Computer Science in Ocean and Climate Research

Lecture 2: Modularization of Climate Models

Prof. Dr. Thomas Slawig

CAU Kiel
Dep. of Computer Science

Summer 2020

Contents

1. Modularization of Climate Models
   - A Second Model
   - Different Time Integrators
   - Modularization
   - Modularization in a Structural Programming Setting
   - Modularization in an Object-Oriented Setting

Contents

1 Modularization of Climate Models
- A Second Model
- Different Time Integrators
- Modularization
- Modularization in a Structural Programming Setting
- Modularization in an Object-Oriented Setting

Mathematical formulation of a climate model

- Climate system is time-dependent. $\rightsquigarrow$ Mathematical formulation:
- **Initial value problem (IVP)** for a system of **ordinary differential equations (ODEs)**:
- A **differential equation** is an equation

  $$\dot{y}(t) = f(y(t), t), \quad t \geq t_0,$$

  for an unknown function $y : t \mapsto y(t)$ where the derivative of this function appears:

  $$\dot{y}(t) := y'(t) := \frac{dy}{dt}(t) := \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t}.$$

- Example: $y(t)$ (global mean) temperature at time t.
- An **ordinary** differential equation just has the derivative w.r.t. one variable (here t).
- In most climate models, y and f are vector-valued. Then we have a **system** of ODEs.
- Often the model function f (or rhd side of the equation) depends on some parameters.
- To be well-posed, an **initial value** y_0 has to be given:

  $$y(t_0) = y_0.$$

Example: Zero-dimensional Energy Balance Model (EBM)

- Models Earth as a point in space, no spatial resolution.
- Balance between incoming and outgoing radiation (energy).
- Typical modeling principle: balance equation (conservation property).
- Only variable: (global mean) temperature $y = y(t)$ as function of time.
- Resulting ODE:

$$\dot{y}(t) = C\left(\underbrace{\frac{S}{4}(1-\alpha)}_{\text{incoming}} - \underbrace{\epsilon\sigma y(t)^4}_{\text{outgoing}} \right)$$

- S: energy per surface area from solar radiation (solar "constant"), unit: Wm^{-2}.
- $\alpha \in (0,1)$: albedo = reflection of incoming radiation,
- $\epsilon \in (0,1)$: emissivity.
- σ: Stefan-Boltzmann constant.

Predator-prey model

- Example for different model
- Ecosystem with two species: x prey, y predator
- Developed for fish population in the Mediterranean Sea (Lotka and Volterra).
- ODE system:

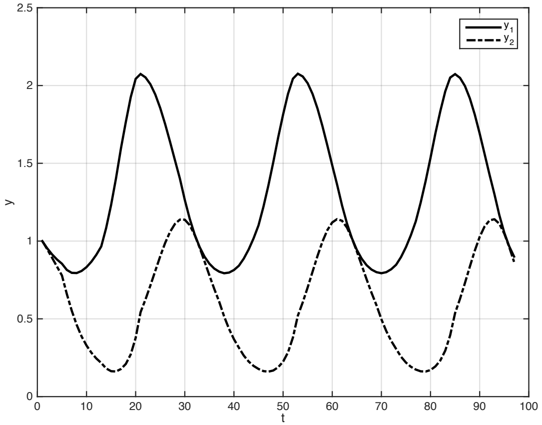
$$\begin{aligned}\dot{x} &= x(\alpha - \beta y - \lambda x)\\ \dot{y} &= y(\delta x - \gamma - \mu y), \quad t \geq 0.\end{aligned}$$

- $+$ initial values $x(0) = x_0, y(0) = y_0$.
- Easiest case: constants $\alpha, \beta, \gamma, \delta > 0, \lambda = \mu = 0$.

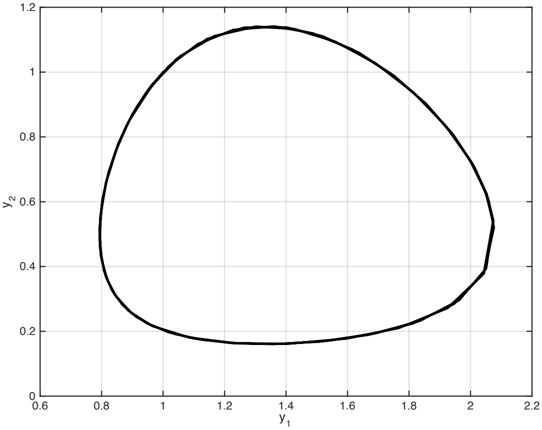
  $\rightsquigarrow$ x, y: periodic behavior,

  stationary points at $(x, y) = (0, 0)$ and $x = \frac{\gamma}{\delta}, y = \frac{\alpha}{\beta}$.
- $\lambda, \mu > 0$: additional terms for social interaction, logistic behavior.

Predator-prey model without quadratic terms

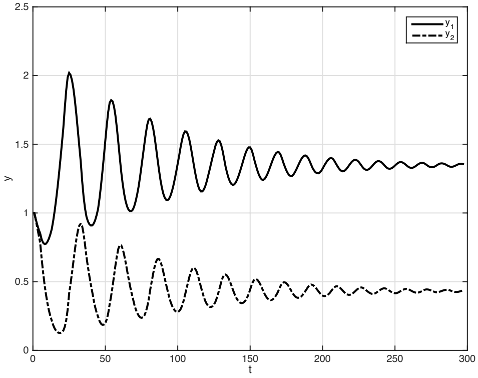


$x(t)$ and $y(t)$ vs. t

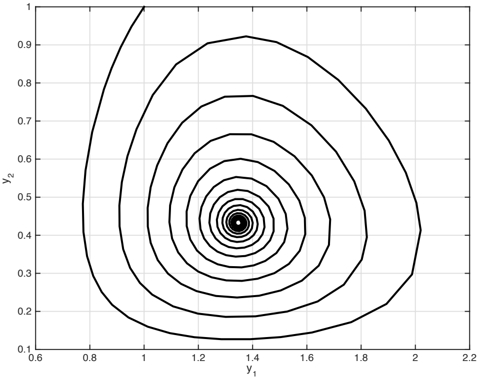


y vs. x

Predator-prey model with quadratic terms



$x(t)$ and $y(t)$ vs. t



y vs. x

Contents

1 Modularization of Climate Models
- A Second Model
- Different Time Integrators
- Modularization
- Modularization in a Structural Programming Setting
- Modularization in an Object-Oriented Setting

Ordinary Differential Equations: Discretization

- Discretize time:

$$t_0 < t_1 < \ldots < t_{k+1} = t_k + \Delta t < \ldots t_n = T$$

- Approximate derivative, replace limit process by use of fixed step-size:

$$\dot{y}(t_k) = \lim_{\Delta t \to 0} \frac{y(t_k + \Delta t) - y(t_k)}{\Delta t} \approx \frac{y(t_k + \Delta t) - y(t_k)}{\Delta t}. \tag{1}$$

- Define approximation $y_k \approx y(t_k)$ and use (1) in model ODE:

$$\frac{y_{k+1} - y_k}{\Delta t} \approx \dot{y}(t_k) = f(y_k, t_k)$$

$\leadsto$ Simplest algorithm: **(explicit) Euler method**:

$$y_{k+1} = y_k + \Delta t f(y_k, t_k), \quad k = 0, \ldots, n-1.$$

- Accuracy (under some assumptions on differentiability of f):

$$\text{exact solution – numerical solution:} \quad \|y(t_k) - y_k\| = \mathcal{O}(\Delta t), \quad k = 1, \ldots, n$$

Alternative interpretation: Time integrators for ODEs

- Integrate ODE
$$\dot{y}(t) = f(y(t), t)$$
from t_k to t_{k+1}:
$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} \dot{y}(t)dt = \int_{t_k}^{t_{k+1}} f(y(t), t)dt$$
- Approximate integral, e.g. by rectangular rule:
$$\int_{t_k}^{t_{k+1}} f(y(t), t)dt \approx (t_{k+1} - t_k)f(y_k, t_k) = \Delta t f(y_k, t_k).$$
- Setting again $y_k \approx y(t_k)$ gives
$$y_{k+1} - y_k = \Delta t f(y_k, t_k),$$
- This is again the (explicit) Euler method:
$$y_{k+1} = y_k + \Delta t f(y_k, t_k), \quad k = 0, \ldots, n-1.$$

Alternative time-integrator

- **Improved (explicit) Euler method**:

$$\left.\begin{array}{rcl} y_{k+\frac{1}{2}} & = & y_k + \frac{\Delta t}{2} f(y_k, t_k) \\ y_{k+1} & = & y_k + \Delta t\, f\left(y_{k+\frac{1}{2}}, t_k + \frac{\Delta t}{2}\right) \\ t_{k+1} & = & t_k + \Delta t \end{array}\right\} k = 0, 1, \ldots, n-1$$

- Can be summarized to:

$$y_{k+1} = y_k + \Delta t\, f\left(y_k + \frac{\Delta t}{2} f(y_k, t_k), t_k + \frac{\Delta t}{2}\right)$$

⇝ General form of a **one-step method**:

$$y_{k+1} = y_k + \Delta t\, \Phi(f, y_k, t_k, \Delta t)$$

- Higher accuracy (again under some assumptions on differentiability of f):

$$\|y(t_k) - y_k\| = \mathcal{O}(\Delta t^2), \quad k = 1, \ldots, n.$$

⇝ second order method.

Alternative interpretation: Improved (explicit) Euler method

- Integrate ODE again from t_k to t_{k+1}:

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} \dot{y}(t)dt = \int_{t_k}^{t_{k+1}} f(y(t), t)dt$$

- Approximate integral, but now by midpoint rule:

$$\int_{t_k}^{t_{k+1}} f(y(t), t)dt \approx \Delta t\, f\left(y\left(t_k + \frac{\Delta t}{2}\right), t_k + \frac{\Delta t}{2}\right).$$

- Value $y(t_k + \frac{\Delta t}{2})$ at midpoint not given, approximate it by Euler step:

$$y(t_k + \tfrac{\Delta t}{2}) \approx y_k + \tfrac{\Delta t}{2} f(y_k, t_k).$$

- This gives

$$y_{k+1} - y_k = \Delta t\, f\left(y_k + \tfrac{\Delta t}{2} f(y_k, t_k), t_k + \tfrac{\Delta t}{2}\right),$$

- This is again the improved Euler method.

Contents

1. Modularization of Climate Models
   - A Second Model
   - Different Time Integrators
   - Modularization
   - Modularization in a Structural Programming Setting
   - Modularization in an Object-Oriented Setting

Modularization

Many climate models have the following basic structure:

- init model
- run model, would mean here: time loop
    - using time integrator
    - which itself uses/needs the model function f
- finalize model

Modularization of time-dependent climate models

Aims of modularization:

- Flexibility w.r.t climate model/ODE: It shall be possible to integrate
  - arbitrary ODE system
  - of arbitrary dimension
  - on arbitrary time interval
  - with arbitrary initial value
  - and additional model parameters (in f)
- Flexibility w.r.t. time integration method:
  - stepsize
  - type of method

Examples here:

- Energy Balance Model and Predator-prey model
- Explicit Euler and improved Euler method
- Different initial values, parameters, time intervals, step-sizes.

Modularization: "structural" vs. object-oriented programming

- Structural (procedural) programming:
  - Separation of data and functionality (operations, functions)
  - Functions operate on data
  - Clearly defined interfaces/signatures:
    ```
    real y = function do_something(real x)
    ```
  - Usage: function operates on data:
    ```
    real x,y
    y = do_something(x)
    ```
- Object-oriented programming:
  - Combination of data and functionality (operations, functions) in **classes**
  - **Methods (member functions)** defined inside class with clear interfaces:
    ```
    class ClassA
      doSomething(real x)
    ```
  - Usage: "methods send signal to object"
    ```
    ClassA y = new ClassA()
    A.doSomething(x)
    ```

Contents

1 Modularization of Climate Models
- A Second Model
- Different Time Integrators
- Modularization
- Modularization in a Structural Programming Setting
- Modularization in an Object-Oriented Setting

Modularization: Possible functions in a structural programming setting

- Functions for each time integrator:
    - function euler
    - function improved_euler
    - Interface: What are necessary parameters?
    - What do these functions return?
- Functions for each climate model/rhd side f:
    - function ebm
    - function predator_prey
    - Interface: What are necessary parameters?
    - What do these functions return?
    - Where and how will these functions be used?
- Where and how are model parameters set ...
- ... and how are they passed to the model function f?
- Where are parameters like the time-interval $[t_0, T]$ set?
- Is there one "main" function that gets model function and time integrator function as parameters?

Contents

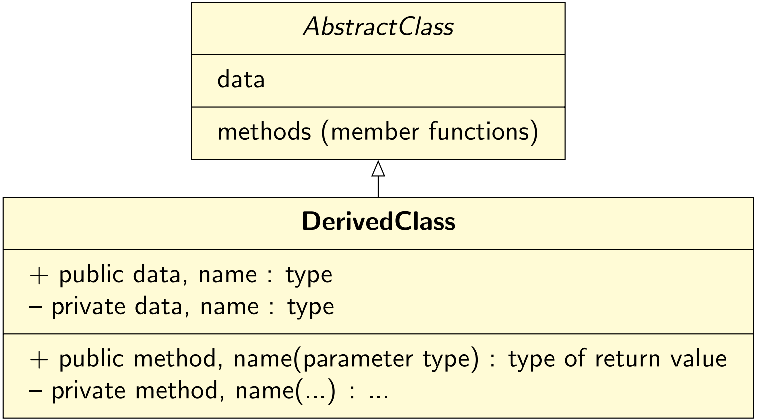
1. Modularization of Climate Models
   - A Second Model
   - Different Time Integrators
   - Modularization
   - Modularization in a Structural Programming Setting
   - Modularization in an Object-Oriented Setting

Modularization in an Object-Oriented Setting

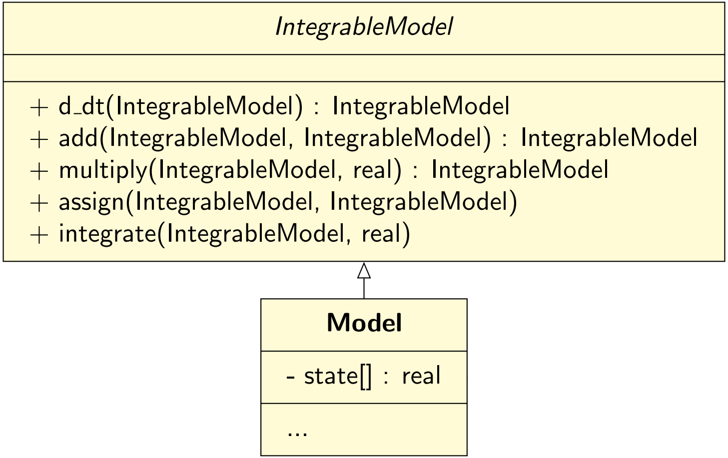
- Object-Orientation: Paradigm of programming
- Idea: Combine data and operations on data (functions) to **classes**
- ... since operations depend on the data that they are applied on.
- Example: Multiplication means different things for real, complex numbers, vectors, matrices ...
- One main feature of OO: inheritance:
  - Define common data and functionality (operations, functions) in superclass
  - allow specialization/extension in subclass(es)
- Might be useful also for our examples in climate modeling ...
- We have ...
  - several model functions ... what do they have in common?
  - several time integrators ... what do they have in common?
- Additional feature: Access control for data and functionality.

Unified Modeling Language (UML): Class (abstract data type) diagram

Abstract class and derived class (inheritance)

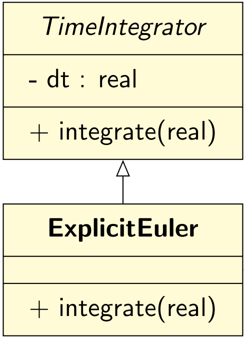


Semi-discrete pattern: UML class diagram

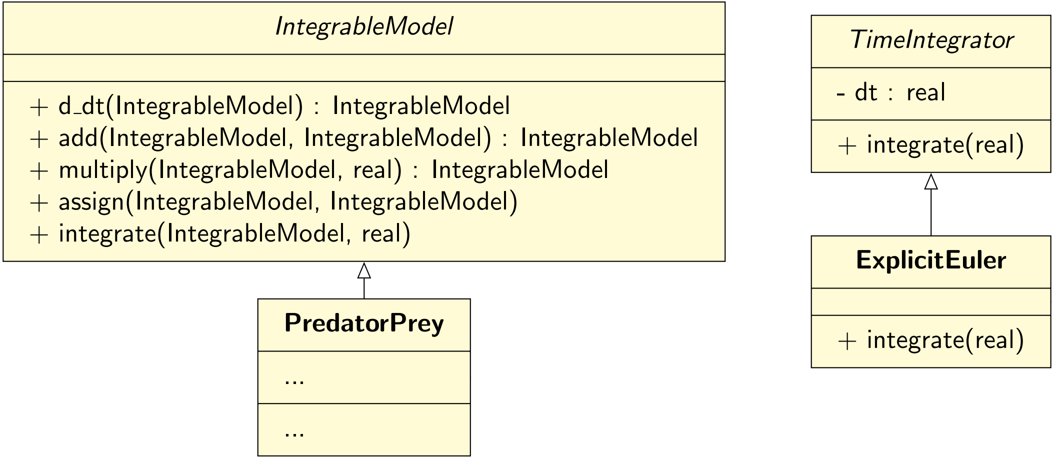


Strategy pattern: UML class diagram

- Time integration strategy



Semi-discrete + strategy pattern: UML class diagram



Idea and basic structure from: Rouson, Adelsteinsson, Xia: "Design Patterns for Multiphysics Modeling in Fortran 2003 and C++", ACM TOMS 37(1), 2010.

What is important

- Climate models have the same general structure.
- Time integrators have also a similar structure ...
- ... but different accuracy and effort.
- They can be motivated also by integrating the ODE and then approximating the integral.
- A flexible software framework is useful for time integration of different models.
- It can be realized in a structural and an object-oriented setting.
- The feature of inheritance of OO allows a hierarchical software design.