more challenging as we connect models operating in dissimilar scales or paradigms. As the number of models in a framework increases, it becomes less desirable to modify and adapt individual models to facilitate particular pair-wise combinations, since this may unnecessarily restrict the combinations of models able to be used. In our state-variable approach, we anticipate the use of interposer models that perform the transformations and aggregations needed to adapt between temporal and spatial scales and dissimilar model paradigms.

Our work is an attempt to address the complexity that arises in multi-model biophysical modelling. This includes both the complexity of the biophysical systems, and the complexity of integrating multiple models of multiple sub-systems and processes, sometimes developed independently. In our approach the *collective simulation* description is a high-level description of the simulation and models only in terms of the models and their datasets. The virtual queue shown in Figure 3 represents a bulk synchronous parallel (Valiant, 1990) computing scheme, with synchronisation points having a tree-barrier structure. Our approach extends the common directed acyclic graph reduction of compute workloads (Aho and Garey et al., 1972) to allow interdependent processes such as those describing interconnected biophysical systems to be scheduled reliably for execution.

The interdependencies in the orchestrated workloads are managed effectively using the synchronisation points, which arise naturally from the decomposition of the collective simulation structure. The overall approach is general with high level abstractions for the main components Our model integration strategy using a state-variable based approach, combined with the ability to orchestrate interdependent biophysical models, lays the foundation for horticultural modelling platforms providing a means for building and running simulations comprised of independently developed biophysical models in a way that facilitates the reuse of models in different situations.

ACKNOWLEDGMENTS

This research was funded by the New Zealand Ministry of Business, Innovation, and Employment (MBIE) through the Strategic Science Investment Fund (contract number C11X1702) as part of the Digital Horticultural Systems Ngā Pou Rangahau, a Growing Futures™ programme at The New Zealand Institute for Plant and Food Research Limited.

The authors acknowledge the wider Digital Horticulture Systems teams and research programs for their collaborations and contributions to this work.

RESOURCE AVAILABILITY

Requests regarding original code developed by our team may be made to the corresponding author.

REFERENCES

Aho AV, Garey MR, Ullman JD (1972) The transitive reduction of a directed graph. SIAM J. Compute 1(2), 131-137

Bell ST, Liu D, Philip A, Cerecke C, Tang J, Nichols M, Gilmore Z, and Lin HT (2025) The Minerva data platform – towards a data platform for research applications. MODSIM2025, 26th International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, December 2025.

Brémaud P (1999) Markov Chains Gibbs Fields, Monte Carlo Simulation, and Queues. Springer-Verlag, New York.

Butcher JC (1987) The Numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods, J. Wiley, New York. 408-505.

Cichota R, Vogeler I, Sharp J, Verburg K, Huth N, Holzworth D, Dalgliesh N, Snow V, (2021) A protocol to build soil descriptions for APSIM simulations. MethodsX, 8:101566.

Fritzson P and Pop A et. al., (2020) The OpenModelica Integrated Environment for Modeling, Simulation and Control, MIC journal, 41(4), 241-295, doi: 10.4173/mic.2020.4.1

Gerbessiotis AV. and Valiant LG, (1994). Direct Bulk-Synchronous Parallel Algorithms. Journal of Parallel and Distributed Computing 22(2), 251-267.

Holzworth D, Huth NI, Fainges J, Brown H, Zurcher E, Cichota R, Verrall S, Herrmann NI, Zheng B, and Snow B, (2018) "APSIM Next Generation: Overcoming Challenges in Modernising a Farming Systems Model." Environmental Modelling & Software 103(1) 43–51. doi.org/10.1016/j.envsoft.2018.02.002.

Kniemeyer O, and Kurth W (2008) "The modelling platform GroIMP and the programming language XL."Applications of Graph Transformations with Industrial Relevance AGTIVE '07, edited by A. Schürr, M. Nagl, A. Zündorf, Springer, 570-572.

Marshall-Colon A, Long SP , Allen DK, Allen G, Beard DA, Benes B, von Caemmerer S, Christensen AJ, Cox DJ, Hart JC, Hirst PM, Kannan K, Katz DS, Lynch JP, Millar AJ, Panneerselvam B, Price ND, Prusinkiewicz P, Raila D, Shekar RG, … , Zhu XG, (2017) Crops In Silico: Generating Virtual Crops Using an Integrative and Multi-scale Modeling Platform. Frontiers in Plant Science: Plant systems and synthetic biology, 8. doi.org/10.3389/fpls.2017.00786.

Mawson AJ, Stanley CJ, Zhu J, Pattemore DE, Chooi KM, Oliver RJ, Lin HT, and Harker FR (2023) Developing a digital twin of apple production and supply chain ecosystems. Acta Hortic. 1360, 129-136, doi: 10.17660/ActaHortic.2023.1360.17

Midingoyi CA, Pradal C, Enders A, Fumagalli D, Raynal H, Donatelli M, Athanasiadis IN, Porteri C, Hoogenboom G, Holzworth D, Garcia F, Thorburn P, Martre P, (2021) Crop2ML: An open-source multi-language modeling framework for the exchange and reuse of crop model components, Environmental Modelling and Software, 142, 105055

Mitchell EEL, Gauthier JS. Advanced Continuous Simulation Language (ACSL). SIMULATION. 1976;26(3):72-78. doi:10.1177/003754977602600302

Stanley J, Zhu J, Rojo F, Kaneko T, Breen K, Friend A, Teixeira E, Niemann N, Mawson J (2024). Advancing high-performance apple production systems through digital twin modelling. Acta Hortic. 1395, 125-132, doi: 10.17660/ActaHortic.2024.1395.17

Zhu J, Gou F, Rossouw G, Begum F, Henke M, Johnson E, Holzapfel B, Field S, Seleznyova A, (2021) Simulating organ biomass variability and carbohydrate distribution in perennial fruit crops: a comparison between the common assimilate pool and phloem carbohydrate transport models, in silico Plants, 3(2), 1-20, doi.org/10.1093/insilicoplants/diab024

Valiant LG, (1990) A bridging model for parallel computation. Communications of the ACM. 33(8) 103-111

