Computer Science in Ocean and Climate Research

Lecture 1: Introduction

Prof. Dr. Thomas Slawig

CAU Kiel
Dep. of Computer Science

Summer 2020

Contents

Contents

Computer Science for Ocean and Climate Research

- What is the topic of this lecture?

  Methods, techniques and algorithms that are used in the field of ocean/climate science.
- Why do we study this?

  Climate science is an important and growing research area

  Motivated by (discussion about) global warming

  Simulation and data becoming more and more important

  $\rightsquigarrow$ Growing need for methods of Computer Science

Computer Science for Ocean and Climate Research

- How does Computer Science contribute to climate research?

  Data analysis, simulation, parameter estimation

  Programming, software engineering, parallelization High performance computing (HPC)
- What if we can use these methods?

  Making simulations faster (or feasible at all)

  Analyzing and using data

  Couple models in a flexible way

  Improve model structure (sustainable software)

  Ocean: important part of climate system and research focus in Kiel (GEOMAR)

Topics of the module

- The climate system
- Basic structure of climate models
- Basic structure of climate simulation
- Methods of Computer Science and their usage / importance
- Interdisciplinary work

Contents

The climate system

- Difference climate $\leftrightarrow$ weather: Different scales in space and time
- Climate system is forced externally basically only by solar radiation
- ... that is changing on long time scales due to variation in the Earth's orbit
- Annual changes due to the Earth's orbit,
- daily changes due to the Earth's rotation,
- ... both resulting in different behavior in different regions of the Earth.
- Climate system is in a "stable" dynamical state, i.e., we observe

  ... an annual cycle (Earth moving around the sun)

  ... a daily cycle (day/night, Earth's rotation).
- But there exists additional internal variability (e.g., El Niño).
- Long-time differences in climate (glacial cycles).

Long-time differences in climate

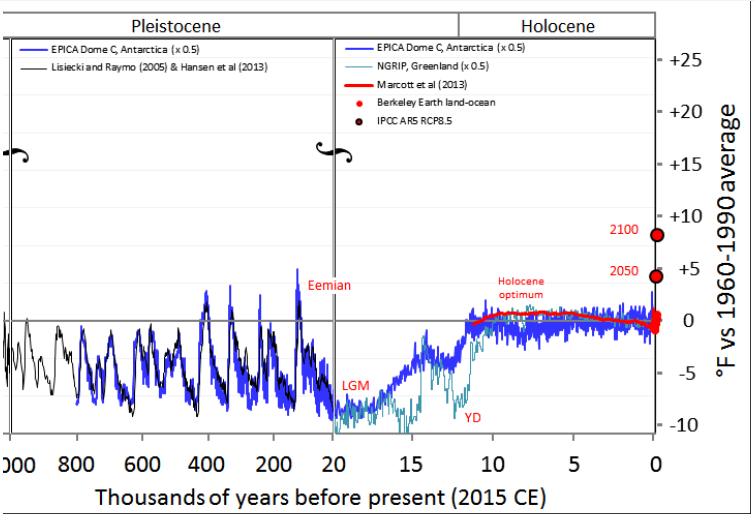


Image: Glen Fergus, License: Creative Commons BY-SA 3.0

Global warming: global temperature $\leftrightarrow$ atmospheric CO_2

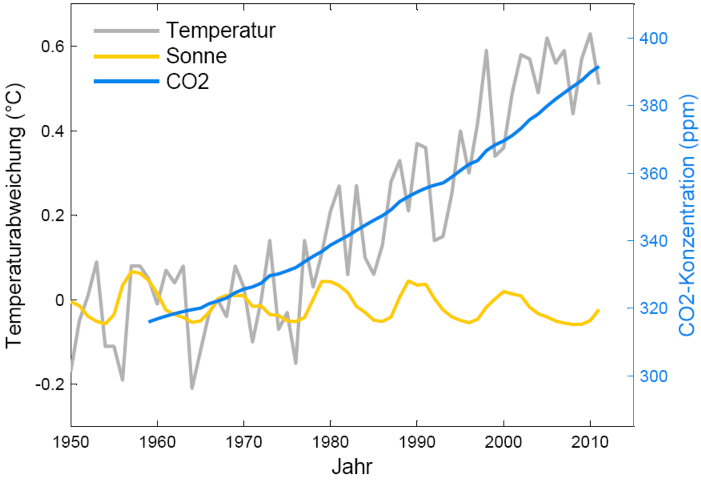


Image: Stefan Rahmstorf, License: Creative Commons BY-ND
`www.scilogs.de/klimalounge/die-populaerste-trickgrafik-der-klimaskeptiker-vahrenholt/`

Example: Internal variability

- Internal variability can appear even though the system data are time-independent.
- Example: Oscillation of a spring or pendulum without friction.
- $y(t)$ spatial coordinate at time t.
- Behavior described by a differential equation (simplified):

$$\underbrace{\ddot{y}(t)}_{\text{second derivative, acceleration}} + \omega^2 y(t) = 0, \quad \omega \in \mathbb{R} \text{ given constant}$$

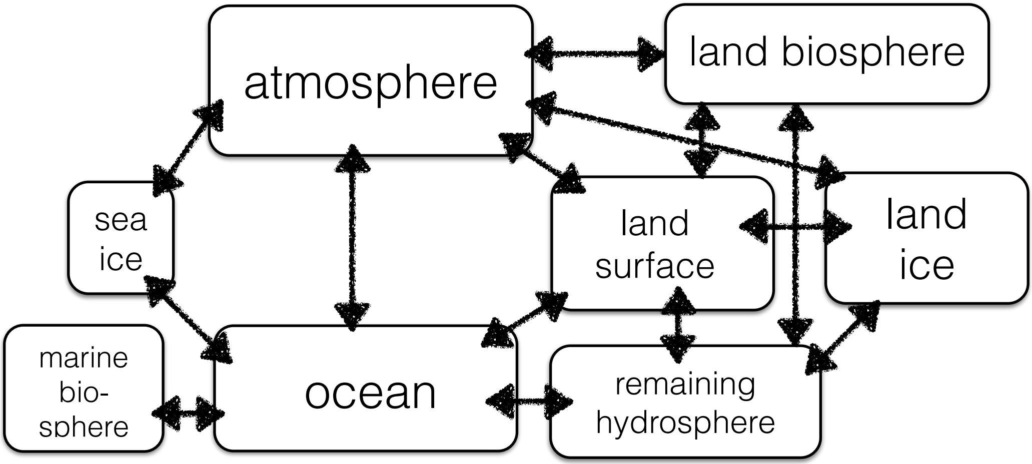
- Initial values (example): $y(0) = 1$ (fixed initial position),$\dot{y}(0) = 0$ (zero initial velocity).
- No periodicity in the equation or in the data, ...
- ... but in the solution, which is:

$$\begin{aligned} y(t) = \cos(\omega t) \quad &\Rightarrow \quad \dot{y}(t) = -\omega \sin(\omega t) \\ &\Rightarrow \quad \ddot{y}(t) = -\omega^2 \cos(\omega t) = -\omega^2 y(t) \end{aligned}$$

- Second initial condition $\dot{y}(0)$ needed for uniqueness of the solution, otherwise $y(t) = \cos(\omega t) + ct$ would be another one.

Contents

Components of the climate system



Contents

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:

$$\dot{y}(t) = c_1 S(1-\alpha) - c_2 y(t)^4 = f(y(t), t)$$

- S: energy per surface area from solar radiation (solar "constant"), unit: Wm^{-2}.
- $\alpha \in (0,1)$: albedo = reflection of incoming radiation,
- c_1, c_2: constants.
- This is a **continuous** model (time t is continuous, time derivative still present).

Contents

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:

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

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

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

Time-discrete version of an ODE

- **(Explicit) Euler method**:
  Simplest time-stepping scheme for $t \in [t_0, T]$:

$$\left.\begin{array}{rcl} y_{k+1} & = & y_k + \Delta t f(y_k, t_k) \\ t_{k+1} & = & t_k + \Delta t \end{array}\right\} k = 0, 1, \ldots, n-1, \quad \Delta t = \frac{T - t_0}{n}$$

- (Nearly) all climate models have this structure.
- Result: $(y_k)_{k=0}^n$ approximation of solution $y_k \approx y(t_k)$.
- Solution y_k can be a scalar (see example above, EBM) ...
- ... or vector, examples:
  - temperature + pressure + ... (ocean model)
  - temperature + ... at **different points in space**.

$\rightsquigarrow$ High-dimensional data: time- and space dependent (4D), several variables.

Components of the climate system and time loop visible in a climate model

```
c...1) MOdel initialisation

c     ==================
       call INI_CLIMBER
c     ==================

c...2) Time integration

c     ==================
       call TIME_LOOP
c     ==================

       stop
       end
```

main program

```
*************************************
*
*  NTS  - number of step (days)
*
*************************************

       do NTS=1,NTSMX

c      ==============
        call time_step
c      ==============

       enddo

      return
      end
```

time loop

```
c... Atmosphere model & ASI

c       =========
        call ATM
c       =========

c... Ocean fluxes

c      ==============
        call COUPLER
c      ==============

c... Ocean model & sea ice

      if (KOCEAN.eq.1.and.KCOUP.eq.1)  then

c      ==================================
       if (KOCN.ne.0) then
        if (KOCN.eq.2.or.NYR.le.10) then
         call MUZON
        else
         call SLAB
        endif
       endif
```

time step

Contents

Concepts and paradigms of programming languages

- imperative
- functional
- logical
- query languages
- procedural (structural)
- object-oriented
- modern languages are combining/integrating different paradigms

Compiled vs. interpreted (or scripting) languages

- Compiled languages:

	compiler		linker		execution	
source code	→	object code	→	machine code (executable)	→	result
Ex.: C: (.c)		(.o, .obj)		(.exe, a.out)		

Examples: C, C++, Fortran.

- Interpreted (scripting) languages:

	interpreter
source code (script)	→ machine code (internal) → result
Example Matlab®: (.m)	

Examples: Python, Matlab®, octave, R.

Compiled vs. interpreted (or scripting) languages

- Mixtures:

	compiler		virtual machine			
source code	$\rightarrow$	byte code	$\rightarrow$	machine code (internal)	$\rightarrow$	result
Java: (.java)		(.class)				

- Python also generates byte code (.pyc) in the first interpretation of the script.
- Compiled languages are fast(er) in runtime $\leftrightarrow$ interpreted languages are faster for writing and testing (often no variable declaration necessary)
- Combine both advantages: Just-in-time compilation, compile during runtime
- Store internal machine code in the first run, for faster execution in following runs (e.g., Matlab®).

Scientific programming: languages

- Most scientific programming takes place using **higher programming languages** (sometimes also: 3rd generation languages):
  - Examples: Fortran, C, C++, Java, Pascal/Delphi.
  - More abstraction than machine languages, assembler
  - Control structures
  - Faster coding
  - Have to be compiled or interpreted to obtain machine code.
  - $\rightsquigarrow$ platform/machine/compiler dependency
- General purpose languages $\leftrightarrow$ domain specific languages (DSL)
- DSL: Languages with even higher abstraction level:
  - Examples: Matlab®, octave, R.
  - Faster prototyping, usually slower
  - Ideas to combine both advantages: Julia, just-in-time compilation.
- Climate models are high-dimensional $\rightsquigarrow$ higher programming languages for simulation runs.
- Scripting languages used for simple models, visualization, data analysis.

What is important

- The climate system is a dynamic system with several components.
- Both properties are reflected in the structure of climate models:
  - transient models, time loops,
  - coupled models with different (software) components
- Two mathematical formulations:
  - IVP for ODE systems: continuous (in time)
  - discrete in time, approximating the time derivative with fixed time step-size.
- On the computer we can only use the time-discrete version.
- Simplest method is the Explicit Euler method.
- Programming climate models:
  - higher programming languages for fast runtime,
  - scripting languages for prototyping and visualization, data analysis.