Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

TreeSim: An object-oriented individual tree simulator and 3D visualization tool in Python

Abbas Nabhani [*], Hanne K. Sjølie

Department of Forestry and Wildlife Management, Faculty of Applied Ecology, Agricultural Sciences and Biotechnology, Inland Norway University of Applied Sciences, P.O. Box, 2480, Koppang, Norway

ARTICLE INFO

Article history:
Received 8 July 2022
Received in revised form 5 September 2022
Accepted 29 September 2022

Keywords:
Large scale simulation
Forest dynamics
Forest growth modelling
Ecosystem services
Forest carbon

ABSTRACT

TreeSim is an open-source, user-extendable framework that offers new opportunities for users to model, simulate, visualize, and animate the dynamics of forest. It provides a general environment for modellers to implement and evaluate forest growth models. In this paper, we introduce the object-oriented architecture for our proposed simulator and examine its performance to model and simulate forest growth, management, and dynamics. The simulator uses various models to predict the dynamics and attributes of forest (e.g., regeneration, increment, mortality, biodiversity, biomass, carbon, dead wood). Finally, TreeSim can be integrated into a simulation–optimization framework for decision-making support.

Code metadata

Current code version	v0.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00188
Permanent link to reproducible capsule	https://codeocean.com/capsule/8889864/tree
Legal code license	Apache License 2.0
Code versioning system used	git
Software code languages, tools and services used	Python, vtk, matplotlib
Compilation requirements, operating environments and dependencies	Python v3.4.x and higher, vtk, django, django-utils, tqdm, vtk, moviepy, shutil, xlrd, seaborn
If available, link to developer documentation/manual	https://github.com/AbbasNabhani/TreeSim/tree/master/docs
Support email for questions	forsim@inn.no

1. Motivation and significance

Forest ecosystems are dynamic entities with attributes and characteristics that greatly vary in time and space. The main component in these ecosystems, trees, may be subject to management, with the most common pivotal aim being wood supply. This management alongside natural disturbances lead to changes in the ecosystems' characteristics, that in turn shift not only the quality and quantity of supplied wood, but many other ecosystem services crucial for human wellbeing [1]. The dynamics of tree regeneration, growth, and mortality together with management is complex and understanding of how forests may develop in the future requires simulation models. Forest growth simulators (FGSs) have been widely implemented to provide support for sustainable forest management decision-making [2,3], in addition to global warming mitigation [4] and assess the impacts of forest policies [5]. In addition to the fundamental dynamics like growth and species interactions, mortality, and regeneration, FGSs may encompass attributes like harvest of wood and non-wood products, deadwood decomposition, carbon dynamics, biomass, water balance, and conservation of biodiversity resulting from management practices and/or disturbances. In its simplest form, a FGS consists of a set of models, like functions for ingrowth, diameter and height growth and mortality that are implemented and integrated into a computer program/tool.

Abbreviations: FGS, Forest Growth Simulator; ITS, Individual Tree Simulator; VTK, Visualization toolkit; DBH, Diameter at Breast Height; FMA, Forest Management Alternative

* Corresponding author.
E-mail address: abbas.nabhani@inn.no (Abbas Nabhani).

https://doi.org/10.1016/j.softx.2022.101221

FGSs may be divided into stand-level and individual-tree simulators (ITSs). While stand-level simulators have been widely applied and are fast [6], they are limited by the fact that they project the mean tree in the management unit (stand) and thus ignore intra-stand variation in age and size. While this assumption may hold for monoculture stands managed mainly for timber, tree variation and coarse woody debris is crucial for non-wood ecosystem services like biodiversity [7]. Current regulations dictate measures to increase stand complexity, like retention trees to support biodiversity [8]. For instance in Norway, more than 40% of the productive forest area is natural forests that have not been clear-cut [9] and thus entail considerably within-stand variation calling for ITSs to make more accurate projections.

There exist already a handful of ITS, including FVS [10], SILVA [11], Capsis [12], TREEGROSS [13], SPATE-HPC [14], SExI-FS [15], Heureka [16,17] and SiTree [18]. Although growth and yield projections are the core of all these platforms and tools, they vary in structure and simulation capabilities of the forest dynamics (empirical, process-based, distance-independent tree level, distance-dependent tree level, whole-stand, etc.). For example, individual tree growth and yield may be projected through an imputation/copula-based approach [19] and by a distance-dependent individual-based model [11]. In addition, some of the FGSs benefit from visualization techniques to depict the simulated results [10,15].

In order to achieve faster simulations than interpreted languages, ITSs are mostly developed using compiled languages such as C, C++, Fortran or Java (the latter does both compilation and interpretation) or using a combination of these languages. However, they require extra memory and time to run the executable file. Recently the cross-platform, open-source framework SiTree [16] has been developed using R programming, with the advantage that R is widely used by forestry scientists. From our point of view, R as a domain-specific language is not robust and applicable in large-scale simulation and 3D visualization. Despite the number of existing FGSs most are not readily transferable across countries as growth models and national forest inventory data typically are country specific. In addition, most FGSs are not open source.

To address these shortcomings, we propose a new FGS, TreeSim, written in Python using interpreted high-level programming language. Python [20] is known as general-purpose language widely applied in multiple fields including forestry by non-programmers and researchers. The simulator can be used for smaller forest areas (like property or municipalities) or likewise for larger areas (like regions or countries), with the needed initial forest inventory data.

TreeSim, a pure Python ITS, uses an object-oriented design where for each plot and tree an instance of a generic class of plot and tree respectively were defined. By applying models, we simulate plot and tree objects to mimic the dynamics of forest and obtain a coherent dynamic representation of the complex natural environment. Unlike most other ITSs, which focus on simulating the growth, mortality, regeneration, and management, TreeSim simulates all the following characteristics: the growth in height and diameter at breast height (DBH), wood volumes of growing stock (live trees), deadwood and harvest, all by species, in addition to above and below-ground carbon storage in living tree biomass, deadwood carbon, soil carbon and biodiversity. Not at least, it simulates a range of the most common forest management methods currently used in Norway. Another important capability is the visualization of the simulated results and implementation of the state-of-the-art 3D computer graphic algorithms using the visualization toolkit (VTK) [21], an open source and powerful graphics library with extensive sophisticated visualization algorithms.

While the code is original, some ideas are taken from the [22]. TreeSim uses the programming paradigm to generate plot-level tables and implement management.

This paper is arranged as follows. In Section 2, we describe the components of the framework and their implementation details and in Section 3 we provide some code illustrating the framework with examples.

2. Software description

The proposed simulator is an object-oriented simulator and entirely written in Python. To make the codes function-oriented, the simulator is decomposed into a set of interacting Python modules. The datasets required to run a simulation in TreeSim are tree-level data, plot-level data, and meteorological data. To convert the datasets into meaningful state and store the variables in tree and plot objects, the simulator processes and manipulates the data in simulation steps. The simulator is designed in modular and object-oriented form which makes it user-centric and easily extensible. TreeSim performs the simulations based on a set of site indices specified in Main and simulation modules by users. This enables users to apply either serial or parallel computing within a simulation. Using parallel computing requires to split the simulator to independent problems which can maximize the CPU utilization and reduce the simulation time. Fig. 1 depicts the operation of the simulator when serial and parallel computing applied in TreeSim.

Here we provide information on TreeSim's characteristics, the architecture of the simulator, and briefly discuss its functionalities.

2.1. Software architecture

TreeSim comprises six core modules (Main, simulation, Tree_Models, Plot_Models, yasso, and titles_tree) and three visualization modules (animation, graph_animator, timer_animator). UML class diagram of TreeSim v0.1.0 is shown in Fig. 2.

2.1.1. TreeSim core modules

The core modules of TreeSim are the following:

- Main: this module contains site index setting, scenario setting, and initialization of tree and plot objects.
- simulation: this module contains five classes: Class Data that reads and stores the inventory data in two classes; tree class for storing tree-level data into the individual objects called trees, and plot class for storing plot-level data into the individual objects named plots. Management_prescription class is used to specify forest management prescriptions and simulator class is implemented to simulate stand tables based on individual-tree growth models.
- Tree_models: it contains all the models required for projections (diameter increment, height increment, mortality, regeneration, management alternatives, biodiversity, carbon and biomass, dead wood, etc.). This module does all the heavy computations.
- Plot_models: this module initializes and computes plot-level variables (growing stock, soil carbon, basal area, etc.)
- yasso: implementation of the soil carbon model yasso [23]
- titles_tree: it provides a format for the tree-level output file
- animation, graph_animator, and timer_animator: process of creating visualization of data using VTK library (see Fig. 3)

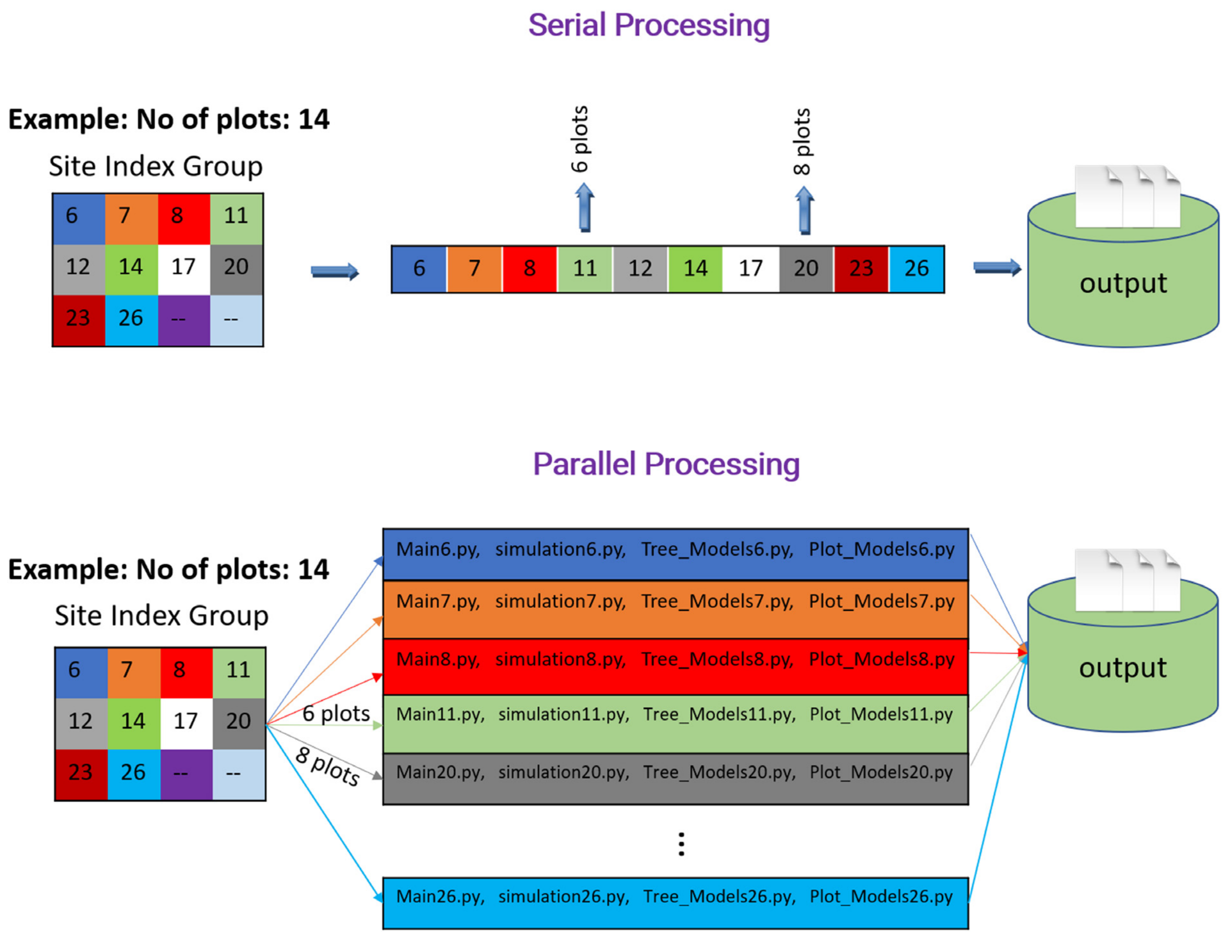
Serial Processing
Example: No of plots: 14
Site Index Group
6 7 8 11
12 14 17 20
23 26 -- --
6 plots
8 plots
6 7 8 11 12 14 17 20 23 26
output
Parallel Processing
Example: No of plots: 14
Site Index Group
6 7 8 11
12 14 17 20
23 26 -- --
6 plots
8 plots
Main6.py, simulation6.py, Tree_Models6.py, Plot_Models6.py
Main7.py, simulation7.py, Tree_Models7.py, Plot_Models7.py
Main8.py, simulation8.py, Tree_Models8.py, Plot_Models8.py
Main11.py, simulation11.py, Tree_Models11.py, Plot_Models11.py
Main20.py, simulation20.py, Tree_Models20.py, Plot_Models20.py
Main26.py, simulation26.py, Tree_Models26.py, Plot_Models26.py
output

Fig. 1. Serial processing: All modules are executed for each site index in a sequence. Parallel processing: All modules are executed for multiple site indices simultaneously.

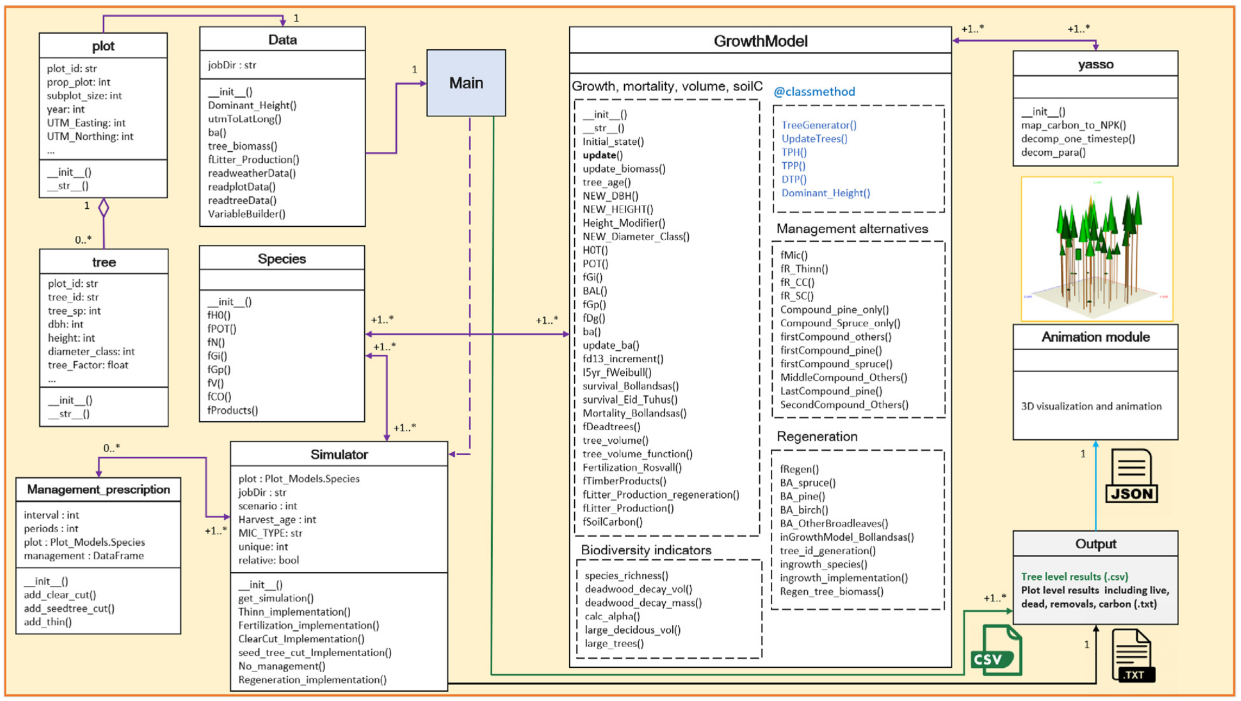

Fig. 2. UML class diagram of TreeSim components and their associations. Each block represents a class which is a code template for creating objects, and headings (except Output and Animation module) representing the class name.

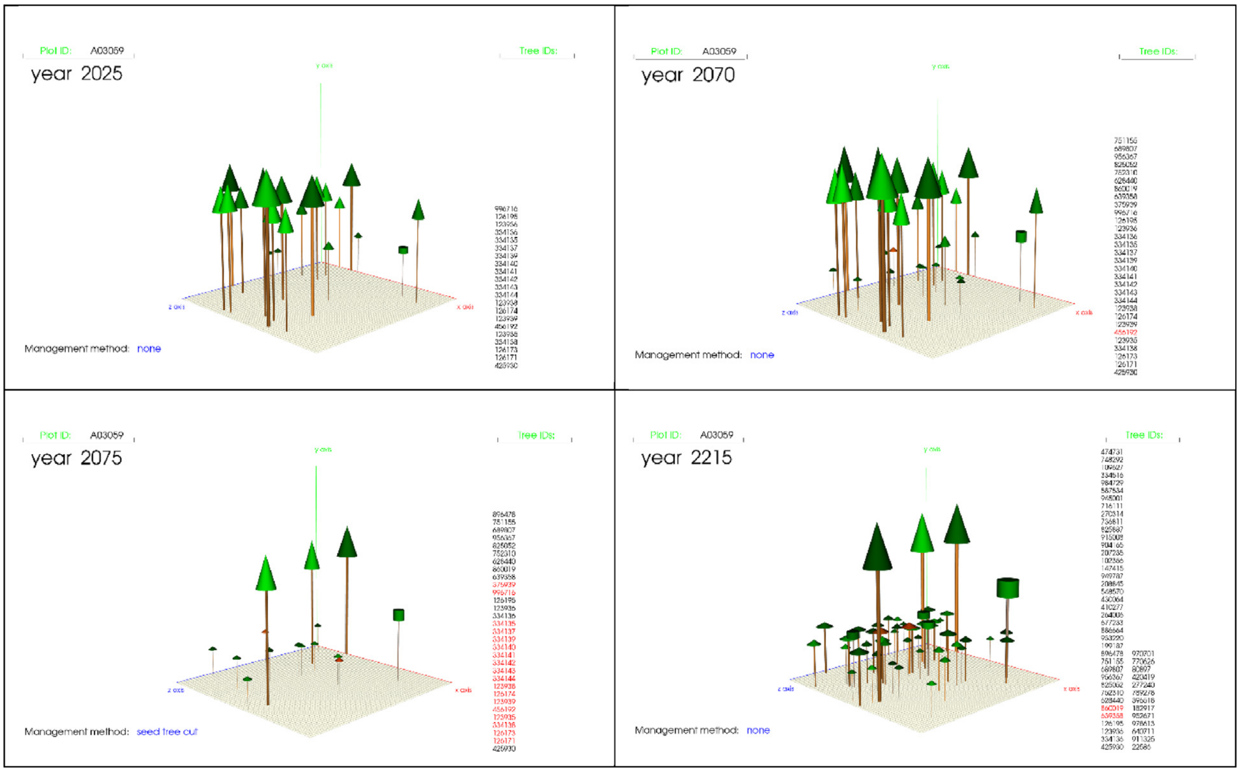

Fig. 3. 3D graphic tree plotting. The trees with red crown and red tree ID represent either removed trees or standing dead trees. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

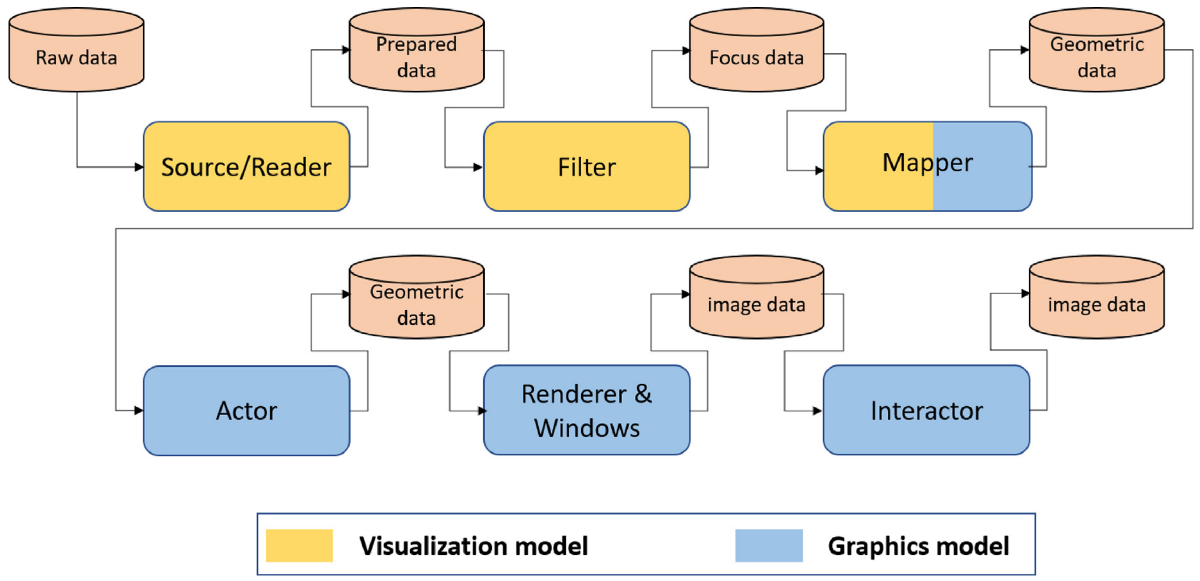

Fig. 4. The visualization pipeline architecture.

2.1.2. *Brief description of VTK*

VTK is a widely used open-source, cross-platform C++ library, and freely available software for visualization of different kinds of data [21,24,25]. VTK has an object-oriented design with several interpreted language wrappers including Python. However, internally VTK is entirely written in C++.

VTK is based on a visualization pipeline that transforms geometric data obtained from a source (either generating data or reading from a file) into an image rendered on the screen (graphical data). The VTK pipeline architecture can be seen in Fig. 4 which is divided into two main phases, the visualization processing, and graphics processing. The main components of the VTK pipeline are:

- Source/Reader: it provides data input from files or generates data.
- Filter(s): the data from source can be modified (e.g., reduction, merging, interpolation, etc.)
- Mapper: the data are mapped onto objects that can be rendered by VTK. This step is a transition layer from data processing to rendering segment.

- Actors: it represents an object (geometry plus display properties) within the scene. In this step, programmers can adjust the visible properties (opacity, colour mapping, shading, etc.)
- Renderers & Windows: geometric data are transferred to image data
- Interactors: it allows users to rotate/zoom/pan the camera, select and manipulate actors, etc.

2.2. Software functionalities

As outlined above, TreeSim provides an object-oriented framework for simulating the dynamics of forest. In the following, the core TreeSim modules accompanied with code examples are described in detail.

2.2.1. Module: Main

This module provides a functionality to simulate plots and enables users to generate multiple scenarios. To perform these tasks and simulate each plot and each scenario, importlib.reload reloads the previously imported modules (simulation, Tree_Models, and Plot_Models).

```
# Example code for reloading the modules
import importlib
import simulation
import Tree_Models
import Plot_Models

importlib.reload(simulation)
importlib.reload(Tree_Models)
importlib.reload(Plot_Models)
```

Example code for reloading the previously imported modules

Simulating plots with no recorded tree (if tree-level data are unavailable) in forest inventory data was handled in both Main and simulation modules. If there are no trees in a plot at period 0 (the initial state), the simulator will automatically regenerate one tree in the plot at period 0 to start the simulation and thus assure the functionality of the growth models.

2.2.2. Module: simulation

simulation module prepares data objects by loading the data input from files located in input folder and storing them in tree and plot objects. The module constructs some new variables for the initial state of trees and plots using methods such as utmToLatLong(), tree_biomass(), etc. This module enables detection of insufficient input data and generates tree level data if the plot has no initial state of trees.

The implementation of object of class Simulator which is the heart of the simulation starts with a loop for each plot and then for each scenario.

```
# Example code to run a simulation
prob = simulation.Data(jobDirectory)
plot = Plot_Models.Species(Data= prob,
year=prob.plots[x].year, plot_id=prob.plots[x].plot_id,
t=prob.plots[x].stand_age_years, H40=prob.plots[x].SI_m,
N=prob.plots[x].N_tree_ha,
treeList=prob.plots[x].treeList,
Latitude=prob.plots[x].Latitude, Region=prob.plots[x].region,
Altitude=prob.plots[x].altitude_m,
subplot_size=prob.plots[x].subplot_size, mortality=True)
simrun = simulation.Simulator(plot = plot,
jobDir=jobDirectory, scenario=scenario,
Harvest_age=Each_period, MIC_TYPE=MIC_type, unique=uniqueNO,
YearN=True, Data= prob)
```

Example code to run a simulation

TreeSim also enables users to simulate multiple management (e.g., planting, thinning, fertilization, clear cut, seed tree cut) at different phases of the development of a plot or a group of trees defined as forest management alternatives (FMAs). By default, no management is always an option.

Within any given FMA, a number of criteria have been implemented in both simulation and Tree_models modules.

2.2.3. Module: Tree_models

All individual-based models such as increment etc. and FMAs are implemented in Tree_models module. This module relies on dictionary object using class method to store and update the values in a structured (key:value) pairs for each period. Tree_models module is flexibly extendable and allows advanced users to add functionality and adapt the framework to their specific needs.

2.2.4. Module: Plot_models

Plot_models module develops plot table projections which include growing stocks by species group (by default six groups), per hectare basal area, number of trees, above and below ground biomass, soil organic carbon, deadwood, carbon stock, and biodiversity indicators. We projected plot-level impacts of FMA on biodiversity by assigning a score between 0 and 100 for each of the five indicators species richness, Shannon index, deadwood volume, large deciduous trees volume and the number of large trees per hectare. Next, we calculated the arithmetic mean of these five scores to obtain the biodiversity level for each plot.

2.2.5. Module: yasso

The decomposition and dynamics of soil carbon of TreeSim is based on Yasso soil model and implemented in yasso module to project the amount of soil organic carbon over time [23].

2.2.6. Modules: animation, graph_animator, and timer_animator

To visualize the simulated output in 3D, the open-source library VTK was used in the following six steps. (1) Converting the simulated tree data to JSON file and importing into tree objects. (2) Feeding the data through filters (e.g., geometry filter, transform filter, and Transform Poly Data filter). (3) Mapping data onto objects for rendering. (4) Packing data into Cone and Cylinder actors. (5) The actors were added to renderers and the renderers were added to a render window. (6) Interactors were introduced and connected to the render window.

3. Illustrative examples

We created an input data consists of 14 sample plots having site index H40 = 11 and H40 = 20, representing the most common productivity level in Norwegian forests (11) and high-productive sites (20). We used this dataset to demonstrate the capabilities and components of TreeSim. To speed up the performance time, we used parallel processing which simultaneously break up and run the simulation tasks. By parallelizing the simulation tasks, the execution time for this case study was notably reduced. The total computational times of the proposed simulator using serial and parallel processing on two different computer systems are illustrated in Table 1.

Table 1
Comparison of computational times for two different methods and computer systems.

Computer information	Computational times	
	Serial processing	Parallel processing
MacBook Pro, Intel Core i5 CPU at 2.9 GHz, RAM 8 GB, 64-bit Windows 10 (OS)	32–33 min	21–22 min
MacBook Pro, M1 Max 10-core CPU, RAM 64 GB, Mac OS	16–17 min	10–11 min

The results of 5 sample plots (out of 14) were plotted. Fig. 5a displays the key results with no management for 200 years (40 periods) including mean height of the dominant trees in a plot (m), number of trees per hectare (stems/ha), total plot basal area (m^2/ha), total growing stock (m^3/ha), total living trees carbon (t C/ha), total dead wood carbon (t C/ha), soil organic carbon (t C/ha), and biodiversity level (index). In addition, development of one sample plot with management was projected (Fig. 5b, Table 2).

4. Impact

TreeSim was developed with the aim to be an efficient and flexible FGS readily available for non-programmers to be used as

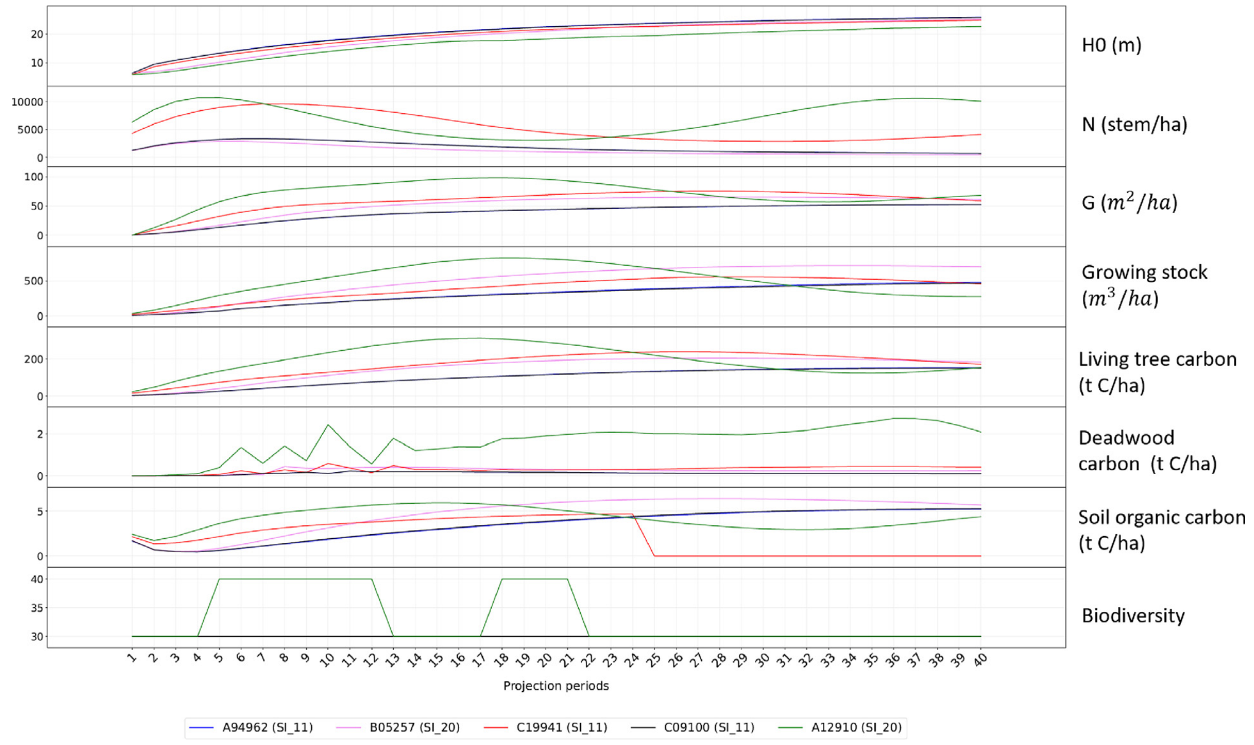

Fig. 5a. Plot-level simulation results for five sample plots.

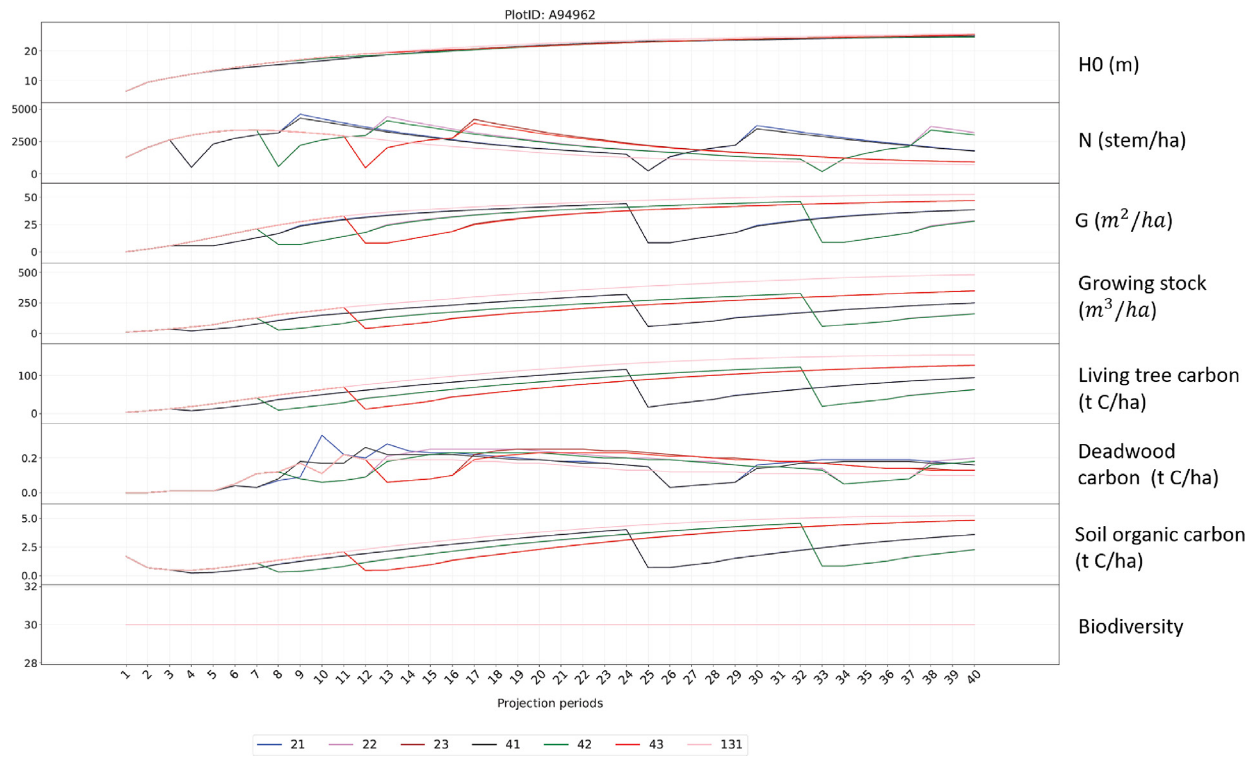

Fig. 5b. Simulation results for sample plot ID A94962 under different management scenarios (see TreeSim tool documentation for scenario definitions).

Table 2
Definition of management scenarios in Fig. 5b.

Scenario	Minimum final harvest age	Harvest method	Natural regeneration species/density	Thin
21	80	Seed-tree cut	Pine/1800	No
22	100	Seed-tree cut	Pine/1800	No
23	120	Seed-tree cut	Pine/1800	No
41	80	Seed-tree cut	Pine/1200	No
42	100	Seed-tree cut	Pine/1200	No
43	120	Seed-tree cut	Pine/1200	No
131	–	None	-/0	No

a stand-alone or in combination with other model/optimization frameworks. The following major impacts are expected:

- This powerful open-source tool can be easily extended or modified; TreeSim provides a framework for modelling of forest dynamics, and advance visualization features which may be used by academia, companies, policy makers, or whoever benefits from TreeSim's open-source availability, flexibility, and robustness. This saves the time required for coding and data manipulation.
- Since the modelling of natural environment like forest is very complex, visualization can provide a better understanding of this complex process.
- TreeSim includes well-documented models that drive forest dynamics which can be replaced with other models.
- TreeSim takes advantage of the most efficient techniques like parallel computing, to run the simulations in acceptable time.
- TreeSim has an object-oriented structure consisting of multiple Python modules which allow for easier debugging and reuse of codes through inheritance.
- Our proposed simulator is completely flexible on the size and the number of plots.
- A major advantage of TreeSim is the outputs that include a variety of economic and ecological indicators and stand structure. TreeSim output can readily serve as input in an optimization model to find the optimal management given prices of wood and other considerations [26,27]. Furthermore, it can be used in a forest sector model of forestry and wood industries to analyse interactions between wood markets, policies, and forest management [28,29].

5. Conclusions

This paper has described TreeSim, a new FGS that effectively provides support to research about the complexity of forest ecosystem services and to decision-making in sustainable forest management. The overall aim behind the TreeSim development was to obtain an efficient and flexible FGS by code and models optimization; that is open-source and is readily available for anyone to use and further develop. The simulator is based on an object-oriented structure and provides a framework to modellers for implementing and evaluating their models; it also visualizes and animates forest dynamics. Our simulator has undergone extensive testing and validation during the development process and has been applied for case studies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is funded by Inland Norway University of Applied Sciences.

References

[1] Duncker P, et al. How forest management affects ecosystem services, including timber production and economic return: Synergies and trade-offs. Ecol Soc 2012;17. http://dx.doi.org/10.5751/ES-05066-170450.
[2] Perin J, Pitchugin M, Hébert J, Brostaux Y, Lejeune P, Ligot G. SIMREG, a tree-level distance-independent model to simulate forest dynamics and management from national forest inventory (NFI) data. 2021, http://dx.doi.org/10.1016/j.ecolmodel.2020.109382.
[3] Crookston NL, Dixon GE. The forest vegetation simulator: A review of its structure, content, and applications. Comput Electron Agric 2005;49(1):60–80. http://dx.doi.org/10.1016/j.compag.2005.02.003.
[4] Seppälä J, et al. Effect of increased wood harvesting and utilization on required greenhouse gas displacement factors of wood-based products and fuels. J Environ Manag 2019;247:580–7. http://dx.doi.org/10.1016/j.jenvman.2019.06.031.
[5] Qin H, Dong L, Huang Y. Evaluating the effects of carbon prices on trade-offs between carbon and timber management objectives in forest spatial harvest scheduling problems: A case study from northeast China. Forests 2017;8(2):2. http://dx.doi.org/10.3390/f8020043.
[6] Eriksson LO, Bergh J. A tool for long-term forest stand projections of Swedish forests. Forests 2022;13(6):6. http://dx.doi.org/10.3390/f13060816.
[7] Rosenvald R, Lõhmus A, Kraut A, Remm L. Bird communities in hemiboreal old-growth forests: The roles of food supply, stand structure, and site type. For Ecol Manag 2011;262(8):1541–50. http://dx.doi.org/10.1016/j.foreco.2011.07.002.
[8] PEFC. Norsk PEFC skogstandard. 2022, https://pefc.no/vare-standarder/norsk-pefc-skogstandard [Accessed 05 Jul. 2022].
[9] Storaunet KO, Rolstad J. Naturskog i norge. en arealberegning basert På bestandsalder I landsskogtakseringens takstomdrev fra 1990 til 2016. NIBIO; 2020, [Online]. Available: https://nibio.brage.unit.no/nibio-xmlui/handle/11250/2650496 [Accessed: 05 Jul 2022].
[10] Dixon GE, Dixon GE. Essential FVS: a user's guide to the forest vegetation simulator. Internal report, Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Forest Management Service Center; 2002, p. 189, 2002.
[11] Pretzsch H, Biber P, Ďurský J. The single tree-based stand simulator SILVA: construction, application and evaluation. For Ecol Manag 2002;162(1):3–21. http://dx.doi.org/10.1016/S0378-1127(02)00047-6.
[12] Dufour-Kowalski S, Courbaud B, Dreyfus P, Meredieu C, de Coligny F. Capsis: an open software framework and community for forest growth modelling. Ann for Sci 2012;69(2):221–33. http://dx.doi.org/10.1007/s13595-011-0140-9.
[13] Nagel J, Nagel J. TreeGrOSS: Tree growth open source software—a tree growth model component. göttingen. In: Germany: niedersächsischen forstlichen versuchsanstalt, abteilung waldwachstum. 2003, 2003. URL http://treegrosssourceforge.net/treegross.pdf https://docplayer.org/10289037-Treegross-tree-growth-open-source-software-a-tree-growth-model-component-treegross-jar-data-generation-basic-stand-analysis-harvest.html [Accessed 12 Jun 2022].
[14] Signell A, Schöring J, Aspnäs M, Westerholm J. Parallelization. In: Spatial decomposition and load balancing of a single tree level forest dynamics simulator, Vol. 2. 2010.
[15] Harja D, Vincent G. Spatially explicit individual-based forest simulator (sexi-FS): for management of agroforests. In: CIFOR-ICRAF. 2013, https://www.cifor-icraf.org/knowledge/publication/__35262/ [Accessed Jun. 12, 2022].
[16] Wikström P, et al. The heureka forestry decision support system: An overview. Math Comput Nat-Resour Sci MCFNS 2011;3(2):87–95, 8.
[17] Fagerberg N, Lohmander P, Eriksson O, Olsson J-O, Poudel BC, Bergh J. Evaluation of individual-tree growth models for picea abies based on a case study of an uneven-sized stand in southern Sweden. Scand J for Res 2022;37(1):45–58. http://dx.doi.org/10.1080/02827581.2022.2037700.
[18] Antón-Fernández C, Astrup R. Sitree: A framework to implement single-tree simulators. SoftwareX 2022;18:100925. http://dx.doi.org/10.1016/j.softx.2021.100925.
[19] Kershaw J, Weiskittel A, Lavigne M, McGarrigle E. An imputation/copula-based stochastic individual tree growth model for mixed species acadian forests: a case study using the Nova Scotia permanent sample plot network. For Ecosyst 2017;4. http://dx.doi.org/10.1186/s40663-017-0102-2.
[20] Van Rossum G, Drake FL. Python reference manual. CWI; 2009.
[21] Hanwell MD, Martin KM, Chaudhary A, Avila LS. The visualization toolkit (VTK): Rewriting the rendering code for modern graphics cards. SoftwareX 2015;1–2:9–12. http://dx.doi.org/10.1016/j.softx.2015.04.001.

[22] González JM, Ulises Diéguez A. Forpylib.simulation — forpylib documentation. 2022, https://forpylib.readthedocs.io/_modules/forpylib/simulation.html [Accessed 13 Jun 2022].

[23] Liski J, Palosuo T, Peltoniemi M, Sievänen R. Carbon and decomposition model Yasso for forest soils. Ecol Model 2005;189(1):168–82. http://dx.doi.org/10.1016/j.ecolmodel.2005.03.005.

[24] Schroeder W, Martin K, Lorensen W. The visualization toolkit. In: An object-oriented approach to 3D graphics. 2006.

[25] Yan R, Guo X, Xu C. Reconstruction and visualization of human gastrointestinal tract. Int J Biomed Sci IJBS 2012;8(1):22–7.

[26] Raymer AK, Gobakken T, Solberg B, Hoen HF, Bergseng E. A forest optimisation model including carbon flows: Application to a forest in Norway. For Ecol Manag 2009;258(5):579–89. http://dx.doi.org/10.1016/j.foreco.2009.04.036.

[27] Bergseng E, Ask JA, Framstad E, Gobakken T, Solberg B, Hoen HF. Biodiversity protection and economics in long term boreal forest management — A detailed case for the valuation of protection measures. For Policy Econ 2012;15:12–21. http://dx.doi.org/10.1016/j.forpol.2011.11.002.

[28] Sjølie HK, Latta GS, Solberg B. Combining backcasting with forest sector projection models to provide paths into the future bio-economy. Scand J for Res 2016;31:1–28. http://dx.doi.org/10.1080/02827581.2016.1186218.

[29] Sjølie HK, Latta GS, Solberg B. Potentials and costs of climate change mitigation in the norwegian forest sector — does choice of policy matter?. 2013, https://cdnsciencepub.com/doi/abs/10.1139/cjfr-2012-0457 [Accessed 05 Jul 2022].