User's Guide to the Code Generation Tool CGT

Hagen Radtke

September 1, 2014

Contents

1 Introduction

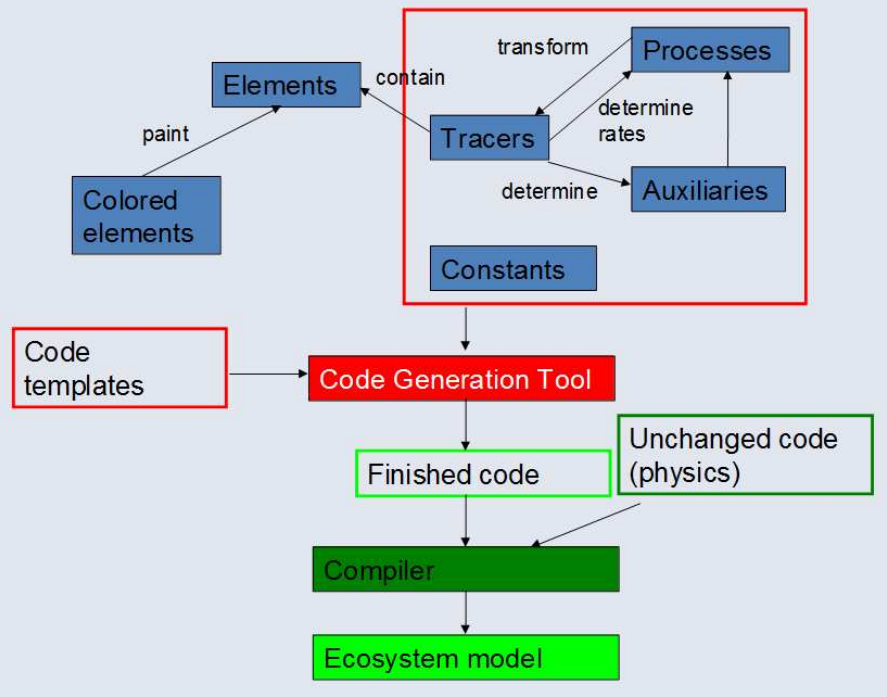

Figure 1.1: Conceptual view of what the code generation tool does.

1.1 What is the Code Generation Tool

The ecosystem Code Generation Tool (CGT) is a tool to create ecosystem model code from two ingredients:

1. a formal description of the ecosystem tracers and processes in a list of text files

2. a set of "code templates" for the host model

The tool then extracts the information from the text files and fills the code templates to create your model code.

1.2 Where can I find it

All you need to generate your ecosystem model can be downloaded from `http://www.ergom.net`. You need:

- A binary: Windows binaries and Linux binaries exist. Linux binaries should run under most recent LINUX distributions.
- The formal model description (text files).
- The code templates for different models. At the beginning, try to start with the MATLAB model, as it is most easy.

1.3 What is the structure of this document

- In Chapter 2, a short overview is given on the concept of process-oriented ecosystem modelling. It outlines the concept of tracers, processes and element conservation.
- In Chapter 3, you find the information on how to describe your ecosystem model in a way such that it can be read by the Code Generation Tool.
- In Chapter 4, you find the information on how create your ecosystem model using CGT, and how to write your own "code templates" to include your ecosystem model in a physical host model of your choice.
- In Chapter 5, you find a description on the numerical time-stepping scheme used that keeps the tracer values non-negative.
- In Chapter 6, you find documentation on how the nutrient tagging and age attribution technique work, both in theory and in practice.
- In Chapter 7, you will find documentation for using advanced features of CGT: Balancing equations, combined tracers, vector tracers and pseudo-3d tracers.

2 The concept of process-oriented modelling

The definition of what a tracer and what a process is shall be outlined here.

3 Modifying the ecosystem model

To modify the ecosystem model (normally) only the textfiles need to be changed. (see "Syntax of the text files" in section 3.2) So, first, create a copy of your text file directory which you want to modify.

3.1 Using the Editor `cgt_edit`

It is recommended to use the special editor `cgt_edit` to modify your ecosystem model. However, you may use any editor to edit the text files, it is straightforward. So here we only describe the usage of `cgt_edit`.

- Run `cgt_edit.exe` (or `cgt_edit`). You will be asked whether to load an existing model or create a new one. So you have to load or save the file `modelinfos.txt`. This will open (or create) the whole set of text files for editing.
- Note that all changes are saved immediately, so be sure you made a working copy of your existing set of text files.
- On the left, you can see different categories (e.g. constants, tracers, processes) marked in different colors. Click on them to view the list of constants, list of tracers etc.
- When you select e.g. a process, you will see its properties on the right. Some words appear with a colored background, these are defined elsewhere (e.g. constants in blue, tracers in green). Right-click them to see their properties.
- Afterwards, use the "Back"-Button on the top left to return to where you were.
- A click on the "Help"-Button at the top right will tell you what properties of the selected tracer, process etc. you may modify. A detailed explanation of what these properties mean is given in the next section.

3.2 Syntax of the Text Files

There are different textfiles that each contain different parts of the ecosystem model. These are:

- `constants.txt` - contains model constants, such as stochiometric ratios, maximum growth rates etc.

- `tracers.txt` - contains the description of the tracers (state variables) of the model.
- `auxiliaries.txt` - contains the auxiliary variables that need to be calculated to determine the process rates
- `processes.txt` - contains the model processes that transform the tracers into each other.
- `elements.txt` - contains a list of chemical elements that are contained in the tracers.
- `celements.txt` - contains a list of "colors" attributed to the elements to track them in the model by a source-attribution technique, seechapter 6.
- `modelinfos.txt` - contains some settings and some general information about the ecosystem model.

In the textfiles, you describe properties if these tracers, processes etc. All of them contain a property `comment=`. Here, please put in literature citations, comments on uncertainty, anything that may help others understand what's going on in your model.

3.2.1 `constants.txt`

`name =`	code name of constant
`description =`	description including unit, default=""
`value =`	value(s) separated by ";" *By default, specify only one value.* *How to use a "vector constant" is described in section 7.3.*

3.2.2 `tracers.txt`

`name =`	variable name in the code
`description =`	description including unit, default="", e.g. "flaggelates"
`atmosDep =`	0=no atmospheric depositon of this tracer (default), 1=atmospheric deposition of this tracer *If you set atmosDep to 1, you will need to provide a forcing file for atmospheric deposition.*
`childOf =`	e.g. "flaggelates" for "red_N_in_flaggelates", default="none" *You will most probably not use this property manually. If you specify colored elements (for source attribution), the new tracers that are created as copies of the old ones will have this property.*

`contents =`	number n of elements contained in this tracer, default=0
	It is most elucidating to see an example:
	`contents = 2`
	`N = 1`
	`P = rfr_p`
	(where ***`rfr_p`*** *is a constant (redfield ratio P/N))*

`dimension =`	how many instances of this tracer exist, e.g. a tracer named "cod" with dimension=2 exists as "cod_$cod" which is "cod_1" and "cod_2", default=0 *This is described in detail in "Using vector tracers" (section 7.3).*
`gasName =`	name of an auxiliary variable containing the concentration [mol/kg] of the dissolved gas in the surface cell, e.g. "co2" for tracer "dic". Default="" meaning it is the tracer concentration itself. *You do not need this parameter if you define a gas flux manually.*
`initValue =`	initial value, default="0.0", set `useInitValue` to 1 to use it *By default, you provide a file with an initial concentration. Alternatively, you can say it is constant everywhere. In this case, set* ***`initValue`*** *to the correct value and* ***`useInitValue`*** *to 1. If* ***`useInitValue=0`****,* ***`initValue`*** *is ignored.*
`isActive =`	1=active (default); 0=virtual tracer to check element conservation *For the generated code, only active tracers are relevant. But to check if the stochiometry of process equations is correct, you sometimes need to consider "tracers" that you would normally neglect, such as water (H2O). For these, set* ***`isActive=0`*** *and they will not appear in your model, but allow you to check its consistency or balance process equations automatically, see section 7.1.*
`isCombined =`	1=combined tracer that accumulates several virtual tracers (isActive=false) in one variable, its contents are tracers rather than elements; 0=not (default) *This is a strange kind of tracer that needs some detailed explanation given in section 7.2.*
`isMixed =`	1=mix with neighbour cells if negative, 0=do not, default=0, only applicable to tracers with vertLoc=WAT

	Legacy code for MOM3 compatibility.
`isOutput =`	1=occurs as output in model results (default); 0=only internal use *In `cgt_edit.exe`, simply click the checkbox next to the tracer to select it for output.*
`isPositive =`	0=may be negative, 1=always non-negative (default) *If `isPositive` is set to 1, this has two effects: a) when the tracer value is negative at the beginning of the time step (e.g. due to advection overshoots), it is treated as zero. b) when a positive-definite time-stepping scheme is used, processes that consume this tracer will be stopped when the tracer value becomes zero.*
`isStiff =`	0=not stiff (default); 1=stiff, use Patankar method if concentration declines; 2=stiff, always use modified Patankar method *A "stiff tracer" is a tracer which is consumed by processes on a very short time scale, that is, on a time scale below the model time step. In this case, a Euler-Forward calculation would typically give negative tracer values, and a simple clipping of the processes would lead to bad estimates for the tracer concentration. However, a special time stepping method can be used to overcome this problem as described in section 5.2.*
`massLimits =`	semicolon-seperated string of (dimension-1) FORTRAN expressions for mass limits for stage-resolving models [mol], default="" *Relevant for vector tracers only. This is described in detail here: Using vector tracers (section 7.3)*
`opacity =`	fortran formula for opacity (for light limitation), default="0" *The opacity of a tracer (given in [m^2/mol]) describes how much it inhibits the light coming from the top. Use a real number or the name of a constant. Is only relevant for tracers with `vertLoc=WAT` or `SUR`.*
`molDiff =`	molecular diffusivity in pore water [m2/s], use the name of a vertLoc=SED auxiliary variable, default="0.0" *Only for use in sediment models*
`opacity =`	fortran formula for opacity [m2/mol] (for light limitation), default="0"

	You typically use the name of a constant here, giving the light attenuation [1/m] at a concentration of 1 mol/m3.
`riverDep =`	0=no river depositon of this tracer, 1=river deposition of this tracer (default) *If you set "riverDep" to 1, you will need to provide concentrations for this tracer in the river deposition file.*
`schmidtNumber =`	name of an auxiliary variable describing the Schmidt number [1] for gasses which flow through the surface, default="0" *You do not need this parameter if you define a gas flux manually.*
`solubility =`	name of an auxiliary variable describing the solubility [mol/kg/Pa] for gasses which flow through the surface, default="0" *You do not need this parameter if you define a gas flux manually.*
`useInitValue =`	1=use initValue as initial concentration, 0=do not (load initial concentration from file) (default) *see initValue*
`vertDiff =`	formula for vertical diffusivity [m2/s], default="0"
`verticalDistribution =`	Relevant for `vertLoc=FIS` only: Name of an auxiliary variable proportional to which the vertical distribution of the tracer is assumed. Default="1.0" *For `vertLoc=FIS` (Pseudo-3d) tracers, see a detailed explanation in section 7.4.*
`vertLoc =`	Where the tracer exists: `WAT`=everywhere in the water column (default), `SED`=in the sediment only, `SUR`=in the surface cell only, `FIS`=Pseudo-3d tracer (fish-type behaviour) *Tracers with `vertLoc=WAT` have the unit [mol/kg]. Tracers with `vertLoc=SED` or `SUR` have the unit [mol/m^2]. These units are, however, automatically converted if both `vertLoc=WAT` and `vertLoc=SED` tracers appear in one process. `vertLoc=SUR` is not supported yet (for tracers) in MOM4, as we would need surface drift. `vertLoc=FIS` (Pseudo-3d tracers) are stored in one vertical layer only (just like `vertLoc=SED` tracers) but can interact with tracers everywhers in the water column, see a detailed explanation in section 7.4.*
`vertSpeed =`	formula for vertical speed [m/day], default="0"

	Positive means upward. The vertical speed and diffusivity need not be constant, but may e.g. also be an auxiliary variable.

3.2.3 auxiliaries.txt

`name =`	variable name in the code
`description =`	e.g. "absolute temperature [K]" default="" *Please give all your auxiliary variables a unit. The dimensionless ones should end with [1].*

`temp1= ... temp9=`	for calculating a temporary value which appears in the formula, default="".
	e.g.
	`temp1=no3*no3`
	`temp2=no3limit*no3limit`
	`formula=temp1/(temp1+temp2)`

`formula =`	formula for calculating this auxiliary variable *See section 3.3 on how to write formulas that work in several programming languages.*
`calcAfterProcesses =`	1=calculate this auxiliary after all process rates are known, default=0 *This property is useful if you want to have a flux appearing in your output file which is the sum of several processes, but you do not want all of the processes in your output.*
`iterations =`	how often this auxiliary variable is calculated in an iterative loop, default=0 meaning no iterations *Iterations are always done before all other auxiliary variables (`iterations=0`) are calculated. If you want to calculate an auxiliary variable before the iterative variables are calculated, set `iterations` to 1.*
`iterInit =`	the initial value in the iterative loop, default="0.0"
`isOutput =`	1=occurs as output in model results; 0=internal use only (default) *In `cgt_edit.exe`, simply click the checkbox next to the auxiliary variable to select it for output.*
`isUsedElsewhere =`	1=make the value of this auxiliary accessible from outside the biological model (e.g. use a "diagnostic tracer" in MOM5); 0=internal use only (default)
`isZGradient =`	1=is a vertical gradient of a tracer, 0=is not (default). *If 1, `formula` must be the name of the tracer, which must have `vertLoc=WAT`. `isZGradient=1` requires `vertLoc=WAT`.*

`isZIntegral =`	1=is a vertical integral (of value times density) of a tracer or an auxiliary variable, 0=is not (default). *If 1 "formula" must be the name of the tracer or auxiliary variable, which must have vertLoc=WAT.* ***`isZIntegral=1`*** *requires* ***`vertLoc=SED.`***
`vertLoc =`	Where the auxiliary variable is calculated: `WAT`=everywhere in the water column (default), `SED`=in the sediment / at sediment-water interface, `SUR`=in the surface cell only

3.2.4 `processes.txt`

`name =`	variable name in the code used for the turnover
`description =`	e.g. "grazing of zooplankton"
`equation =`	equation which, like a chemical equation, lists reaction agents and products of this process. *example:* ***`t_no3 + 1/16*t_po4 -> t_lpp`*** *tracers to the left of the "->" are consumed, tracers to the right of the "->" are produced by this process.*
`feedingEfficiency =`	name of an auxiliary variable (values 0..1) which tells how much of the food in a certain depth is accessible for the predator with `vertLoc=FIS`. Relevant for `vertLoc=FIS` only. Default="1.0" *For creating processes which involve* ***`vertLoc=FIS`*** *tracers, see a detailed explanation in section 7.4.*
`isActive`	1=active (default); 0=process is switched off
`isOutput =`	1=occurs as output in model results; 0=internal use only (default) *In* ***`cgt_edit.exe`****, simply click the checkbox next to the process to select it for output.*
`limitation =` `limitation =`	`TYPE tracer > value` or `TYPE tracer < value` ***`TYPE = HARD`*** *(theta function),* ***`MM`*** *(Michaelis-Menten),* ***`MMQ`*** *(quadratic Michaelis-Menten),* ***`IV`*** *(Ivlev),* ***`IVQ`*** *(quadratic Ivlev),* ***`LIN`*** *(linear),* ***`TANH`*** *(tangens hyperbolicus)* ***`tracer`*** *= name of tracer that needs to be present* ***`value`*** *= value that needs to be exceeded, may also be a constant or auxiliary* *Several of these lines may exist, limitations are multiplied.*

	See section 5 (positive-definite time stepping scheme) to see how limitations can work, apart from introducing a factor into the formula.
`processType =`	type of process, e.g. "propagation", default="standard" *This property can be used to switch off a whole set of processes at once by setting the* ***`disabledProcessTypes`*** *property in* ***`modelinfos.txt`*** *This is especially useful when some processes (e.g. atmospheric gas flux) are done by the host model sometimes and sometimes not.*

`repaint =`	number n of repainting actions to be done by the process, default=0
	This line is followed by n lines of this kind:
	`<oldColor> <element> = <newColor>`, *e.g.*
	`all N = blue`
	`blue P = none`
	`red all = green`
	Use this property to start element tagging, see chapter 6

`turnover =`	formula for calculating the process turnover [mol/kg or mol/m^2] *[mol/kg] applies for processes with* ***`vertLoc=WAT`****, [mol/m2] for all other processes* *See section 3.3 on how to write formulas that work in several programming languages.*
`vertLoc =`	Where the process takes place: `WAT`=everywhere in the water column (default), `SED`=in the sediment / at sediment-water interface, `SUR`=in the surface cell only, `FIS`=process involving pseudo-3d tracers *Processes with* ***`vertLoc=WAT`*** *have a turnover with unit [mol/kg/day]. Tracers with* ***`vertLoc=SED`*** *or* ***`SUR`*** *have the a turnover with unit [mol/m^2]. For creating processes which involve* ***`vertLoc=FIS`*** *tracers, see a detailed explanation in section 7.4.*

3.2.5 `elements.txt`

`name =`	internal name used, e.g. "N"
`description =`	e.g. "nitrogen"

3.2.6 `celements.txt`

`element =`	internal name of element, e.g., "N"
`color =`	e.g. "red", may not contain spaces
`description =`	e.g. "nitrogen from Oder river", default=""
`atmosDep =`	1=atmospheric deposition of marked tracers may occur, 0=not (default)
`isAging =`	1=accumulates time since entering the system, 0=does not (default) *This implies the creation of two new state variables, e.g. "aged_red_N" and "aged_red_N_at_bottom", storing the product of the total concentration of red N in all tracers and its average age. Also, processes which raise or lower this "age concentration" will be created. The average age can then be accessed via aged_red_N/total_red_N. For details, see Section 6.3.*
`isTracer =`	1=store total Element content in a separate tracer, 0=do not (default) *This implies the creation of two new state variables, e.g. "total_red_N" and "total_red_N_at_bottom", storing the total concentration of red N in all tracers.*
`riverDep =`	1=river deposition of marked tracers may occur, 0=not (default)

3.2.7 `modelinfos.txt`

`name =`	bio-model short name or abbreviation
`description =`	bio-model long name
`version =`	bio-model version
`author =`	bio-model author(s)
`contact =`	e.g. e-mail adress of bio-model author
`ageEpsilon =`	small value used for preventing zero division for age calculation; default="1.0e-20"
`autoBurialFluxes =`	1=auto-generate fluxes for burial of colored elements with isTracer=1; 0=do not (default)
`autoLimitProcesses =`	1=add limitations to all processes that stop them when one of their precursors with isPositive=1 becomes zero (default); 0=do not
`autoMassClassProp =`	0=manual mass-class propagation processes (default); 1=mass-class propagation when upper mass limit is reached; 2=advanced propagation; 3=age-class propagation at beginning of each year
`autoSortMoving =`	1=sort tracers: not vertically moving first, then vertically moving; 0=do not (default)

`autoSplitColors =`	1=split tracers and processes according to colored elements (default); 0=do not
`autoUnixOutput =`	1=enforce Unix line-feed output on Windows systems; 0=do not (default)
`autoWrapF =`	1=auto-wrap too long lines in all files with ".f" or ".F" extension (default); 0=do not
`autoWrapF90 =`	1=auto-wrap too long lines in all files with ".f90" or ".F90" extension; 0=do not (default)
`debugMode =`	1=debug mode (output of all values); 0=output only of those values with output=1 (default)
`inactiveProcessTypes =`	semicolon-separated list of process types that are set inactive, e.g. because they are represented in the host model, e.g. "gas_exchange; sedimentation; resuspension"
`outputPath =`	path where to write the output files
`realSuffixF90 =`	e.g. "8", appends a suffix (e.g. _8) to all real values in .f90 files which do not yet contain a suffix (do not include the underscore here); default=' meaning no suffix is added.
`templatePath =`	path to the code template files

3.3 Writing formulas with compatibility to different programming languages

3.3.1 How to write expressions that work everywhere

CGT is designed to generate different models, possibly in different languages, from the same formal description. However, to make that possible, you have to be careful and use only common syntax in the formulas you write. So, please

- do not use the exponentiation operator `x**y` but the power function `power(x,y)`.
- use the step function `theta(...)` instead of using "greater than"-operators.
- never use more than two arguments in the `min(...)` or `max(...)` function - use e.g. `min(a,min(b,c))` instead of `min(a,b,c)`.
- Only use the natural logarithm (base e) as `log(...)`.

3.3.2 External forcing parameters

The following (physical) parameters may be used in formulas and have to be provided by the host model:

`cgt_temp`	potential temperature	[°C]
`cgt_sali`	salinity	[g/kg]
`cgt_light`	downward flux of photosynthetically active radiation	[W/m^2]
`cgt_cellheight`	cell height	[m]
`cgt_bottomdepth`	depth of the bottom of the current cell	[m]
`cgt_density`	(Boussinesq) density of the water	[kg/m^3]
`cgt_timestep`	biomodel timestep	[days]
`cgt_longitude`	geographic longitude	[deg]
`cgt_latitude`	geographic latitude	[deg]
`cgt_current_wave_stress`	combined shear stress of currents and waves at the bottom	[N/m^2]
`cgt_year`	calendar year (integer)	[years]
`cgt_dayofyear`	day since beginning of the year (integer)	[days]
`cgt_hour`	hours since midnight (fractional)	[hours]
`cgt_iteration`	number of current iteration in the iterative loop (integer)	[1]

4 Code templates and code generation

4.1 Generating Code with CGT

4.2 Creating code templates

This section is not ready yet. Please take a look at the examples which exist and refer to the following list of tags:

```
List of allowed tags for CGT code templates
-------------------------------------------

all conditions also work with /= instead of =

<codegen_version>
<now>
<noNewLine>

<numFlatTracers>
<numFlatTracers+n>    n=1..9
<num3DTracers>
<num3DTracers+n>      n=1..9
<numMovingTracers>
<numMovingTracers+n>  n=1..9

<constants name=>
  <name>
<trimName>
  <value>
  <description>
  <comment>
</constants>

<tracers         name=; vertSpeed=; opacity=; vertLoc=; isPositive=; childOf=; hasTimeTendenci
<backwardTracers name=; vertSpeed=; opacity=; vertLoc=; isPositive=; childOf=; hasTimeTendenci
  <name>
<trimName>
  <description>
  <comment>
```

```
  <numFlat>
  <numFlat+n>   n=1..9
  <num3D>
  <num3D+n>     n=1..9
  <numMoving>
  <numMoving+n> n=1..9
  <vertSpeed>
  <vertSpeedValue>
  <-vertSpeedValue>
  <opacity>
  <childOf>
  <childOfNumMoving>
  <timeTendencies vertLoc=>
    <timeTendency>
    <description>
  </timeTendencies>
  <children>
    <childIndex>
    <childName>
    <index>
    <name>
  </children>
  <initValue>
  <molarMass>
  <solubility>
  <schmidtNumber>
  <ceTotalIndex>
  <ceAgedIndex>
  <ceTotalName>
  <ceAgedName>
  <ceAmount>
</tracers>
</backwardTracers>

<auxiliaries name=; calcAfterProcesses=; calcBeforeZIntegtal=; vertLoc=; isZGradien
  <name>
<trimName>
  <temp1> ... <temp9>   (lines containing these tags are deleted if temp1...temp9='
  <formula>
  <description>
  <iterations>
  <iterInit>
</auxiliaries>
```

```
<processes name=; vertLoc=; isOutput=; isStiff=; processType=>
  <name>
<trimName>
  <description>
  <turnover>
  <comment>
  <stiffFactor>
  <stiffTracer>
  <processType>
</processes>

<cElements isTracer= ;isAging=>
  <total>
  <totalTop>
  <totalBottom>
  <totalBottomNumFlat>
  <aged>
  <agedTop>
  <agedBottom>
  <agedBottomNumFlat>
  <totalIndex>
  <totalIndexTop>
  <totalIndexBottom>
  <agedIndex>
  <agedIndexTop>
  <agedIndexBottom>
  <totalIndexNum>
  <totalIndexTopNum>
  <totalIndexBottomNum>=<totalBottomNumFlat>
  <agedIndexNum>
  <agedIndexTopNum>
  <agedIndexBottomNum>=<agedBottomNumFlat>
  <containingTracers vertLoc=; vertSpeed=>
    <ct>
    <ctNumFlat>
    <ctNumMoving>
    <ctAmount>
    <ctIndex>
    <ctIndexNum>
  </containingTracers>
</cElements>
```

5 The positive-definite time-stepping scheme

Here it shall be described how the positive-definite time-stepping scheme works. An article about it is under revision. We will cite it here if it is out.

5.1 Making a Euler-Forward time step positive

5.2 Treating stiff problems with modified Patankar methods

6 Using colored elements

6.1 Theory of element tagging

The approach has been described by [Menesguen et al., 2006] Please refer to [Radtke et al., 2012] for the theory of element tagging.

6.2 Practice of element tagging in CGT

6.3 Theory of age attribution

The age-attribution technique is described in [Deleersnijder et al., 2001]. Please refer to [Radtke et al., 2012] for the application of the age-attribution technique to chemical elements in an ecosystem model.

6.4 Practice of age attribution in CGT

7 Using special features in CGT

7.1 Automatic balancing of process equations

7.2 Using combined tracers (e.g. alkalinity)

A tracer with `isCombined=1` is a strange kind of tracer that needs some detailed explanation. Unlike other tracers which have elements as their contents, this tracer has other tracers (child tracers) as its contents. It represents some kind of sum over these child tracers.

We take the example of "total alkalinity" which is used in the CO2 add-on to explain how this tracer works. Total alkalinity is defined as the amount of a strong acid needed to titrate a solution to a pH of 4.3. In practice, it can be calculated as a sum of concentrations of several ions. In our case, the following formula gives a good approximation:

$$\begin{aligned} \texttt{t\_alk} &= [OH^-] - [H_3O^+] + [HCO_3^-] + 2[CO_3^{2-}] + [B(OH)_4^-] \\ &\quad +2[PO_4^{3-}] + [HPO_4^{2-}] - [H_3PO_4] + [HS^-] \end{aligned} \tag{7.1}$$

As we can see, some concentrations (like $[PO_4^{3-}]$) are considered as tracers in our model (in this case, `t_po4`). For other concentrations, e.g. $[H_3O^+]$ or $[OH^-]$, this is not possible, as the reaction $OH^- + H_3O^+ \rightarrow 2H_2O$ is very quick. However, these fast reactions do not change total alkalinity.

We define total alkalinity as an `isCombined=1` tracer. Now if some processes create or consume e.g. H_3O^+, this will change the value of total alkalinity, even if H_3O^+ is no tracer in our model. In other words, processes which consume or produce any of the "contents" of total alkalinity will generate a time tendency for the total alkalinity tracer.

7.3 Vector tracers

7.4 Pseudo-3d tracers

7.5 Working with add-ons

7.5.1 Why use add-ons?

Ecosystem models differ in complexity. While it is desirable for general application to keep the model as simple as possible, using the model for a special research question may require resolving more details of a special process, a special trophic level etc. For example, "carbon pump" estimations require the representation of the carbon cycle,

making it necessary to include a DIC and total alkalinity tracer. Maintaining two models - one with carbon cycle and one without - however, bears the risk of divergence as one of them is improved.

A better way is the following: All basic processes are maintained in one standard model. All carbon-cycle processes are described in an add-on which can be loaded on top of the standard model. It then extends the original model.

7.5.2 How are add-ons stored

General explanation

Add-ons look very much like formalized ecosystem models - they consist of the same set of textfiles (`modelinfos.txt, constants.txt, tracers.txt` ...). However, they only contain the modifications needed to the original model.

We make an example: The original `constants.txt` looks like this:

```
name = t_low
value = 10.0
description = lower temperature limit [degC]
**************************
name = t_up
value = 20.0
description = upper temperature limit [degC]
```

The `constants.txt` in the add-on looks like this:

```
name = t_low
value = 5.0
**************************
name = s_low
value = 1.0
description = lower salinity limit [g/kg]
**************************
name = delete_in_add_on_t_up
```

The first entry will update the constant `t_low` which already exists (as the name is the same). It will change its value to 5.0. The description will not be changed.

The second entry defines a new constant.

The third entry is a command to delete the constant `t_up`.

Details on add-on text files

- `modelinfos.txt` - this works differently, the original modelinfos are forgotten and only those from the add-on are applied.
- `auxiliaries.txt` - The ordering of the auxiliaries may be changed, because `cgt` and `cgt_edit` automatically try to place them in the order in which they are

calculated.

- `tracers.txt` - When any of the contents has changed, all of them are saved again in the add-on. That means that contents only occurring in the original file are then forgotten.
- `processes.txt` - When any of the limitations has changed, all of them are saved again in the add-on. That means that limitations only occurring in the original file are then forgotten. Use the line `limitations=0` to delete all limitations of a process.

How to use add-ons

This is very simple. If you load your text files in `cgt`, you will be asked whether you want to load an add-on. Click yes and load the `modelinfos.txt` of the add-on. You may load more than one add-on, but be aware that the order in which you load them may make a difference. Then, create your code as normally.

How to create add-ons

- **creating a new add-on**
  First, create a copy of the original model you want to extend. Load it in `cgt_edit`. Then, apply your changes and extend your model. It will be saved as an extended model, not as an add-on.

  Then, click "open a set of text files as reference for comparison" and load the original model (not the copy of course). The program `cgt_edit` will now indicate what you have changed compared to the original model. These changes can be saved as an add-on. To do so, click the button "save differences as add-on".
- **modifying an existing add-on**
  Load the model and the add-on. You will be asked to save the extended model - do so. This storage of the extended model is only needed temporarily, you can delete it after you have finished, as you want an add-on and not an extended model.

  Click "open a set of text files as reference for comparison" and load the original model, but without the add-on. The program `cgt_edit` will now indicate all changes which the add-on makes to the original model.

  Do your modifications until you are done - the changes to the original model might become more. When you are done, click "save differences as add-on". Save your modified add-on to a new folder.

Bibliography

[Deleersnijder et al., 2001] Deleersnijder, E., Campin, J.-M., and Delhez, E. J. (2001). The concept of age in marine modelling: I. theory and preliminary model results. *Journal of Marine Systems*, 28(3–4):229–267.

[Menesguen et al., 2006] Menesguen, A., Cugier, P., and Leblond, I. (2006). A new numerical technique for tracking chemical species in a multisource, coastal ecosystem applied to nitrogen causing ulva blooms in the bay of brest (france). *Limnology and oceanography*, 51(1):591–601.

[Radtke et al., 2012] Radtke, H., Neumann, T., Voss, M., and Fennel, W. (2012). Modeling pathways of riverine nitrogen and phosphorus in the baltic sea. *Journal of Geophysical Research*, 117(C9):C09024.