The Generic Carbon Budget Model (GCBM)

The Generic Carbon Budget Model (GCBM) is a valuable tool jointly developed by the Carbon Accounting Team of Natural Resources Canada's Canadian Forest Services and Moja Global. It serves as a comprehensive platform for efficiently setting up and operating advanced monitoring, reporting, and verification systems for carbon-related data.

At its core, GCBM operates on the Flint platform, developed and managed by Moja Global. Flint, an open-source framework, provides robust infrastructure for handling both spatial (e.g., maps) and non-spatial (e.g., tables) data. This accessibility fosters collaboration and innovation, allowing users to focus on building specific scientific modules without getting bogged down in technical details.

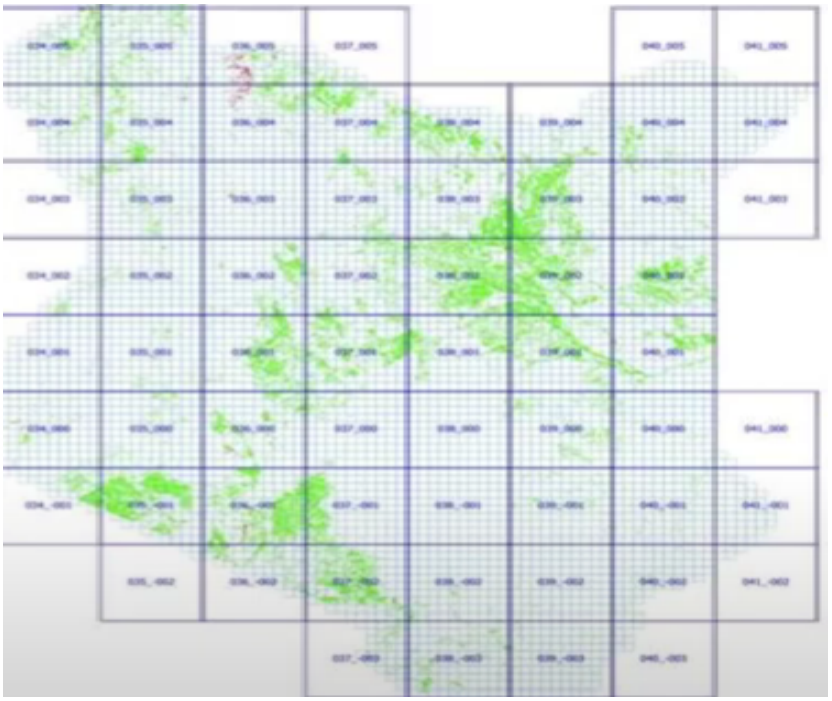
Flint's scalability is a key feature, enabling it to run on regular laptops or scale up to more powerful computing clusters or cloud platforms. This versatility makes it suitable for a wide range of users, from individual researchers to large organizations.Operating on configuration files, Flint follows an event-driven framework, reacting to different events in the simulation lifecycle. This adaptability ensures efficient processing and allows users to tailor simulations to their specific needs.

When modelling carbon dynamics, Flint ensures accuracy by preserving mass balance and accounting for all carbon fluxes. Processes such as growth, mortality, and disturbances are represented as transfers between pools, maintaining fidelity to real-world dynamics.

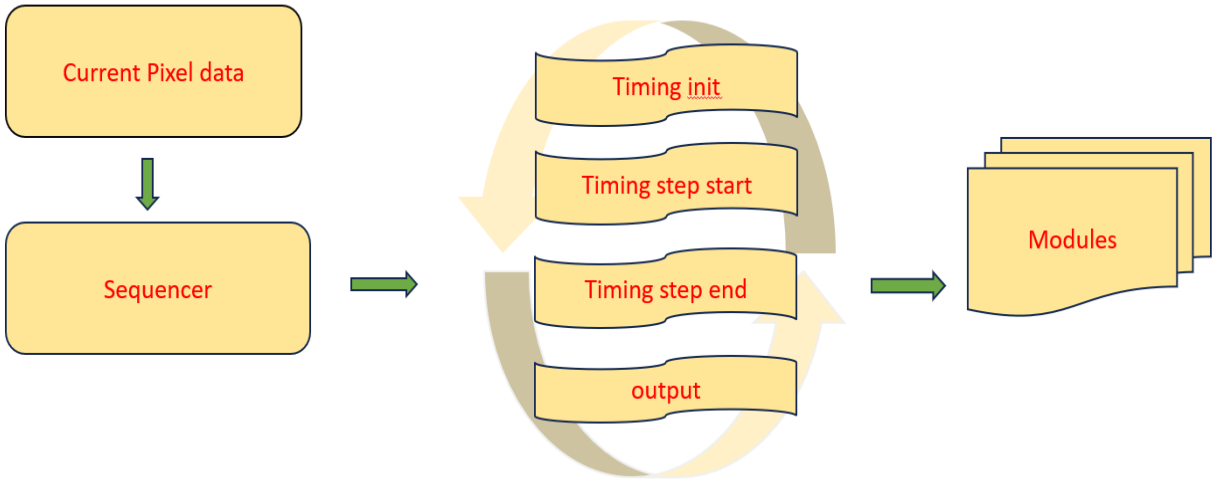
Landscape Division and Pixel Structure

In the landscape model, the entire area is divided into one-degree by one-degree square tiles, which are further subdivided into smaller 0.1-degree blocks. These blocks function as the basic working units for the model's operations. Within each block, the landscape is represented by pixels, with configurable sizes that can be set to values equal to or less than 0.1 degrees. This hierarchical division provides a structured framework for organising and analysing spatial data at various resolutions.

One significant distinction between the Flint and CBMCFS3 models lies in their simulation strategies.



FLINT: Modules Overview

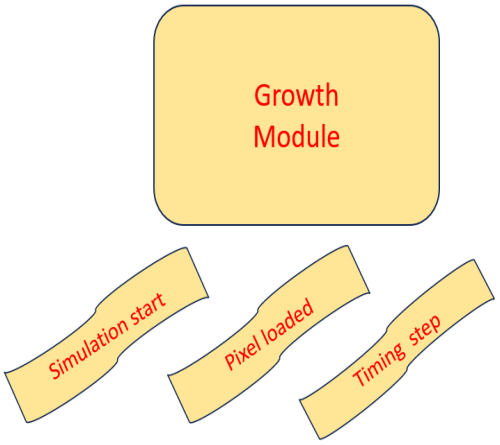


The event-driven system depicted in the flowchart operates by simulating the time series for each pixel comprehensively before moving on to the next. At the core of this system is the sequencer module, which orchestrates the sequence of system lifecycle events. Science modules subscribe to these events, leveraging them to execute their processing tasks. The sequence begins with the loading of data for each pixel, followed by initialization during the Timing Init Event. As the simulation

progresses, the Timing Step Start event triggers processing for the current time step, while the Timing Step End event marks its completion. Subsequently, any post-time step reporting is facilitated through the Output Event. An illustrative example of this system in action is the growth model.

Example: Growth Model in GCBM

One example of a module within the GCBM is the growth model, which subscribes to three system events as mentioned earlier. During simulation start, the growth model loads the biomass equations. Each time a pixel is loaded, the model retrieves the organic matter turnover rates, and during the timing step, it processes the current growth curves.



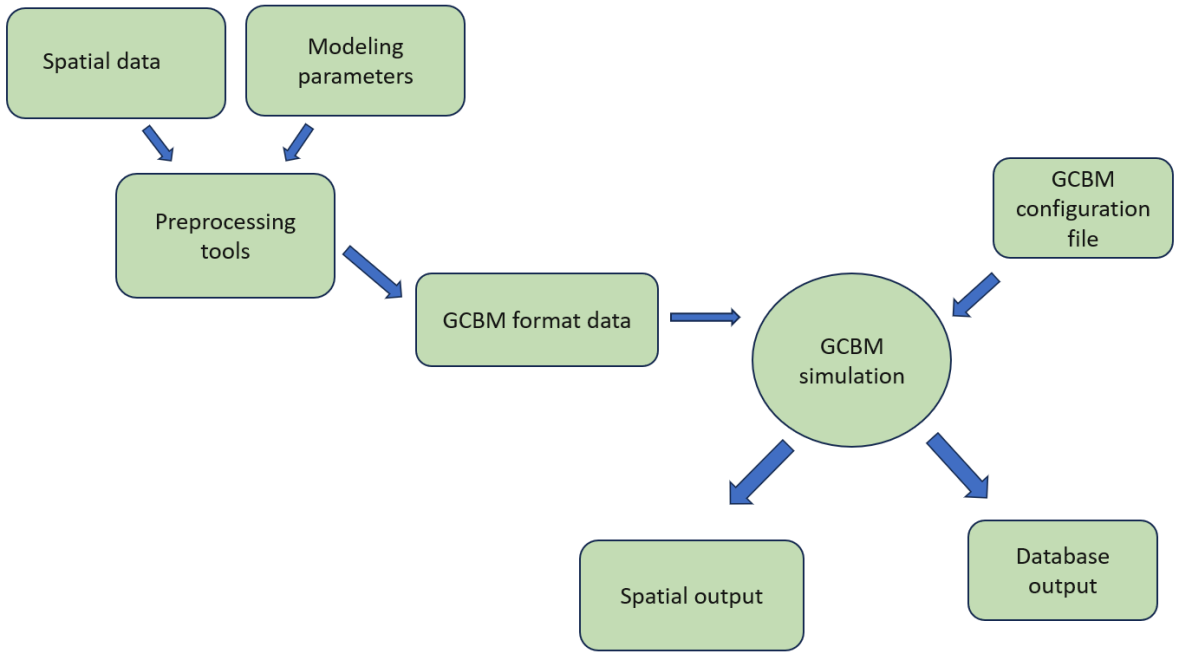
The GCBM's growth model, like its counterpart CBMCFS3, is implemented as a module running on the Flint platform. It represents ecosystem components such as biomass, soil, and greenhouse gases as carbon pools. The growth model incorporates yield curve-based growth modules and accounts for disturbances through transfers between carbon pools. Core modules of the growth model include biomass growth and mortality, dead organic matter and soil dynamics, and disturbances' impact, which can include management activities like harvesting, natural disturbances like fire, and land-use changes like deforestation or afforestation.

In the development of the Flint and GCBM processes, validation against a non-spatial model was conducted. Comparing Canadian national-scale carbon results between the GCBM and CBMCFS3 models showed nearly identical outcomes. However, there are key differences between the GCBM and CBMCFS3 models. In the GCBM, spatial layers are required for inventory and disturbance data, whereas CBMCFS3 utilises database tables. Additionally, the GCBM explicitly identifies the locations of all disturbance events in spatial layers, as opposed to using rule-based methods. The GCBM also provides both spatial and tabular outputs for pools and fluxes, making it easier to analyse and interpret the results. Finally, the GCBM's modular design allows for the addition of new modules and facilitates simulation over very large landscapes with numerous pixels.

Structure of the GCBM Project

The GCBM project follows a structured workflow, starting with the user's spatial data, including forced inventory and disturbances. This data is combined with a database of non-spatial modelling parameters, such as growth curves and disturbances matrices. The combination undergoes preprocessing using Python-based tools to format everything into a GCBM-readable format. Along with GCBM configuration files in JSON format, this prepared data goes into the GCBM simulation.

The simulation generates spatial and SQL database outputs. For spatial data, users require a map of initial forest age or time since the last disturbance, along with a classifier. Additionally, a mean annual temperature layer is recommended, as it affects decay rates. Administrative and ecological boundary layers are also needed.



On the non-spatial or tabular data side, the GCBM relies on a CBMCFS3 archive index database containing a library of non-spatial ecological modelling parameters and a yield table in CSV or Excel format. For users outside Canada, a customised archive index database with country-specific parameters is necessary. This database includes spatially referenced parameters like decay rates, disturbance matrices, and root biomass coefficients, used by both CBMCFS3 and GCBM.

Disturbance events for the GCBM can be in any spatial layer format, with raster files representing disturbances per year or per disturbance type and year. Vector layer files can also be used, containing polygons of completely disturbed areas, with attributes such as year and type of disturbance. These files are paired with a lookup table mapping non-standard disturbance type names to actual names in the GCBM archive index database.

Standalone Project Structure

The standalone project offers users a series of individual pre and post-processing tools organised into a workflow, aiming to familiarise new users with inputs, processes, and options for running the GCBM. This standalone project follows a "what you see is what you get" approach, importing spatial layers and CBMCFS3 standard imports to precisely simulate what is specified by the input data.

Designed for portability, users can copy the standalone project anywhere and run it with the correct Python and GDAL environment. It is best suited for projects where data is already available in the correct format and does not require preprocessing beforehand. While not designed for real-world projects, the standalone project template provides the same basic project structure and scripts, making it an efficient way to kickstart new projects quickly.

Although the Carbon Accounting Team at the Canadian Forest Service has developed some advanced scripting, the standalone project template serves as the basic project structure used for various applications.

Standalone Project Workflow Overview

The standalone project workflow begins with the Tiler, which converts raw user data into a format readable by GCBM. Once the input database is created, the Recliner 2GCBM tool generates the SQL input database. Upon successful completion of the simulation, the process proceeds to two post-processing tools.

Flint Tool: This tool completes the GCBM spatial output, stitching together all output pieces into GeoTiff files.

Completed GCBM Results: This tool generates ecosystem indicators in an SQLite database.

Standalone Project Output

The output of the standalone project includes:

Tabular Output:

- Stored in SQLite or PostgreSQL databases by default.
- Flattened reporting tables for user-friendly access.

Spatial Output:

- Initially stored as an environmental GeoTIFF.
- After post-processing, it was converted to standard GeoTIFF format.

Directory Structure of the Project Template

The project template consists of the following directories:

Documentation:

- Contains important files such as:
    - Installation guide
    - Information about the input database schema
    - Ecosystem indicators in the output database
    - Example Python snippets for configuring scripts

Tiler Script:

- Simulation working directory
- Contains scripts for data preprocessing

Input Database:

- Spatial input data, e.g., the archive index database and the yield table

Prerequisites for building a GCBM project using the "standalone" project template

1. Configure and execute the tiler to convert and process input spatial data into GCBM format.
2. Utilise the recliner2gcbm tool to generate the GCBM input database.
3. Run the GCBM configuration script.
4. Execute GCBM.
5. Use post-processing tools in the final step to generate the ultimate output.

Similar to any CBM-CFS3 project, users can generate standard ecosystem indicators such as net primary productivity and database tables. Additionally, they can convert raw spatial output into final layers.

Project Background

Python is a General-Purpose Scripting Language widely used for most pre and post-processing tasks in the model. Basic knowledge of Python is essential for running GCBM. The configuration files for GCBM are in JSON format, which stands for JavaScript Object Notation. It's a commonly used file format in many software packages, including GCBM and its supporting tools.

There are a few prerequisites to run the standalone project in GCBM. Instructions can be found in the directory:

- Check Tool Installation by Running `run_all.bat`.

Troubleshooting Steps:

> If the tiler script fails, it's likely a Python or GDAL issue. Check environment variables.If recliner2gcbm fails, it's usually due to an Access database driver issue. Switch from recliner2gcbm x64 to recliner2gcbm x86 by changing the platform variables near the top of the `run_all.bat`.If GCBM fails, it's usually because the tiler or recliner2gcbm failed earlier.If compile2gcbm fails, it's either because GCBM failed or due to a Python or Access database driver issue.

Configuring the Tiler Python Script

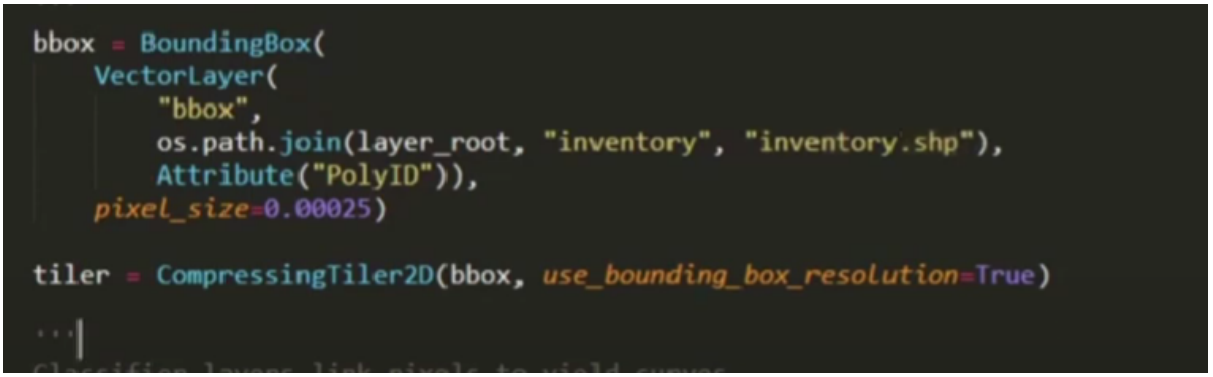
The tiler script is versatile, capable of utilising layers in vector or raster format, and supports any type supported by the GDAL library (Geospatial Data Abstraction Library). This library is essential for working with geospatial data.

The tiler script leverages the "moja" Python package, which is a library designed for converting spatial layers to the FLINT format. It handles tasks such as resampling and reprojection.

To facilitate usage, a project template named tiler.py has been prepared. Users can locate it in the specified directory. This template streamlines the process of configuring the tiler script for their specific needs.

Configuring the tiler.py Script

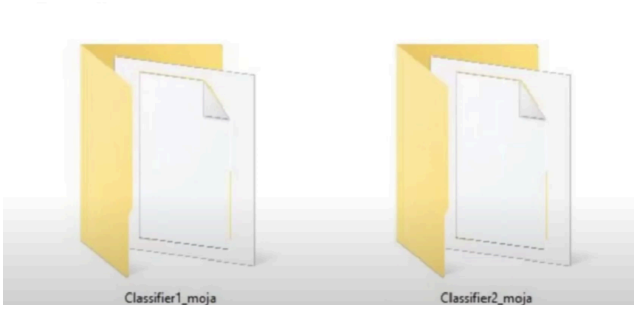
The initial step in the tiler.py file is to set up the boundary box, defining the study area for the plant simulation. This boundary box serves as the reference for cropping, reprojecting, and resampling all other spatial layers. Importantly, it points to the inventory.shp file, although it can be adjusted to point to any spatial file.



Following this, users must designate an attribute within the chosen file to define the simulation area. For example, the poly ID attribute may be selected, representing each polygon in the landscape.Next, users determine the pixel size, representing the resolution at which the simulation will operate. Additionally, data layers such as initial stage and mean annual temperature may be identified, with the latter being optional.

Further down, disturbance layers indicating disturbance type and year must be included, with specific formatting requirements.The tiler script also involves setting up classifier layers, which link inventory spatial data to the yield table. These classifiers are defined in the script and must match those in the recliner to the GCBM

script.Once configured, the tiler script can be run independently using process_spatial_buyer.bat. Alternatively, it can be run alongside other preprocessing and simulation tools using run_all.bat.

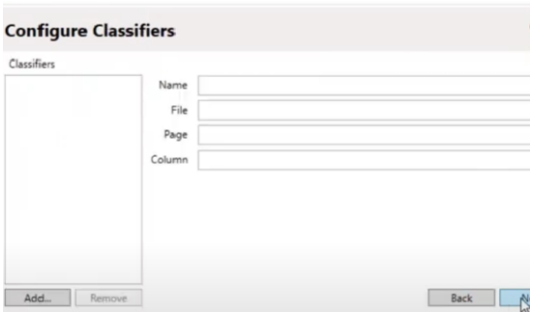


Upon execution, the final GCBM format layer will appear in the tile subfolder within the layers folder. This includes cropped, reprojected TIFF files, a CSV file containing transition rule data, and a JSON file providing metadata about the tiled layer. Additionally, users will find a log file for the tiler script and a FLINT format layer in the tiled subfolder of the layers directory. These components, with successful configuration, facilitate subsequent steps in the process.

Generating the Input Database for GCBM

To generate the input database for GCBM, users will utilize the "Recliner to GCBM" tool. This tool, written in C#, acts as a bridge between the spatial CBM-CFS3 and the spatially explicit GCBM. It employs the same default archive index database with Canadian non-spatial parameters as CBM-CFS3.The Recliner to GCBM tool can import a yield table in CSV or Microsoft Excel format. After some configuration, it generates the SQLite-formatted GCBM input database. The output can be found in the input_database folder, along with a recliner to GCBM .log file.

The GCBM input database relies on a yield table containing data such as species, classifier values, and volume increments. Before configuring the input database, users must note the classifier layer names set up in the tiler script.



These names must remain consistent throughout.To begin using the tool, users should navigate to the input_database folder and double-click on the run_recliner to gcbm_gui.bat file to launch it. Then, in the configuration project window, they must first browse to the archive index database file called archive_beta_install.mdb and specify the output database.

In the subsequent windows, users can add classifier layers by clicking the "Add" button and configuring each component, ensuring consistency with the tiler script. Once classifier layers are set, users can proceed to set up growth curves linked to the spatial inventory using the identified classifiers.This process involves identifying the file containing yield curve data, mapping classifier values, and specifying growth curve intervals. Additionally, users must identify the column representing tree species or forest type in the archive index database.

After completing these steps for both project classifiers, users can proceed to the next windows to configure growth curves. This entails mapping classifier columns, specifying growth curve intervals, and identifying columns representing species and volume increments.

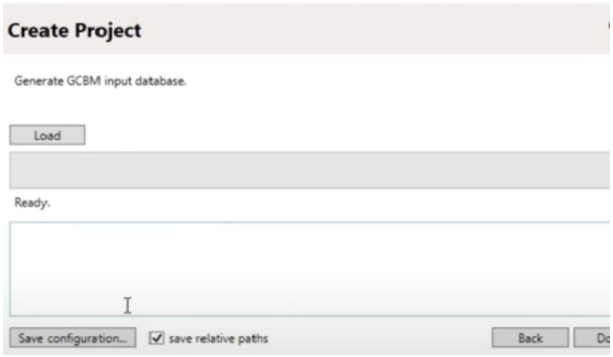
Once all settings are configured, users can proceed to generate the input database by clicking the "Next" button.

****Setting Up Transition Rules****

Transition rules describe the outcomes when a specific stand type is disturbed by a particular disturbance type. These rules, created by the tiler script, specify the actions taken after a disturbance event. In this example dataset, only one transition rule is included, which resets the stand age to zero after any disturbance event. Configuring transition rules is optional, and the included rule in this project serves a basic function of resetting stand age. Users need to inform the tool about their transition rules file, which can be done in the "Configure Transition Rules" window.

In this window, users browse to and open their transition rules file. Once opened, users ensure that the header row checkbox is checked. Mapping the classifier and identifying the column representing ID values are also essential steps in this process.After completing the basic mapping, users can proceed to the next window, where they have the option to make changes to configured disturbance categories. However, this step is more advanced and optional.Once all settings are configured, users proceed to save the configuration, ensuring they have the option to regenerate the input database if needed. This allows for flexibility in making changes to the project without redoing all previous steps.

Executing the run_all.bat file regenerates the input database based on the saved configuration, facilitating modifications such as adding extra yield curves without redoing the entire process.



If everything is set up correctly, users will see the progress bar display without errors. Completing the process entails clicking on the "Done" button, indicating successful setup of transition rules and completion of the configuration process.

Configuring and Running a GCBM Project

The GCBM utilizes several configuration files to execute a simulation. These files are stored in the gcbm_project folder, with templates available in gcbm_project/template. There are two main types of configuration files: provider configuration files and project configuration files.

Provider configuration files define data sources for modules, using file paths that can be absolute or relative to the directory from which GCBM was executed. Project configuration files define simulation details such as module order, carbon pool names, and data variables. While some config files contain settings essential for model functioning, others are typically left unchanged by users.

Commonly modified project configuration files include local_domain.json, modules_output.json, and variables.json. The local_domain.json file specifies simulation start and end years and the number of threads used by the model, adjusted to match the number of computer cores.

The modules_output.json file controls output modules, allowing users to enable/disable spatial output and define new indicators. Variables.json contains data variables and specifies how to retrieve their values.

Though most simulations can be run without modifying these files, scripts in the template project automatically set up config files from template files. Users unfamiliar with editing JSON files can validate modified files using available tools.

Parameters commonly edited include thread count, simulation start/end dates in local_domain.json, and spatial outputs in modules_output.json.

A batch file named update_gcbm_configuration.bat updates the configuration after tiler and recliner to GCBM steps, ensuring everything is up to date. This step is also included in run_all.bat.

To run a GCBM simulation, users can execute run_gcbm.bat in the gcbm_project folder. Similar to configuration, running the model is also integrated into the run_all.bat file.

Configuring and Running a GCBM Project

The GCBM utilises several configuration files to execute a simulation. These files are stored in the gcbm_project folder, with templates available in gcbm_project/template. There are two main types of configuration files: provider configuration files and project configuration files.Provider configuration files define data sources for modules, using file paths that can be absolute or relative to the directory from which GCBM was executed. Project configuration files define simulation details such as module order, carbon pool names, and data variables. While some config files contain settings essential for model functioning, others are typically left unchanged by users.

Commonly modified project configuration files include local_domain.json, modules_output.json, and variables.json. The local_domain.json file specifies simulation start and end years and the number of threads used by the model, adjusted to match the number of computer cores.The modules_output.json file controls output modules, allowing users to enable/disable spatial output and define new indicators. Variables.json contains data variables and specifies how to retrieve their values.

Though most simulations can be run without modifying these files, scripts in the template project automatically set up config files from template files. Users unfamiliar with editing JSON files can validate modified files using available tools.

Parameters commonly edited include thread count, simulation start/end dates in local_domain.json, and spatial outputs in modules_output.json.A batch file named update_gcbm_configuration.bat updates the configuration after tiler and recliner to GCBM steps, ensuring everything is up to date. This step is also included in run_all.bat.To run a GCBM simulation, users can execute run_gcbm.bat in the gcbm_project folder. Similar to configuration, running the model is also integrated into the run_all.bat file.

Running the Post-Processing Tools

After running the model, the final steps involve executing the post-processing tools. The first of these is the "Compiled GCBM Results" tool, which transforms raw database output into user-friendly ecosystem indicators. Users can customise indicators by editing the compilerresults.json file.

This tool can be run independently by double-clicking the compile_gcbm_results.bat file. Alternatively, it is integrated into the run_all.bat file. Executing either will generate a final output database in SQLite format stored in the processed-output folder.

Most results tables include simulation year, classifier set, age class, and area in hectares. Various ecosystem indicators are available for querying carbon results. The tables labelled with the v_prefix contain fully processed results, while others represent intermediate steps and can be ignored by users not familiar with SQL.

Below are basic queries to display available indicators in the reporting table:

SELECT DISTINCT 'v_flux_indicator_aggregates' AS table_name, indicator FROM v_flux_indicator_aggregates UNION ALL

SELECT DISTINCT 'v_flux_indicators' AS table_name, indicator FROM v_flux_indicators UNION ALL

SELECT DISTINCT 'v_pool_indicators' AS table_name, indicator FROM pool_indicators UNION ALL

SELECT DISTINCT 'v_stock_change_indicators' AS table_name, indicator FROM v_stock_change_indicators ORDER BY table_name, indicator

These queries list available indicators categorised by table name, allowing users to select the desired indicators for analysis.

In the output tables, the units are typically in hectares and tons of carbon unless otherwise specified by a column name. These reporting indicators depict the state of a location at the end of each time step, considering disturbances, growth, mortality, and decay.

It's important to understand how aggregation works in the reporting table. Aggregation typically occurs for a specific age group and classifier set in a particular year of the simulation. The displayed areas represent the entire classifier set in age

combinations, except for v_disturbance_indicator and v_disturbance_areas, where the area column refers only to the exact disturbed area. In other tables, the area numbers include more than just the disturbed area.

The final post-processing spatial output combines all the blocks of work into GeoTIFF files covering the entire simulation landscape. This tool can be run independently by double-clicking on the create_tiff.bat file or as part of the run_all.bat file script.

Troubleshooting Tips and Steps for Advanced Installation

When installing the model, it's common to encounter issues that require troubleshooting. Here are some tips and steps for troubleshooting an advanced installation of the model:

Check Logs: Most tools and simulation logs can be found in the log subfolder of the one_standalone_template folder. Start by opening and checking the moja_debug.log file to ensure there are no errors. If errors are detected, proceed to check other log files, starting with the tools that were run first, such as tiler_log.txt followed by update_gcbm_config.log, etc.

Error Information: Additional error information may be found in the error dimension table in the gcbm_output database for GCBM run errors. If no obvious errors are found in the data preparation tool logs, try disabling the spin-up procedure or modules one at a time until the problematic module is isolated.

Configuring Threads: Users can configure the number of threads to use in the local_domain section of the gcbm_config.json file. It's recommended to match the number of cores in their machine, up to a maximum of 10.

Start Small: Begin with a lower resolution when setting up the tiler script. Set the bounding box resolution to something coarse, like 0.01, for the initial run. Once everything is working, the resolution can be changed to something finer, like 0.00025, for larger projects.
Scaling Up: For larger projects, users can scale up to cloud computing or multiple machines with some extra configuration and scripting. This allows for handling more complex simulations efficiently.

System Environment Variables: Users may choose to move configuration settings out of various GCBM batch files for better management. This can be achieved by setting up system environment variables.

These troubleshooting tips and steps should help users diagnose and resolve issues encountered during the installation and configuration of the GCBM model.