in silico Plants Vol. 4, No. 1, pp. 1–13
https://doi.org/10.1093/insilicoplants/diac003
Advance Access publication 12 February 2022
Original Research

in silico Plants
OXFORD

BioCro II: a software package for modular crop growth simulations

Edward B. Lochocki[1,*], Scott Rohde[1], Deepak Jaiswal[1,2,3], Megan L. Matthews[1,4], Fernando Miguez[5], Stephen P. Long[1,6,7,8] and Justin M. McGrath[1,6,9,*]

[1]Carl R. Woese Institute for Genomic Biology, University of Illinois, Urbana–Champaign, Urbana, IL 61801, USA
[2]Environmental Sciences and Sustainable Engineering Center, Indian Institute of Technology Palakkad, Palakkad, Kerala 678557, India
[3]Department of Civil Engineering, Indian Institute of Technology Palakkad, Palakkad, Kerala 678557, India
[4]Department of Civil and Environmental Engineering, University of Illinois, Urbana-Champaign, Urbana, IL 61801, USA
[5]Department of Agronomy, Iowa State University, Ames, IA 50011, USA
[6]Plant Biology Department, University of Illinois, Urbana–Champaign, Urbana, IL 61801, USA
[7]Crop Sciences Department, University of Illinois, Urbana–Champaign, Urbana, IL 61801, USA
[8]Lancaster Environment Centre, Lancaster University, Lancaster, LA1 4YQ, UK
[9]USDA ARS Global Change and Photosynthesis Research Unit, Urbana, IL 61801, USA
*Corresponding author's e-mail address: justin.mcgrath@usda.gov

Handling Editor: Xin-Guang Zhu

Citation: Lochocki EB, Rohde S, Jaiswal D, Matthews ML, Miguez F, Long SP, McGrath JM. 2021. BioCro II: a software package for modular crop growth simulations. *In Silico Plants* **2022**: diac003; doi: 10.1093/insilicoplants/diac003

ABSTRACT

The central motivation for mechanistic crop growth simulation has remained the same for decades: to reliably predict changes in crop yields and water usage in response to previously unexperienced increases in air temperature and CO_2 concentration across different environments, species and genotypes. Over the years, individual process-based model components have become more complex and specialized, increasing their fidelity but posing a challenge for integrating them into powerful multiscale models. Combining models is further complicated by the common strategy of hard-coding intertwined parameter values, equations, solution algorithms and user interfaces, rather than treating these each as separate components. It is clear that a more flexible approach is now required. Here we describe a modular crop growth simulator, BioCro II. At its core, BioCro II is a cross-platform representation of models as sets of equations. This facilitates modularity in model building and allows it to harness modern techniques for numerical integration and data visualization. Several crop models have been implemented using the BioCro II framework, but it is a general purpose tool and can be used to model a wide variety of processes.

KEYWORDS: Dynamical systems; mechanistic crop growth simulation; modular modelling; multiscale modelling.

1. INTRODUCTION

Mechanistic crop growth simulations play key roles in understanding the impact of global change on agriculture and in directing efforts to engineer plants capable of feeding the future world (Clark *et al.* 2001; Menon *et al.* 2007; LeBauer *et al.* 2013; Marin *et al.* 2014). By representing critical aspects of plant biology as biochemical processes rather than statistical associations, these simulations are able to capture the underlying mechanisms determining crop yields and water usage outside of empirical experience, such as those expected to occur due to climate change (Humphries and Long 1995). Given their importance, it is essential to ensure that these crop growth models are accessible to experts in these mechanisms who do not have the time or inclination to delve into details of computer science. Communication between these experts is critical for assembling and maintaining a state-of-the-art capacity to predict crop responses to environmental change. The software package WIMOVAC (Humphries and Long 1995; Song *et al.* 2017) was an early success in this area, facilitating investigations into the response of plants to elevated atmospheric CO_2 concentrations (Rogers and Humphries 2000; Wittig *et al.* 2005) and eventually giving rise to a successor, BioCro, that expanded the list of available crops

(Miguez et al. 2012, 2009; Wang *et al.* 2015; Larsen *et al.* 2016; Jaiswal *et al.* 2017).

As the field of mechanistic crop growth simulation has progressed, the individual components of these models have grown more realistic but also more complex and specialized. For example, idealized roots can now be replaced with realistic three-dimensional structures (Postma *et al.* 2017) and simplified photosynthetic equations can be replaced by detailed kinetic models (Zhu *et al.* 2013). However, the previous version of BioCro and other similar software packages do not facilitate the process of replacing one model subcomponent with another because their source code intermixes parameter values, model equations, numerical solution algorithms and user interfaces. Modifying one component, such as the photosynthesis equations, required changes in many places throughout the code, is challenging for those unfamiliar with the software, and is inefficient and error prone. Consequently, these packages are slow to take advantage of new developments.

A more flexible structure, where a model's equations are separated from the details of its solution, would address this. Such a modular crop growth simulation software can be understood as being analogous to acquiring a car by passing specifications to a dedicated factory (Fig. 1). By contrast, the prevailing approach is more like the brute-force alternative of fabricating and assembling a car from raw materials. The advantages of the modular approach are numerous, with some of the most significant being that the user (i) does not need the specialized knowledge required to perform the simulation, (ii) is easily able to swap model components for newer or more specialized versions when they become available and (iii) can build an understanding of individual model components as well as their interactions within the context of the entire model, taking advantage of the logical boundaries made by the module writers. Here we present a fully modular version of BioCro—*BioCro II*—that implements this vision and allows modellers to focus on biology rather than computer programming. We describe the essential elements of its design and demonstrate how it allows users to harness the power of new model components and quantitatively compare their responses to environmental and physiological factors.

1.1 An illustrative example

To give an overview of BioCro II's design and usage, a simple model is presented (Listing 1). It is a simplistic growth model that demonstrates the main parts of BioCro II but does not represent a particular crop; the light profile and growth rate are both unrealistic. The canopy assimilation rate (A) is calculated based on the incident light intensity (Q), leaf mass ($Leaf$), specific leaf area (SLA) and a radiation use efficiency (RUE) factor (a_{RUE}). Leaf and root mass increase as fractions (f_{leaf} and f_{root}) of A are allocated to those tissues, with known initial values of those masses (Leaf(0) and Root(0)). Light intensity is given as a table of values at every second in a 24-h period. For this model, we desire a solution at those same time points.

It is noteworthy that this model can be viewed as a set of equations; even the discrete values of Q can be considered a set of equations. Treating all components as equations allows one to think about all parts of a model in the same manner, enabling simpler model specification and less code. BioCro II has been designed to facilitate this way of thinking, and one of its key features is that sets of equations can be

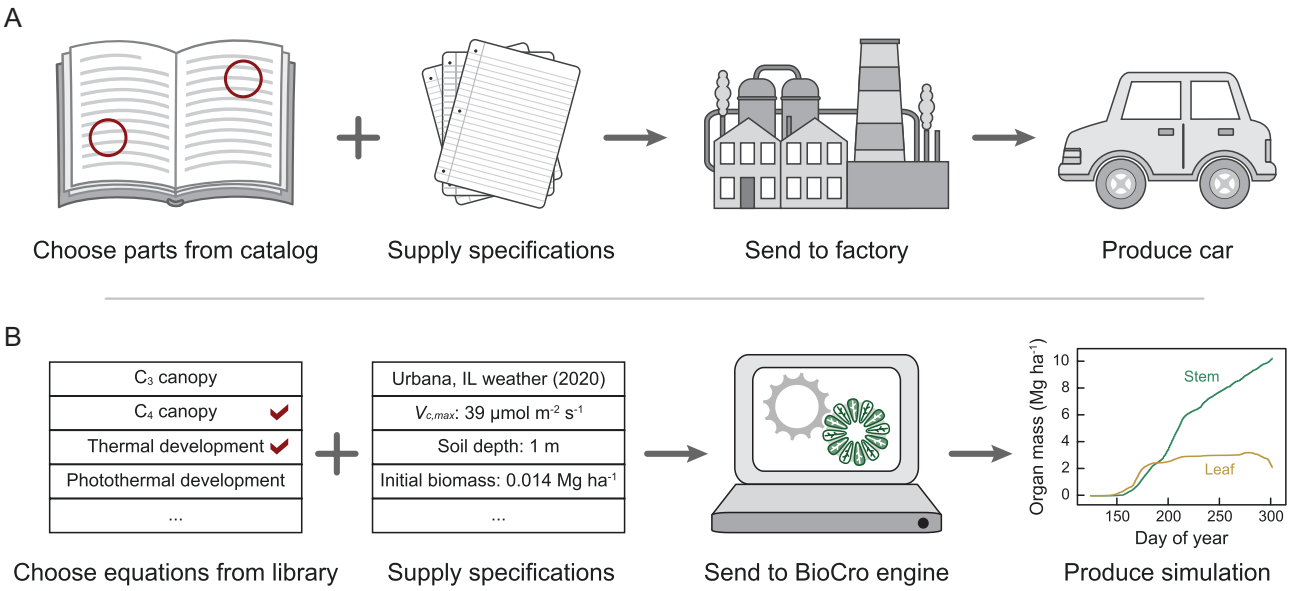

Figure 1. (A) Schematic diagram illustrating the process of acquiring a car by choosing parts from a catalogue, supplying additional specifications and sending this information to a factory where dedicated experts take care of fabrication and assembly. (B) Analogous schematic diagram illustrating the process of running a crop growth simulation by choosing equations from a library, supplying additional specifications and sending this information to the BioCro engine, which solves the model and calculates the results. Book, paper, factory, car and computer clipart images were obtained under open licenses from https://creazilla.com.

$$A = Q \cdot \alpha_{RUE}$$
$$Mass = A \cdot Leaf \cdot SLA \cdot C_{conversion}$$

} `example_model_mass_gain`

$$\frac{dLeaf}{dt} = Mass \cdot f_{leaf}$$
$$\frac{dRoot}{dt} = Mass \cdot f_{root}$$

} `example_model_partitioning`

$$Leaf(0) = 1\ \text{kg}$$
$$Root(0) = 1\ \text{kg}$$

} `initial_values`

$$\alpha_{RUE} = 0.07\ \text{mol mol}^{-1}$$
$$SLA = 25\ \text{m}^2\ \text{kg}^{-1}$$
$$f_{leaf} = 0.2\ \text{kg kg}^{-1}$$
$$f_{root} = 0.8\ \text{kg kg}^{-1}$$
$$C_{conversion} = 0.03\ \text{kg mol}^{-1}$$

} `parameters`

$Time$ (s)	Q (mol m^{-2} s^{-1})
0	0.00E+00
1	7.27E-08
2	1.45E-07
...	...
86398	1.45E-07
86399	7.27E-08
86400	2.45E-19

} `light_intensity` (values are interpolated to create a continuous function of time)

Listing 1. A simple crop growth model simulating root and leaf growth over a period of 1 day. The model itself is simply a list of equations, but the equations can be arbitrarily grouped into sets for convenience.

grouped and named. In the BioCro II R package, a model is defined and solved by passing the names of equation sets to the `run_biocro` function, which assembles the equations, performs basic checks that the model is solvable and then numerically solves it. Using the names shown for the equation subsets in Listing 1, the model solution would be obtained in the R environment as follows:

```
run_biocro(initial_values, parameters,
light_intensity, 'example_model_carbon_
gain', 'example_model_partitioning')
```

The result contains values for each quantity in the model at each desired time point in the period (Fig. 2). Details of the syntax and how to define sets of equations are given in Appendix 1 and Section 2 of *A Practical Guide to BioCro* (see Supporting Information). In this manuscript, the emphasis is that the concept of named sets of equations provides a way to piece together models that is flexible and familiar to modellers. This feature, along with the ability to combine models or exchange similar submodels, allows for easier development and comparisons. Although no individual feature of BioCro II is unique, we believe that the combination of modularity, scripting access through R and open-source code is unique among popular modelling software. A summary of BioCro II compared to other modelling software is provided in **Supporting Information—Section S4**, and an example of swapping submodels and comparing their performance is presented in Section 3.

Since several crops can be modelled using the existing library of equations included in the R package, much can be accomplished using the R environment by only varying parameters and other inputs, such as weather conditions. Thus, simulations can be performed for various regions and climates, parameters can be optimized to match observed data and sensitivity analyses can be performed without modifying the source code of BioCro II.

2. OVERVIEW OF BIOCRO II ORGANIZATION AND FUNCTIONALITY

As illustrated by the example above (Listing 1), BioCro II facilitates the construction of models from modular sets of equations. This ability is enabled by the careful arrangement of its source code, which contains several examples of modularity at different levels of its organization. Here we give a broad overview of the source code structure. While of general benefit to most users, this knowledge is required to modify or add new sets of equations.

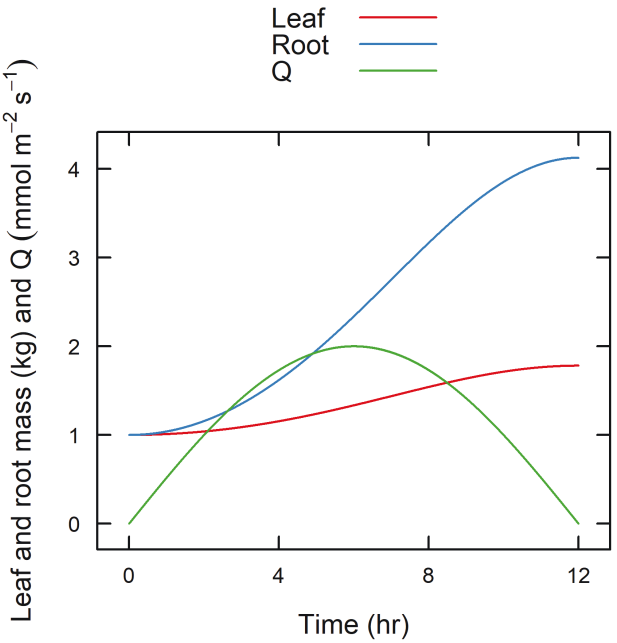

Figure 2. Graphical representation of the solution of the simple model presented in Listing 1 showing how some of the model's quantities change overtime.

BioCro II is written primarily in C/C++ and is accessible through an R package that provides a more user-friendly terminal-based interface (Fig. 3A; source code is available in the Biocro.tar.gz file of the Supporting Information). The interface consists of two sets of functions: R-to-C functions, which convert R objects to C objects and pass them to core BioCro II functions, and R functions, which pass R objects to the R-to-C functions. This is an example of modularity at the highest level of the code organization and it provides several key benefits. Accessing BioCro II via R provides simple routes for data input and analysis, especially considering the wide array of libraries available for statistical operations and plotting. This design is also flexible enough to allow other interfaces in the future; for example, a Python interface could be developed by writing appropriate Python and Python-to-C files, without needing to duplicate any of the central code related to BioCro's main operations.

The central code itself is also split into several distinct groups: the module library, the ordinary differential equation (ODE) solver library and the solution framework (Fig. 3B). In BioCro II, a *module* represents one or more equations that define a model component; for example, the *ball_berry* module uses the first equation in Ball *et al.* (1987) to calculate a leaf's stomatal conductance to water vapour (g_{sw}). Each module has associated *input quantities* and *output quantities*, which are all numeric variables with names such as *conductance_stomatal_h2o* (representing g_{sw}). Modules are divided into two broad types based on how time is used in the equations: *differential modules*, which calculate rates of change for output quantities, and *direct modules*, which calculate instantaneous values for output quantities. When performing a simulation, the user specifies a list of modules that then are combined to form the overall crop model. BioCro II's standard module library is capable of replicating the full functionality of the original BioCro, but researchers can also develop their own module components privately and use them in conjunction with the standard BioCro II modules. The

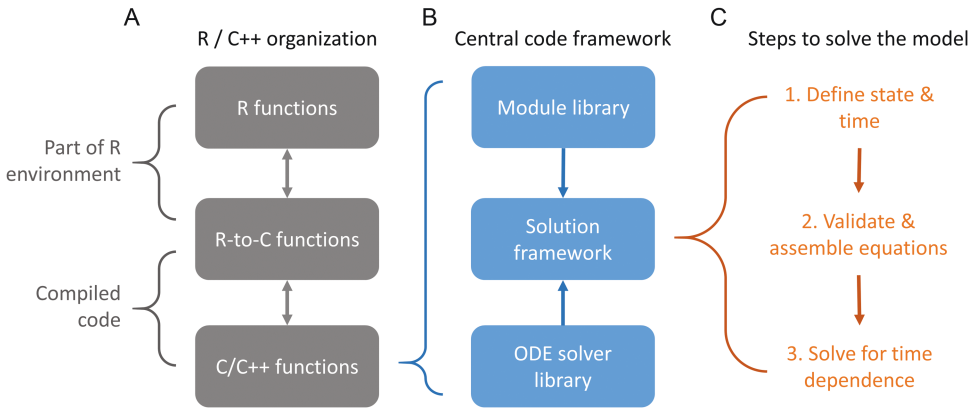

Figure 3. Schematic diagram illustrating the organization and functionality of BioCro's source code. Components are shown as boxes, and steps to determine the model solution are shown as numbered items. (A) At the highest level, an R interface to the central C/C++ files is provided via R and R-to-C files. (B) The main C/C++ code is divided into three distinct groups: the module library, the ODE solver library and the solution framework. (C) To perform a simulation, the user passes a set of modules, an ODE solver and other specifications to the framework, which defines the state, checks the equations and inputs for validity and solves for the value of each of the unknown quantities over the time domain of the simulation.

BioCro II R package additionally gives users the ability to calculate output quantities from individual modules (via the *evaluate_module* function), allowing a single module's behaviour to be demonstrated and visualized.

Another essential component of BioCro II is the *numerical ODE solver*, which determines the time evolution for those quantities whose derivatives are calculated by differential modules (henceforth called 'differential quantities'). In the original BioCro, ODE solving was carried out exclusively using the fixed-step Euler method. However, this algorithm is known to be computationally inefficient and inaccurate when solving oscillatory systems or those with exponential growth or decay, often producing an unstable output (Flannery *et al.* 1992). In BioCro II, users can instead choose an algorithm from the built-in ODE solver library. The library includes more advanced options from Boost *odeint* library ('Boost C++ Libraries, version 1.71.0,' 2019), such as the adaptive fourth-order Rosenbrock method, which is particularly well suited for solving stiff systems. The separation between a crop growth model's equations and its solution method is a key feature of BioCro II that allows those without expertise in numerical methods to harness the power of modern ODE solving algorithms without needing to delve into their intricate details.

To perform a simulation, a user must supply lists of direct and differential modules, a numerical ODE solving algorithm and the following values for quantities: (i) the initial values of the differential quantities, (ii) weather data and other time-dependent quantities that are taken as known beforehand and (iii) values of quantities that are assumed constant for the period of the simulation. The separation of model parameters and weather data from the model equations represents another instance of modularity in BioCro II. Using the R package, a simulation can be run with the *run_biocro* function. Its arguments are ultimately passed to the solution framework, which handles the details of defining and solving the model (Fig. 3B and C): First, a set of state quantities is identified from the modules and other inputs. Then, the quantities and modules are checked for consistency and validity—for example, to ensure all module input quantities are defined. Finally, the ODE solver is used to determine the time evolution of the quantities defined by derivatives, which simultaneously determines the time evolution for other unknown quantities defined by direct modules. These actions are all performed automatically by the BioCro II framework, which handles the details of the solution, so the user can focus on the science.

Using the same equations, values and solution algorithms, BioCro II produces results similar to those produced by the previous version and uses slightly less computational time [**see Supporting Information—Section S3**]. The amount of model code is much smaller though, since modules are reused, eliminating code repetition. For more details about the functions and library entries available in the BioCro II R package, see *A Practical Guide to BioCro* (see Supporting Information) or the online documentation (https://ebimodeling.github.io/biocro-documentation/).

3. QUANTITATIVE COMPARISON BETWEEN TWO PHOTOSYNTHESIS MODELS

A common reason for developing or using a new mathematical model is to represent a particular behaviour or phenomenon in a more realistic way. For example, replacing an empirical statistical model with a mechanistic process-based model can improve the accuracy of the model's predictions in situations outside experience. The ability to compare the output of two models quantitatively is key to this effort, since it can provide evidence that the new model is truly an improvement.

The modular structure of BioCro helps facilitate these types of comparisons, which can be illustrated by contrasting two models for carbon assimilation in soybean. Many crop growth simulations have relied on the concept of RUE to relate biomass accumulation to the available sunlight. This model simplifies complex environmental and biological interactions into a simple-to-use empirical model (Sinclair and Muchow 1999) with several variants that are all based on the observation that a measure of cumulative growth (e.g. net canopy CO_2 uptake or total above-ground dry matter) is often directly proportional to a measure of cumulative radiation exposure (e.g. intercepted solar radiation or absorbed photosynthetically active radiation) (Demetriades-Shah *et al.* 1992; Arkebauer *et al.* 1994). In contrast to RUE, it is also possible to calculate the CO_2 assimilation rate for a C_3 plant leaf using a set of equations representing the steady-state biochemical processes of photosynthesis, most commonly the Farquhar–von-Caemmerer–Berry (FvCB) model (Farquhar *et al.* 1980).

3.1 Soybean simulation details

Surface radiation, temperature, relative humidity and wind speed data were obtained for the years 1995 through 2020 from SURFRAD's Bondville, Illinois station (40°03′07″N, 88°22′23″W) (Augustine *et al.* 2000) and converted to hourly values using a previously described method (Lochocki and McGrath 2021). Daily precipitation totals were obtained from WARM's Champaign, Illinois weather monitoring station (40°05′02″N, 88°22′23″W) (Illinois State Water Survey 2015) and converted to hourly values by assuming a constant rate of rainfall during each day. Annual global mean values for atmospheric CO_2 concentration were obtained from the National Oceanic and Atmospheric Association's Global Monitoring Laboratory (NOAA 2021).

Default modules and parameter values for soybean simulations are identical to those reported previously (Matthews *et al.* 2021). As in that study, ODE solving was performed using the fifth-order Runge–Kutta method with fourth-order Cash–Karp error estimation ('Boost C++ Libraries, version 1.71.0,' 2019), available in the BioCro II ODE solver library as *boost_rkck54*. The algorithm adaptively chooses time steps based on absolute and relative error tolerances, which were set to 1×10^{-4} for all simulations. While solving ODEs, each weather quantity was treated as a continuous piecewise function defined by linear interpolation between the hourly values.

3.2 RUE versus FvCB photosynthesis models in soybean for a single year

For a full soybean simulation, a photosynthesis model must be combined with other model components that represent processes such as phenological development, assimilate distribution, senescence, water dynamics and others. The BioCro II R package includes preset lists of modules and parameters that have been optimized to reproduce experimentally observed soybean biomass in both ambient and elevated levels of CO_2 (Matthews *et al.* 2021). These can be used to run a full simulation for the year 2002 with the following R command:

```
run_biocro(soybean_initial_values, soybean_parameters, soybean_weather2002, soybean_direct_modules, soybean_differential_modules, soybean_ode_solver)
```

The objects `soybean_direct_modules` and `soybean_differential_modules` are lists of module names used by the soybean model. In this simulation, photosynthesis at the canopy level is determined using the `ten_layer_c3_canopy` module, which divides the canopy into 10 equal layers and then divides each layer into shaded and sunlit proportions. It then uses the light level for shaded and for sunlit leaves in each layer to determine CO_2 assimilation rate from coupled equations representing respiration, stomatal opening, transpiration and the FvCB model for photosynthesis (Humphries and Long 1995).

Alternatively, it is possible to replace the mechanistic photosynthesis model with a simpler one based on RUE, where the canopy is divided into layers as before but the gross CO_2 assimilation rate at the leaf level (A_{gross}) is determined from the incident photosynthetically active photon flux density (PPFD; Q) according to

$$A_{gross} = \alpha_{RUE} \cdot Q \tag{1}$$

where α_{RUE} is the efficiency with which light energy is used to assimilate carbon. Here, both A_{gross} and Q are expressed in units of μmol m^{-2} s^{-1}, making α_{RUE} dimensionless. A canopy photosynthesis module implementing Equation (1) is available in the BioCro II module library, so the FvCB model can be replaced with a RUE model from R using just two lines:

```
soybean_direct_modules$canopy_photosynthesis <- 'ten_layer_rue_canopy'
  soybean_parameters$alpha_rue <- XXX
```

where `XXX` is a placeholder for an arbitrary value assigned to α_{RUE}. Following this change, the `run_biocro` function can now be used to simulate soybean growth with the RUE model in place of the mechanistic FvCB model. By varying the value of α_{RUE} to minimize the difference between total end-of-season biomass values calculated using each of the FvCB or RUE models [**see Supporting Information—Section S2**], it is possible to achieve close agreement throughout the growing season despite the very different approaches used to determine A_{gross} (Fig. 4A).

The origin of this agreement can be investigated by examining the relationship between gross assimilation and PPFD in the two models. In the RUE model, there is a purely linear relationship, while in the FvCB model, photosynthesis eventually begins to level off at higher light intensities (Fig. 4B). The value of α_{RUE} that minimizes the difference between end-of-season yield of the two models is simply one that agrees with the FvCB model on average for the conditions experienced throughout the season; in general, the RUE model underestimates assimilation at lower light intensities and overestimates at higher intensities, producing an overall similar effect on A_{gross} and end-of-season yield. However, this balance is fragile and specific to a particular time and place because the FvCB model also responds to environmental factors such as the atmospheric CO_2 concentration ($[CO_2]$), relative humidity (RH) and temperature (T) without re-parameterization, as evidenced by the multiple values for A_{gross} that are possible for each value of Q in Fig. 4B.

3.3 Sensitivity analysis of the RUE and FvCB photosynthesis models

To better understand the response of the mechanistic photosynthesis model to environmental factors, and hence to reveal the dynamics that

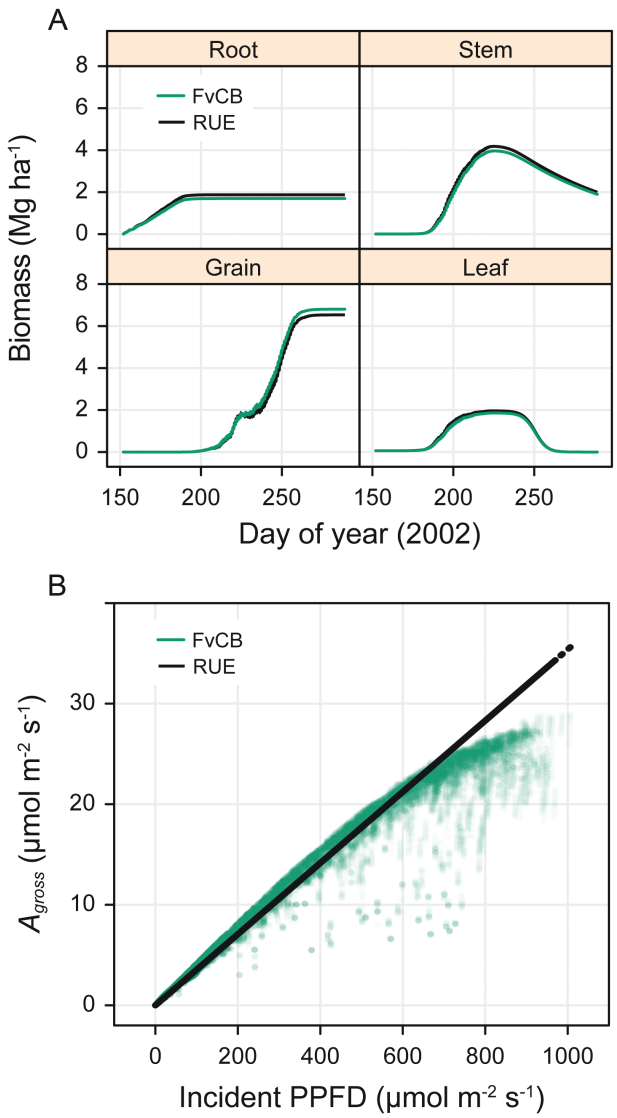

Figure 4. (A) Biomass for several soybean tissues (root, stem, grain and leaf) calculated throughout the 2002 growing season in Champaign, IL using either the mechanistic FvCB model for C_3 photosynthesis or an empirical RUE model with α_{RUE} = 0.0354. (B) Individual (A_{gross}, Q) pairs calculated with the FvCB model extracted from all leaf classes, canopy layers and times during 2002 (transparent circles) compared with the output from the RUE model (solid black line).

are not included in the RUE model, it is useful to calculate normalized sensitivity coefficients for A_{gross}, defined by

$$c_x(A_{gross}) \equiv \left[\frac{\partial A_{gross}/\partial x}{A_{gross,0}/x_0}\right] \quad (2)$$

where $\partial A_{gross}/\partial x$ is the partial derivative of A_{gross} with respect to some parameter x (e.g. light intensity or temperature) determined at a specific base value (x_0), and $A_{gross,0}$ is the value of A_{gross} at x_0. Essentially, $c_x(A_{gross})$ represents the fractional increase in A_{gross} per fractional increase in x and therefore is a measure of the influence of parameter x on the model output. In the BioCro R package, the `evaluate_module` function provides a generic interface to any module, allowing the user to specify the values of the module's input quantities and solve its equations for the values of its output quantities. Applying this function to the `c3_leaf_photosynthesis` and `rue_leaf_photosynthesis` modules, it is possible to calculate numerical derivatives for A_{gross} from each module, and therefore values for the normalized sensitivity coefficients $c_x(A_{gross})$. This process can be repeated with multiple independent variables x and at multiple values for the incident PPFD to produce a curve of sensitivity coefficients at different PPFD values, which shows how the models respond to environmental conditions at different light levels (Fig. 5A).

From this analysis it is clear that atmospheric $[CO_2]$ and air temperature have the largest impact on gross assimilation. Increases in atmospheric $[CO_2]$ cause increases in gross assimilation at all light levels, while increased temperature reduces assimilation at low light levels and increases it at high light levels. The corresponding sensitivity curves for the RUE model (not shown) are all identically zero since the RUE model does not consider any of these environmental factors when calculating assimilation rates. Thus, this analysis highlights the major discrepancies between the RUE and FvCB models.

However, when calculating total biomass throughout a soybean growing season, the photosynthesis model is embedded in a larger model representing additional physiological processes such as development, carbon distribution, carbon utilization, respiration, senescence, water availability and others. Since some of these processes may also respond to air temperature and atmospheric $[CO_2]$, it is also useful to calculate sensitivity coefficients for the total biomass M throughout a season in response to changes in a variable x according to

$$C_x(M) \equiv \left[\frac{\partial M/\partial x}{M_0/x_0}\right] \quad (3)$$

Although this formula is nearly identical to Equation (2), the method of calculation is different since M is the output of an entire simulation rather than a single model component; thus, instead of using `evaluate_module`, the `run_biocro` function is used to couple and numerically solve the sets of model equations.

The result of this analysis applied to temperature (Fig. 5B) shows that the temperature response of total biomass is not significantly different between the RUE and FvCB models, due to the temperature dependence of other submodels involved in the simulation. On the other hand, the response of biomass to atmospheric $[CO_2]$ is significantly different between the two models, with the RUE model having almost no response at all (Fig. 5C). Thus, although the RUE and FvCB models for photosynthesis differ greatly in the response of gross assimilation to several environmental factors such as relative humidity, wind speed, air temperature and $[CO_2]$, it is likely that changes in $[CO_2]$ will play the largest role in any discrepancies between models differing only in C_3 and RUE equations when considering the growth simulations as a whole.

3.4 RUE versus FvCB photosynthesis models in soybean for multiple years

Applying the model to weather data from 2006 but using the same value for a_{RUE} determined in 2002 reveals that the models have diverged, exhibiting larger differences in grain biomass (Fig. 6A). As expected, the delicate balance that produced good agreement in 2002 does not hold in another year where environmental conditions, including atmospheric $[CO_2]$, are different. Looking across all years, the two models disagree more often than they agree (Fig. 6B). Finally, by plotting the discrepancy against atmospheric $[CO_2]$, there is a clear downward trend, indicating that the RUE model underestimates total

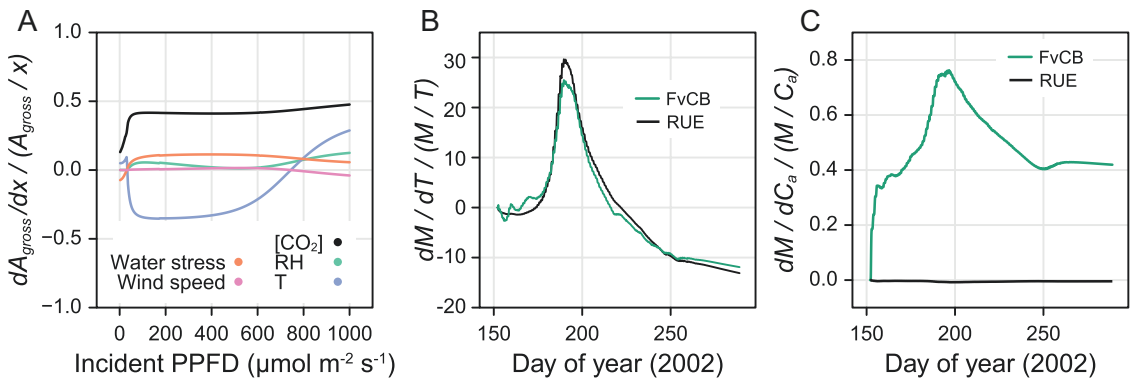

Figure 5. Normalized sensitivity coefficients for (A) FvCB gross assimilation calculated with respect to $[CO_2]$, relative humidity, water stress, air temperature and wind speed for different levels of incident PPFD, where base values are RH = 0.89, air temperature = 25 °C, wind speed = 5 m s^{-1}, atmospheric $[CO_2]$ = 372 ppm and water stress = 0.99; (B) total biomass calculated with respect to air temperature during the 2002 growing season; and (C) atmospheric $[CO_2]$ during the 2002 growing season.

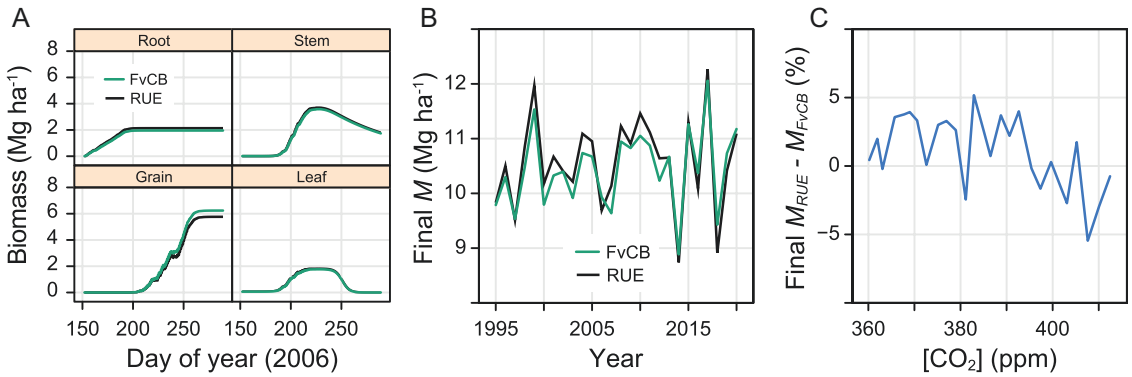

Figure 6. (A) Biomass for several soybean tissues (root, stem, grain and leaf) calculated throughout the 2006 growing season in Champaign, IL using either the mechanistic FvCB model for C_3 photosynthesis or an empirical RUE model with $\alpha_{RUE} = 0.0354$, which was optimized for agreement in 2002. (B) Total biomass at the end of the growing season calculated for the years 1995–2018 using the RUE and FvCB models for photosynthesis. (C) The biomass difference $M_{RUE} - M_{FvCB}$ expressed as a percentage plotted against average global atmospheric $[CO_2]$ for the years 1995–2018.

biomass when $[CO_2]$ increases, causing an error as large as 5 % for the years considered as compared to the more mechanistic FvCB model (Fig. 6C).

It is crucial to note that while high-level qualitative differences between the RUE and FvCB models can be gleaned immediately by the forms of their equations (e.g. the RUE model will not respond to changes in humidity), it is difficult to tell which differences will be most significant when embedding these models in larger crop growth simulations, and impossible to determine the size of the differences introduced into a simulation by using the empirical RUE model. The modularity of BioCro II facilitates a quantitative exploration of the differences between these models, where an understanding can be built up from the leaf-level photosynthesis scale to the field season scale. Not only is this analysis possible with BioCro II, it is also straightforward (see the *Quantitative Model Comparison* document and its associated R script in the Supporting Information), and can easily be applied to other aspects of plant growth modelling, such as assimilate distribution and stomatal opening. Here we have provided just one example of how BioCro II can be utilized to examine the costs and benefits of using simple versus more mechanistic models for different processes underlying crop yield in different environments, allowing the user to choose what is fit for their specific purpose.

4. BIOCRO II AS A CROSS-PLATFORM REPRESENTATION OF A DYNAMICAL SYSTEM

Introducing modularity into BioCro by separating the model equations, parameter values and the method of solution from each other makes the software easier to use and maintain, but this modification also has the additional benefit of allowing BioCro II to follow the general structure of a dynamical system. Dynamical systems can be used to specify a wide variety of mathematical models and consist of three parts:

1. A set of quantities of interest, which is called the *state* and symbolized by $\boldsymbol{X}$ (**see Supporting Information—Section S1** for a discussion of this terminology). Here, a quantity is specified by a name (e.g. *temperature*) and takes a numeric value (e.g. 25 °C); likewise, a state is specified by the names of its constituent quantities and takes a vector value formed from the values of its constituent quantities.
2. An independent quantity on which the state's value depends, which is often taken to be the *time* (symbolized by t).
3. An *evolution rule* that describes how to determine the state's value at a future time given its value at the current time, which is often implicitly formed from a set of equations coupled with a numerical solution algorithm.

With these three components, the state's value can be determined at any time point; a sequence of state values determined at a sequence of times is called the *solution* of the dynamical system. This is an extremely flexible design that inherently separates the definition of a model from the calculation of its solution. Defining a dynamical system is also relatively simple, as it is formed by a list of equations and quantities. Owing to this simplicity and flexibility, framing models as dynamical systems is nearly ubiquitous across a wide range of fields.

Although we use a definition that makes no distinction between types of state quantities in the concept of a dynamical system, for practical reasons, BioCro divides the state quantities into several subsets based on the structure of their equations. Each of these subsets is an input argument of the `run_biocro` R function and defines one or more of the components of a dynamical system as follows (see also Fig. 7 and *A Practical Guide to BioCro* in the Supporting Information):

- **parameters**: specifies $\boldsymbol{X}_{parameter}$, the state quantities that do not change with time; also defines the corresponding values.
- **drivers**: specifies $\boldsymbol{X}_{driver}$, the state quantities whose time-dependent values are taken as known beforehand at a discrete set of time points; also defines the corresponding values and the valid time domain.
- **direct modules**: specifies $\boldsymbol{X}_{direct}$, the state quantities whose instantaneous values are determined from functions of time and the other state elements; also defines the corresponding equations.

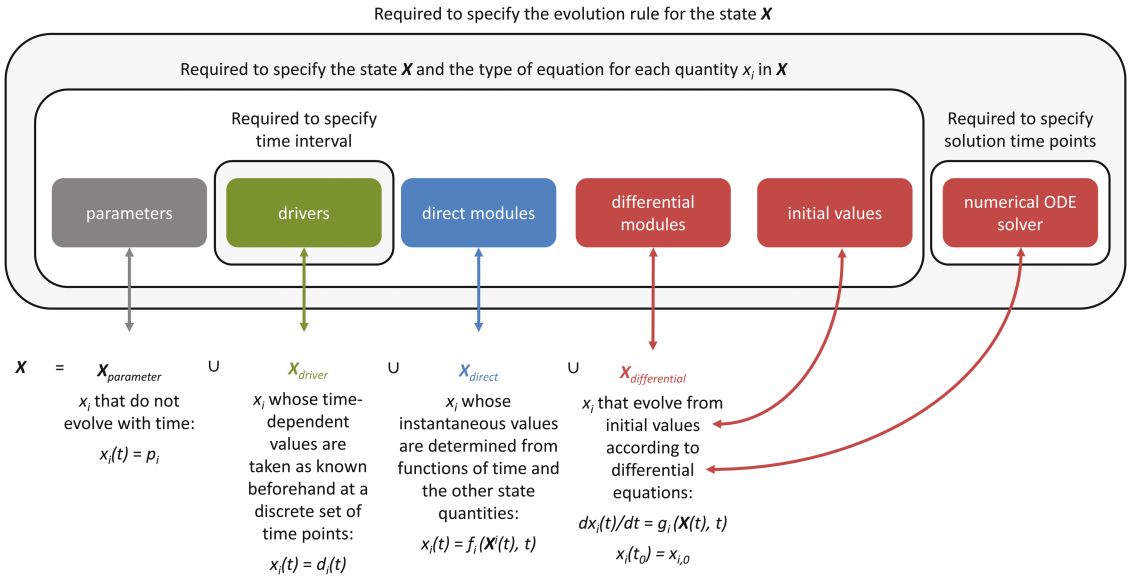

Figure 7. Schematic illustration of the relationship between the inputs to BioCro's `run_biocro` function (solid boxes) and the essential components of a dynamical system. p_i and $x_{i,0}$ are constant values; $d_i(t)$ is a continuous function of time determined from a set of values at discrete time points, for example, by linear interpolation; and $\mathbf{X}_i$ is the set of state quantities excluding x_i.

- **differential modules**: specifies $\boldsymbol{X}_{\text{differential}}$, the state quantities that evolve from initial values according to differential equations; also defines the corresponding equations.
- **initial values**: defines the initial values of the quantities in $\boldsymbol{X}_{\text{differential}}$.
- **numerical ODE solver**: defines the algorithm used to determine the time dependence of the quantities in $\boldsymbol{X}_{\text{differential}}$; also defines the set of time points at which the state's value should be determined.

This information is sufficient to fully define and solve the dynamical system, as described in Appendix 2.

Framed in this way, BioCro II is a general tool to define and solve a dynamical system. Its framework is in fact completely free of any assumptions about the simulation's purpose. While the default module library is clearly specialized for crop growth simulations, BioCro II could in principle be used for models representing other aspects of plant biology or even models from entirely different scientific fields, provided they can be represented using the available types of evolution rules. This is not a strong constraint, although it would require reformulating any higher-order differential equations as systems of first-order differential equations and discretizing any partial differential equations.

5. DISCUSSION AND CONCLUSIONS

As a fully modular crop growth simulation software package that implements a cross-platform representation of a general dynamical system solver, BioCro II has several important advantages to offer to its users:

- Defining a model is completely separate from solving it, allowing the user to focus on biology instead of computer science.
- Model components can easily be swapped for alternative versions, to better match the available experimental inputs, to take advantage of new developments or to compare alternative components, as in the example of RUE versus FvCB.
- BioCro II modules can be developed privately, so users can withhold access to model components until after publication.
- Sensitivity analysis is straightforward to perform since all parameters are specified outside the model equations and their values can easily be changed.
- The performance and sensitivity of alternative model components can be compared in a quantitative way within the framework.
- The robustness of the simulation results can be probed by convergence tests or by comparing different solution algorithms.
- The central framework can conveniently be accessed through the R package interface or directly through C/C++, and other interfaces can be developed without duplicating the essential code.
- Improvements to the framework can proceed in parallel with improvements to the module library.
- As an open-source package, the software is free for any user.

The full impact of these properties is difficult to assess, but new possibilities are clear in three different areas: model development, sensor integration, and crop predictions.

- *Model development*: Some plant-related models, such as the transport-resistance model for carbon allocation and utilization between and within different plant organs (Thornley 1972), cannot be solved using the fixed-step

Euler method that was previously hard-wired into BioCro (and is currently hard-wired into other crop growth simulators). In this sense, separating the model equations from the numerical ODE solver is more than just a convenience to the user because it expands the space of possible model components that can be used in a simulation.

- *Sensor integration*: In BioCro II, no quantity must inherently follow a particular equation structure (parameter, driver, etc.) because the user is free to change the specifications at the start of a simulation. Consequently, a quantity whose value is determined by a model component could easily be replaced by measured data when it becomes available, facilitating the fusion of crop modelling with smart agriculture (Prathibha *et al.* 2017; Vasisht *et al.* 2017). For example, one could switch from calculating soil water content using rainfall and modelled transpiration to measuring soil water content using sensors in the field; in the terminology of BioCro II, this would simply be a change from a differential quantity to a driver and would not require making changes to any other part of the overall simulation.
- *Crop predictions*: Given weather data spanning multiple years or locations and a corresponding set of observed crop traits such as biomass, it is possible to use BioCro II to compare the predictive power of different models that may differ in just a few of their components. Not only can this process help determine which models are most appropriate for a particular type of prediction, it also provides a method for quantifying and assessing improvements in predictive power as new models and techniques are developed, similar to what has been done with weather prediction (Teweles and Wobus 1954; Kalnay *et al.* 1998).

In a larger context, there is currently an ongoing effort to develop tools for coupling models to each other in a range of fields, including earth system sciences and soil sciences (Lafolie *et al.* 2014; Barbi *et al.* 2021). The ultimate goal of this effort is to use models at different scales to answer questions that the individual models cannot address alone (Marshall-Colon *et al.* 2017). Tools such as Yggdrasil (Lang 2019) have been developed to pass information between models, primarily as a means to couple models in different programming languages. These tools will make it much easier to couple models in different languages, but passing information is not itself sufficient to ensure that models can be coupled—there must also be a shared understanding of the structure of the information. Without such a standardized design, coupling models would require in-depth knowledge of the one-of-a-kind implementation of each model. This is already a difficult process for just two models; coupling multiple models could be prohibitively complex. Moreover, if the solution methods are tightly coupled with the model specifications, there may not be a place in the code where state values can be passed between the programs, making coupling impossible.

Dynamical systems provide an ideal format to facilitate model coupling since the technical details of coupling dynamical systems are straightforward. The overall design of a coupled dynamical system is itself a dynamical system, so to combine them, a modeller simply specifies the desired sets of state quantities and evolution equations and solves the system as usual. From this point of view, BioCro II in conjunction with Yggdrasil could be a streamlined way to combine models across different languages and scales. Any model that can be expressed as one or more BioCro II modules, possibly using Yggdrasil to communicate with languages other than C/C++, could then be integrated into a larger crop growth simulation that operates across scales and disciplines with a minimum of code duplication.

There will still be code to write, but it will mostly be rote code. This process still leaves considerable work, such as identifying which equations should be coupled and which variable names in one model match the names in a second model, but thinking about the appropriate equations and quantities is the more enjoyable part of modelling. By settling on a common design, the community can reduce time spent writing the mundane parts of code and focus on the scientific parts.

Dynamical models are nearly ubiquitous as a tool to write models, and although many plant biology studies also use them, it is still commonplace to find designs that do not cleanly separate the logic of specifying models from that of solving them. BioCro II provides a way to use this common structure in a way that is modular, flexible and computationally efficient, with minimal overhead required to write the code itself. We hope that researchers find this useful for writing their own models, and that the models provided can be used by the community as is or coupled to other models.

SUPPORTING INFORMATION

The following additional information is available in the online version of this article—

Supplemental_information.pdf, which contains sections S1 (A note on the term 'state' as used in BioCro), S2 (Choosing an optimal value for α_{RUE}), S3 (Performance comparison between BioCro and BioCro II) and S4 (Comparison between BioCro II and other general purpose software for solving dynamical systems)

A_practical_guide_to_biocro.pdf, which contains basic examples of running a simulation and calculating a response curve using BioCro, along with R essentials and other information helpful to new BioCro users; this document is included with the BioCro R package as a vignette, ensuring that an up-to-date version is always available to users

Quantitative_model_comparison.pdf, which contains annotated R code used to perform the analysis in Section 3 of the main manuscript; this document is also included with the BioCro R package as a vignette, ensuring that an up-to-date version is always available to users

Quantitative_model_comparison.R, which is an R script for reproducing the analysis in Section 3 of the main manuscript; this script is automatically generated when the associated vignette is built

Biocro.tar.gz, which contains the compressed source code. The package can be installed by uncompressing the file and running 'R CMD INSTALL biocro' from the directory that contains the extracted directory. The R environment is required. Rtools is required on Windows, Xcode is required on Mac, gcc or clang is required on Linux.

ACKNOWLEDGEMENTS

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the US Department

of Agriculture (USDA). Mention of trade names or commercial products in this publication is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the USDA. USDA is an equal opportunity provider and employer.

SOURCES OF FUNDING

This work was supported, in whole or in part, by the Realizing Increased Photosynthetic Efficiency (RIPE) project, that is funded by the Bill & Melinda Gates Foundation, Foundation for Food & Agriculture Research (FFAR) and the UK Foreign, Commonwealth and Development Office (FCDO) under grant number OPP1172157. Under the grant conditions of the Bill & Melinda Gates Foundation, a Creative Commons Attribution 4.0 Generic License has already been assigned to the Author Accepted Manuscript version that might arise from this submission. The work was also supported by FFAR under award number 602757. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the Foundation for Food & Agriculture Research.

CONTRIBUTIONS BY AUTHORS

All authors contributed to conceptualization of models or the design for the new framework. J.M.M., E.B.L. and S.R. implemented source code changes for the new framework. J.M.M, E.B.L, S.R, D.J and M.L.M modified models for use with the new framework. E.B.L. and S.R. wrote source code documentation. E.B.L., J.M.M. and M.L.M drafted the manuscript, and all authors revised the manuscript.

CONFLICT OF INTEREST

None declared.

LITERATURE CITED

Arkebauer TJ, Weiss A, Sinclair TR, Blum A. 1994. In defense of radiation use efficiency: a response to Demetriades-Shah *et al.* (1992). *Agricultural and Forest Meteorology* **68**:221–227.

Augustine JA, DeLuisi JJ, Long CN. 2000. SURFRAD—a national surface radiation budget network for atmospheric research. *Bulletin of the American Meteorological Society* **81**:2341–2358.

Ball JT, Woodrow IE, Berry JA. 1987. A model predicting stomatal conductance and its contribution to the control of photosynthesis under different environmental conditions. In: Biggins J, ed. *Progress in Photosynthesis Research: Volume 4 Proceedings of the VIIth International Congress on Photosynthesis Providence*, Rhode Island, USA, 10–15 August 1986. Dordrecht, The Netherlands: Springer Netherlands, 221–224.

Barbi D, Wieters N, Gierz P, Andrés-Martínez M, Ural D, Chegini F, Khosravi S, Cristini L. 2021. ESM-Tools version 5.0: a modular infrastructure for stand-alone and coupled Earth system modelling (ESM). *Geoscientific Model Development* **14**:4051.

Boost C++ Libraries, version 1.71.0 [WWW document]. 2019. https://www.boost.org/users/history/version_1_71_0.html (1 December 2019).

Clark JS, Carpenter SR, Barber M, Collins S, Dobson A, Foley JA, Lodge DM, Pascual M, Pielke R Jr, Pizer W, Pringle C, Reid WV, Rose KA, Sala O, Schlesinger WH, Wall DH, Wear D. 2001. Ecological forecasts: an emerging imperative. *Science* **293**:657–660.

Demetriades-Shah TH, Fuchs M, Kanemasu ET, Flitcroft I. 1992. A note of caution concerning the relationship between cumulated intercepted solar radiation and crop growth. *Agricultural and Forest Meteorology* **58**:193–207.

Farquhar GD, von Caemmerer S, Berry JA. 1980. A biochemical model of photosynthetic CO_2 assimilation in leaves of C_3 species. *Planta* **149**:78–90.

Flannery BP, Press WH, Teukolsky SA, Vetterling W. 1992. *Numerical recipes in C*. New York: Press Syndicate of the University of Cambridge.

Humphries SW, Long SP. 1995. WIMOVAC: a software package for modelling the dynamics of plant leaf and canopy photosynthesis. *Computer Applications in the Biosciences* **11**:361–371.

Illinois State Water Survey. [WWW document]. 2015. Water and atmospheric resources monitoring program. Illinois Climate Network. doi:10.13012/J8MW2F2Q (7 September 2021).

Jaiswal D, De Souza AP, Larsen S, LeBauer DS, Miguez FE, Sparovek G, Bollero G, Buckeridge MS, Long SP. 2017. Brazilian sugarcane ethanol as an expandable green alternative to crude oil use. *Nature Climate Change* **7**:788–792.

Kalnay E, Lord SJ, McPherson RD. 1998. Maturity of operational numerical weather prediction: medium range. *Bulletin of the American Meteorological Society* **79**:2753–2770.

Lafolie F, Cousin I, Marron PA, Mollier A, Pot V, Moitrier N, Moitrier N, Nouguier C. 2014. The «VSOIL» modeling platform. *Revue Forestière Française* **66**:187.

Lang M. 2019. Yggdrasil: a Python package for integrating computational models across languages and scales. *In Silico Plants* **1**:diz001; doi:10.1093/insilicoplants/diz001.

Larsen S, Jaiswal D, Bentsen NS, Wang D, Long SP. 2016. Comparing predicted yield and yield stability of willow and *Miscanthus* across Denmark. *GCB Bioenergy* **8**:1061.

LeBauer DS, Wang D, Richter KT, Davidson CC, Dietze MC. 2013. Facilitating feedbacks between field measurements and ecosystem models. *Ecological Monographs* **83**:133–154.

Lochocki EB, McGrath JM. 2021. Integrating oscillator-based circadian clocks with crop growth simulations. *In Silico Plants* **3**:diab016; doi:10.1093/insilicoplants/diab016.

Marin FR, Ribeiro RV, Marchiori PER. 2014. How can crop modeling and plant physiology help to understand the plant responses to climate change? A case study with sugarcane. *Theoretical and Experimental Plant Physiology* **26**:49–63.

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, Shrivastava S, Shukla D, Srinivasan V, Stitt M, Turk MJ, Voit EO, Wang Y, Yin X, Zhu XG. 2017. Crops in silico: generating virtual crops using an integrative and multi-scale modeling platform. *Frontiers in Plant Science* **8**:786.

Matthews ML, Marshall-Colón A, McGrath JM, Lochocki EB, Long SP. 2021. Soybean-BioCro: a semi-mechanistic model of soybean growth. *In Silico Plants* 4(1):diab032; doi:10.1093/insilicoplants/diab032.

Menon S, Denman KL, Brasseur G, Chidthaisong A, Ciais P, Cox PM, Dickinson RE, Hauglustaine D, Heinze C, Holland E, Jacob D,

Lohmann U, Ramachandran S, Leite da Silva Dias P, Wofsy SC, Zhang X. 2007. *Couplings between changes in the climate system and biogeochemistry (no. LBNL-464E)*. Berkeley, CA: Lawrence Berkeley National Lab. (LBNL).

Miguez FE, Maughan M, Bollero GA, Long SP. 2012. Modeling spatial and dynamic variation in growth, yield, and yield stability of the bioenergy crops *Miscanthus* × *giganteus* and *Panicum virgatum* across the conterminous United States. *GCB Bioenergy* **4**:509–520.

Miguez FE, Zhu X, Humphries S, Bollero GA, Long SP. 2009. A semimechanistic model predicting the growth and production of the bioenergy crop *Miscanthus* × *giganteus*: description, parameterization and validation. *GCB Bioenergy* **1**:282–296.

NOAA. 2021. Global Monitoring Laboratory—carbon cycle greenhouse gases [WWW document]. https://gml.noaa.gov/ccgg/trends/data.html (7 April 2021).

Postma JA, Kuppe C, Owen MR, Mellor N, Griffiths M, Bennett MJ, Lynch JP, Watt M. 2017. OpenSimRoot: widening the scope and application of root architectural models. *The New Phytologist* **215**:1274–1286.

Prathibha SR, Hongal A, Jyothi MP. 2017. IOT based monitoring system in smart agriculture. In: Guerrero JE, ed. *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT). Presented at the 2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)*. Piscataway, NJ: The Institute of Electrical and Electronics Engineers, 81–84.

Rogers A, Humphries SW. 2000. A mechanistic evaluation of photosynthetic acclimation at elevated CO_2. *Global Change Biology* **6**:1005.

Sinclair TR, Muchow RC. 1999. Radiation use efficiency. In: Sparks DL, ed. *Advances in agronomy*. San Diego, CA: Academic Press, 215–265.

Song Q, Chen D, Long SP, Zhu XG. 2017. A user-friendly means to scale from the biochemistry of photosynthesis to whole crop canopies and production in time and space—development of Java WIMOVAC. *Plant, Cell & Environment* **40**:51–55.

Teweles S, Wobus HB. 1954. Verification of prognostic charts. *Bulletin of the American Meteorological Society* **35**:455–463.

Thornley JHM. 1972. A model to describe the partitioning of photosynthate during vegetative plant growth. *Annals of Botany* **36**:419–430.

Vasisht D, Kapetanovic Z, Won J, Jin X, Chandra R, Sinha S, Kapoor A, Sudarshan M, Stratman S. 2017. FarmBeats: an IoT platform for data-driven agriculture. In: Akella A, Howell J, eds. *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, 515–529.

Wang D, Jaiswal D, LeBauer DS, Wertin TM, Bollero GA, Leakey AD, Long SP. 2015. A physiological and biophysical model of coppice willow (*Salix* spp.) production yields for the contiguous USA in current and future climate scenarios. *Plant, Cell & Environment* **38**:1850–1865.

Wittig VE, Bernacchi CJ, Zhu XG, Calfapietra C, Ceulemans R, Deangelis P, Gielen B, Miglietta F, Morgan PB, Long SP. 2005. Gross primary production is stimulated for three *Populus* species grown under free-air CO_2 enrichment from planting through canopy closure. *Global Change Biology* **11**:644–656.

Zhu XG, Wang Y, Ort DR, Long SP. 2013. e-Photosynthesis: a comprehensive dynamic mechanistic model of C_3 photosynthesis: from light capture to sucrose synthesis. *Plant, Cell & Environment* **36**:1711–1727.

APPENDIX 1. A SHORT INTRODUCTION TO THE SYNTAX OF BIOCRO II

To run models built from modules already written in BioCro, one does not need to understand many of the code details. The essential parts are understanding how to create the objects in R that are passed to the `run_biocro` function. Initial values and parameters are given as list objects as follows.

```
parameters = list(
     alpha_rue = 0.07,   # kg / mol
     SLA = 25,           # m^2 / kg
     C_conv = 0.03,      # kg / mol
     f_leaf = 0.2,       # kg / kg
     f_root = 0.8,       # kg / kg
     timestep = 1        # s
)
initial_values = list(
     Leaf = 1,           # kg
     Root = 1            # kg
)
```

The drivers are given as a data frame, which must include the times at which their values are defined. Although drivers are typically derived from experimental measurements rather than closed-form equations, for simplicity the drivers for the example model are defined as follows:

```
Q = function(time) sin(time/3600/12 * pi)
* 2000e-6 # mol / m^2 / s
times = 0:(3600 * 12) # seconds
light_intensity = data.frame(time = times,
Q = Q(times))
```

The final two arguments of `run_biocro` accept the names of sets of equations that are already written. A library of sets of equations is provided with BioCro II. Each set is called a `module`, and `modules` come in two types, direct and differential, that can calculate either instantaneous values or time rates of change, respectively (see Section 4 of the main text). The example model in Section 1.1 of the main text has one of each type of `module`. If one wants to use multiple `modules` in a model, the names of the same type are grouped into separate lists. For example, if there are direct modules 'A' and 'B', they would be passed as `list('A', 'B')`.

To add new models, one must write a module. Modules are written in C++, specifying the input and output variables of the module, as well as the equations. Due to language requirements, the input and output variables are repeated in several places. There are plans to handle the repetition better. The full source code for `example_model_carbon_gain` is as follows.

```
#ifndef EXAMPLE_MODEL_MASS_GAIN_H
#define EXAMPLE_MODEL_MASS_GAIN_H

#include "../modules.h"
#include "../state_map.h"

/**
 * @class example_model_mass_gain
 *
 * @brief An example for the BioCro II manuscript
 *
 * ### Model overview
 *
 * Model mass gain as a linear function of other quantities.
 *
 * It is not meant to be biologically realistic.
 *
 */
class example_model_mass_gain : public direct_module
{
   public:
    example_model_mass_gain(
        state_map const& input_quantities,
        state_map* output_quantities)
        :  // Define basic module properties by passing its name to its
parent class
          direct_module{"example_model_mass_grain"},

          // Get pointers to input quantities
          Q{get_input(input_quantities, "Q")},
          alpha_rue{get_input(input_quantities, "alpha_rue")},
          SLA{get_input(input_quantities, "SLA")},
          C_conv{get_input(input_quantities, "C_conv")},
          Leaf{get_input(input_quantities, "Leaf")},

          // Get pointers to output quantities
          mass_gain_op{get_op(output_quantities, "mass_gain")}
    {
    }
    static string_vector get_inputs();
    static string_vector get_outputs();

   private:
    // References to input quantities
    double const& Q;
    double const& alpha_rue;
    double const& SLA;
    double const& C_conv;
    double const& Leaf;

    // Pointers to output quantities
    double* mass_gain_op;

    // Main operation
    void do_operation() const;
};

string_vector example_model_mass_gain::get_inputs()
{
    return {
        "Q",            // mol / m^2 / s
        "alpha_rue",    // mol / mol
        "SLA",          // m^2 / kg
        "C_conv",       // kg / mol
        "Leaf"          // kg
    };
}

string_vector example_model_mass_gain::get_outputs()
{
    return {
        "mass_gain"  // kg / s
    };
}

void example_model_mass_gain::do_operation() const
{
    double const assimilation = Q * alpha_rue;
    double const mass_gain = assimilation * Leaf * SLA * C_conv;

    update(mass_gain_op, mass_gain);
}

#endif
```

The code declares a new class `example_model_mass_gain` that inherits the `direct_module` class. The constructor accepts two collections of quantities and their values. These are represented as state maps, a collection of key-value pairs of quantity names and values. Within the constructor, references to the items in the map are stored within the class, which improves speed. The input and output quantities are listed in the `get_inputs()` and `get_output()` functions. These are used to validate the model when assembling multiple modules. Lastly, the equations are written in the `do_operation()` function. Calculations are local to the function, so `update(quantity, local_value)` is used to update values in the model's state map.

For more in-depth information about using BioCro II, see *A Practical Guide to BioCro* in the Supporting Information and the up-to-date documentation available online (https://ebimodeling.github.io/biocro-documentation/).

APPENDIX 2. BIOCRO'S METHOD FOR SOLVING DYNAMICAL SYSTEMS

As described in the text, each argument of the `run_biocro` R function specifies one or more parts of a dynamical system. Assuming the set of equations is solvable, these arguments are sufficient to define and solve the dynamical system. To understand how, note that for given values of t and $\boldsymbol{X}_{\text{differential}}$, it is possible to determine the instantaneous derivatives for the quantities in $\boldsymbol{X}_{\text{differential}}$ as follows:

1. The values of the quantities in $\boldsymbol{X}_{\text{parameter}}$ are given in the original input.
2. The values of the quantities in $\boldsymbol{X}_{\text{driver}}$ are determined from the current value of t.
3. The values of the quantities in $\boldsymbol{X}_{\text{direct}}$ are determined from the values of t and the other state quantities $(X_{\text{parameter}} \bigcup X_{\text{driver}} \bigcup X_{\text{differential}})$ using the equations specified by the direct modules.
4. The derivatives of the quantities in $\boldsymbol{X}_{\text{differential}}$ are determined from the values of t and the state quantities $(X_{\text{parameter}} \bigcup X_{\text{driver}} \bigcup X_{\text{differential}} \bigcup X_{\text{direct}})$ using the equations specified by the differential modules.

This process determines a function $g(t, \boldsymbol{X}_{\text{differential}})$ which calculates $d\boldsymbol{X}_{\text{differential}}/dt$. In turn, this function can be passed to a numerical ODE solver, which can use it to determine the time evolution of $\boldsymbol{X}_{\text{differential}}$. Finally, the time evolution of the state as a whole can be determined using steps 1–3 above. This process—arranging the equations in the order described here to create this function, passing the function to a numerical ODE solver and determining the state at all desired times—is a major part of what the BioCro II framework does.