Computer Science in Ocean and Climate Research

Lecture 10: Model Parameter Optimization

Prof. Dr. Thomas Slawig

CAU Kiel
Dep. of Computer Science

Summer 2020

Contents

Model Parameter Optimization

- What is it?

  Finding model parameters such that the model output matches given observational data
- Why are we studying this?

  Method to improve the model quality
- How does it work?

  Defining a meaningful cost function that measures the model-to-data misfit

  Applying a mathematical optimization algorithm

  Approximation or exact computation of the model derivative w.r.t. the parameters
- What if we can use it?

  Improve model output

  Check model quality

  Determine new measurement locations that further improve model quality

Contents

Parameter-dependent models: Fully discrete setting

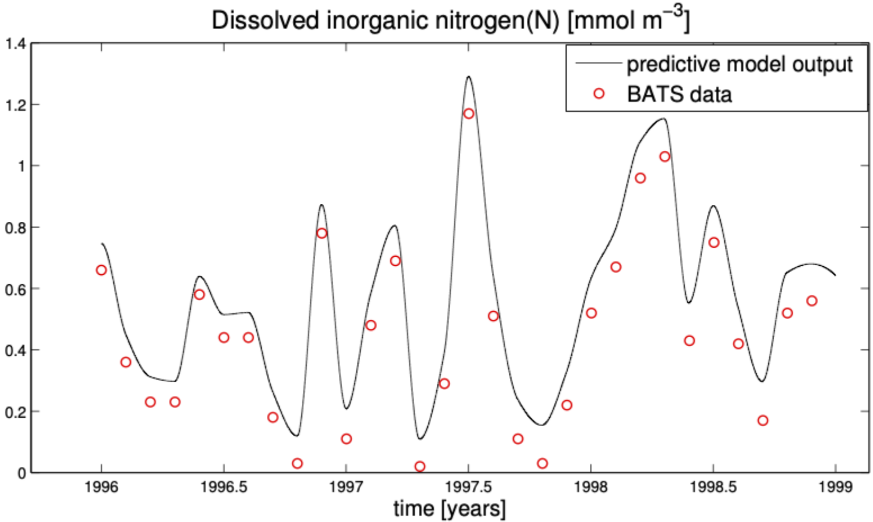
- We use the following general fully-discrete form of a climate model as

$$y_{k+1} = y_k + \Delta t\, \Phi(f, p, y_k), \quad k = 0, \dots, N-1.$$

  + initial values.
- For simplicity of notation, we omit the dependency of Φ on
  - time step-size Δt,
  - time $t_k, t_{k-1}, \dots$,
  - eventually used additonal values $y_{k-1}, \dots$
- Solution depends on model parameters p.
- The model shall reproduce “reality” in the best possible way.
- We have measurement data $z_k, k = 1, \dots, N$.
- Aim: find parameter(s) p such that

$$y_k \approx z_k, \quad k = 1, \dots, N.$$

Example: Model-to-data misfit: 1-D model



Simple example: Direct solution of the parameter optimization problem

- Zero-dimensional Energy Balance Model (EBM):
- Only variable: (global mean) temperature $y = y(t)$ as function of time:

$$\dot{y}(t) = \frac{1}{C}\left(\frac{S}{4}(1-\alpha) - \sigma\epsilon y(t)^4\right) = f(y(t))$$

  with coupling constant C, Boltzmann constant σ, solar constant S, and albedo α given.
- Want to find the correct value of the emissivity ϵ such that steady state (with $f(y) = 0$) matches current real value ($z \approx 287$ K).
- Steady state satisfies:

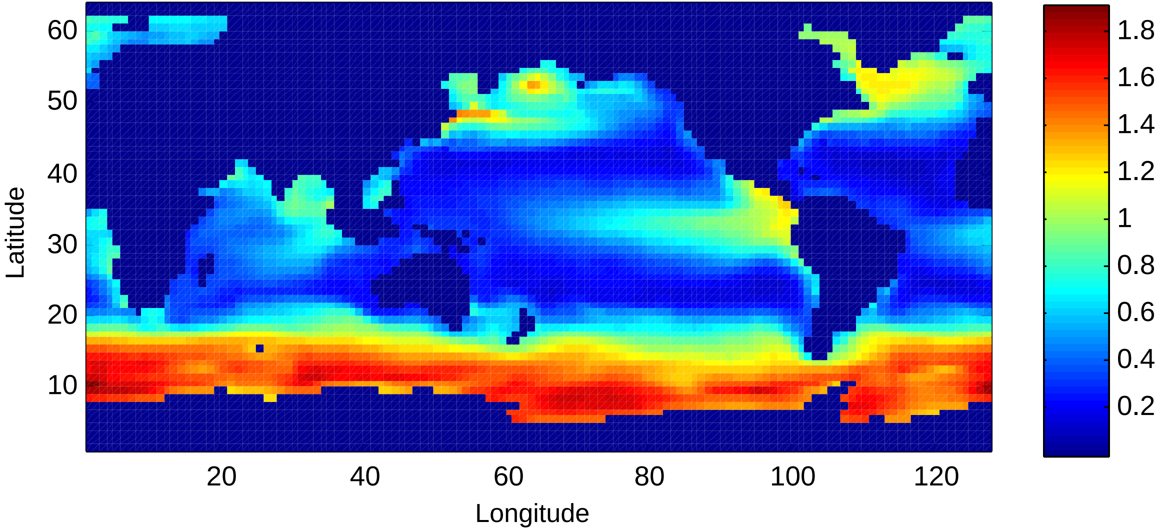
$$\frac{S}{4}(1-\alpha) - \sigma\epsilon y^4 = 0 \quad \Rightarrow \quad y = \sqrt[4]{\frac{S(1-\alpha)}{4\sigma\epsilon}}$$

$\rightsquigarrow$ Compute ϵ from

$$\epsilon = \frac{S}{4\sigma z^4}(1-\alpha)$$

- Clearly not that easy for more complex models.

3-D model, model output at surface



Contents

General setting

- Model has several parameters, here summarized in a vector

$$p \in \mathbb{R}^n.$$

- Measurement data are usual 3-D or 4-D fields , here also in a vector

$$z \in \mathbb{R}^m.$$

- Model output is 3-D or 4-D, consists of several variables, here stored in some arbitrary data structure Y.
- Observation operator C, maps the output to a vector:

$$Y \mapsto CY \in \mathbb{R}^m.$$

- It selects corresponding values of the output Y, corresponding to the variables and the points in space and time where there are measurements available.
- It optionally performs averaging (e.g., compare global mean temperature).

Example: Predator-prey model

- ODE system:

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

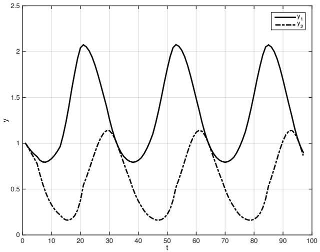
- Parameters: $p = (\alpha, \beta, \gamma, \delta, \lambda, \mu)$.
- Output of discretized model:

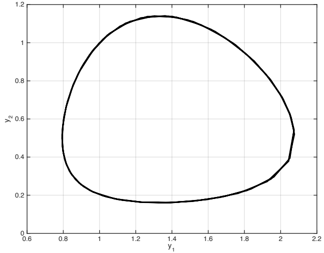
$$Y := (x_k, y_k)_{k=0}^{N} \in \mathbb{R}^{2\times(N+1)}, \quad N : \#\ \text{time steps}$$

- Example: consider only last value of prey, i.e., x_N.

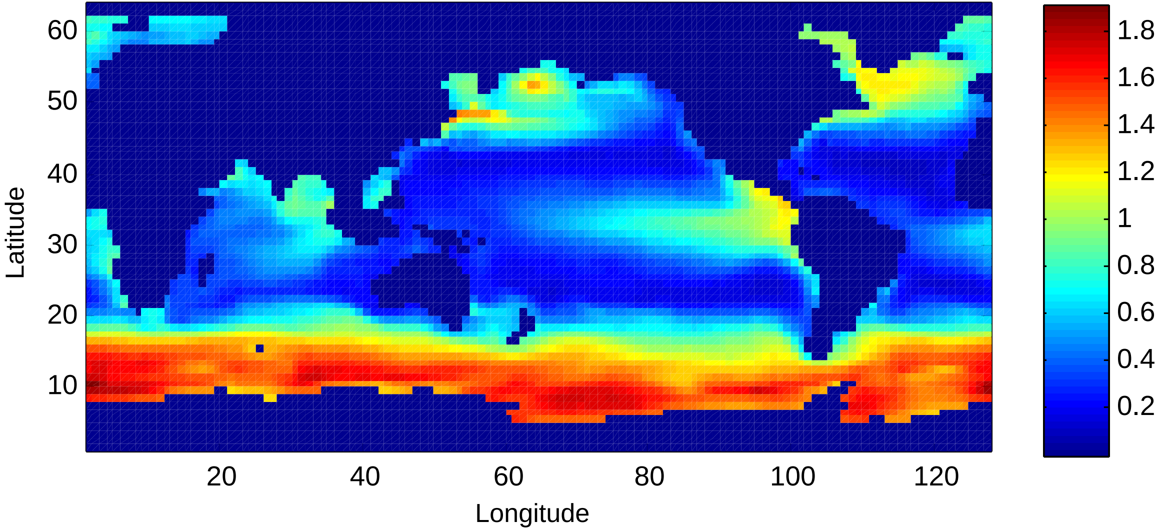
⇝ Observation operator

$$C : Y \mapsto x_N \in \mathbb{R} \quad (m = 1).$$

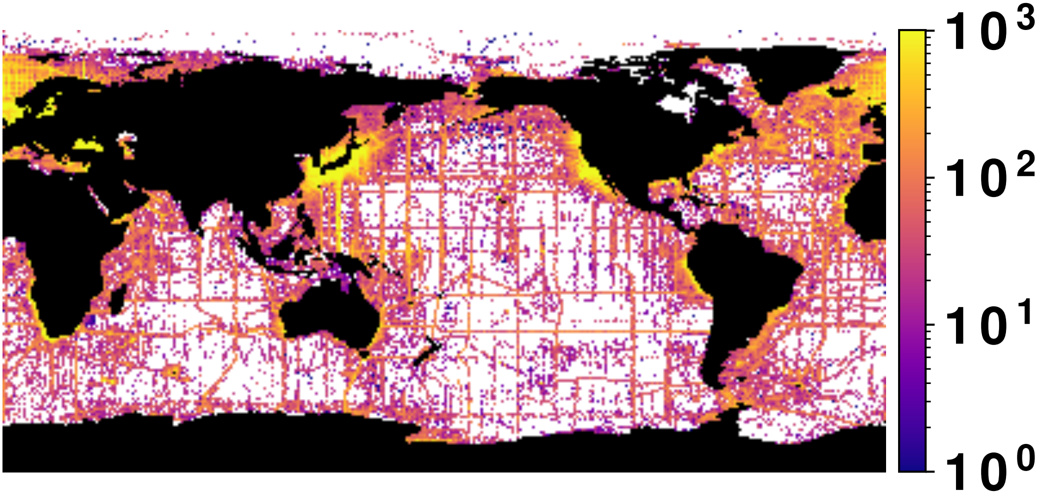




Model-to-data misfit: 3-D model, model output at surface



Model-to-data misfit: 3-D model, number of measurements at surface



Contents

Measuring the model-to-data misfit

- We write the output of the observation operator as

$$CY =: y \in \mathbb{R}^m.$$

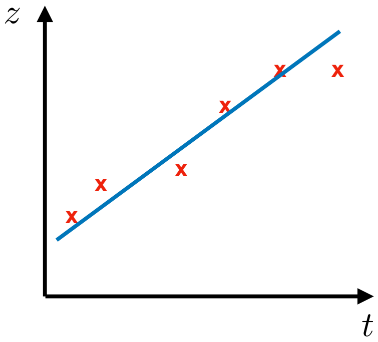
- These values we want to compare with measurement data $z \in \mathbb{R}^m$.

$\rightsquigarrow$ Minimize distance between data and model output:

$$\min_{p \in \mathbb{R}^n} \frac{1}{2} \sum_{k=1}^{m} (y_k - z_k)^2.$$

- Model output depends on parameters p:

$$y = y(p).$$



Best fit of a model (here: affine-linear function) to data.

- Factor $\frac{1}{2}$ just used for easy notation $\rightsquigarrow$ derivative computation below.

Generalization

- Above: Standard least-squares **cost function**:

$$\frac{1}{2}\sum_{k=1}^{m}(y_k - z_k)^2 = \frac{1}{2}\|y - z\|_2^2 = \frac{1}{2}(y - z)^\top (y - z).$$

- Different weights for different measurements $\rightsquigarrow$ weighted least-squares cost function:

$$\frac{1}{2}\sum_{k=1}^{m}\frac{1}{\sigma_k^2}(y_k - z_k)^2 = \frac{1}{2}\|y - z\|_{\Sigma^{-1}}^2 = \frac{1}{2}(y - z)^\top \Sigma^{-1}(y - z), \quad \Sigma := \mathrm{diag}(\sigma_k^2) \in \mathbb{R}^{m\times m}.$$

  where σ^2 is the **variance** of measurement z_k.

- More general: Include interdependency of different measurements.

$\rightsquigarrow$ Generalized least-squares cost function:

$$\frac{1}{2}\|y - z\|_{\Sigma^{-1}}^2 := \frac{1}{2}(y - z)^\top \Sigma^{-1}(y - z).$$

Here, include **covariance matrix** $\Sigma \in \mathbb{R}^{m\times m}$.

Stochastic interpretation

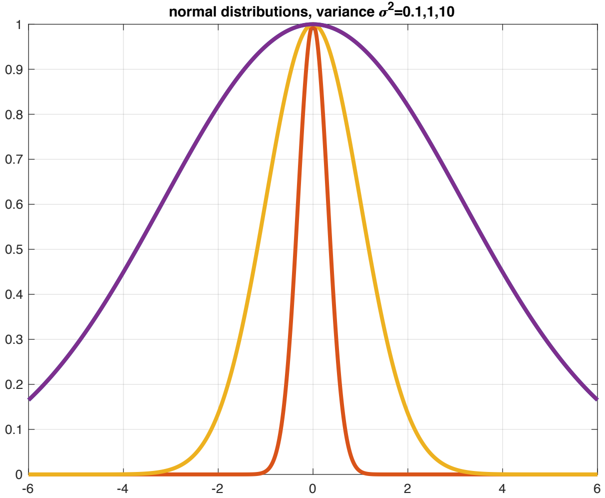
- Minimizing the weighted least-squares cost function:

$$\min_p \sum_{k=1}^m \frac{(y_k - z_k)^2}{2\sigma_k^2} \Leftrightarrow \max_p \left(-\sum_{k=1}^m \frac{(y_k - z_k)^2}{2\sigma_k^2} \right) \Leftrightarrow \max_p \exp\left(-\sum_{k=1}^m \frac{(y_k - z_k)^2}{2\sigma_k^2} \right)$$
$$\Leftrightarrow \max_p \prod_{k=1}^m \exp\left(-\frac{(y_k - z_k)^2}{2\sigma_k^2} \right)$$
$$\Leftrightarrow \max_p \prod_{k=1}^m \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(y_k - z_k)^2}{2\sigma_k^2} \right).$$

- **Probability density function** (pdf) of the normal distribution with variance σ_k^2.
- Interpretation: p maximizes the probability for $y = z$ (in the mean/expectation).

$\rightsquigarrow$ p is called "maximum-likelihood" estimate.

Pdf of normal distribution with different variances



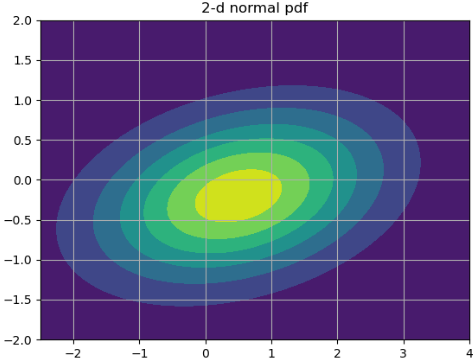
Same stochastic interpretation

- Generalized least-squares cost function:

$$\min_p \frac{1}{2}\|y - z\|^2_{\Sigma^{-1}} \Leftrightarrow \max_p \frac{1}{\sqrt{(2\pi)^n \det \Sigma}} \exp\left(-\frac{1}{2}\|y - z\|^2_{\Sigma^{-1}}\right)$$

- Covariance matrix Σ.
- Probability density function of the multi-variate normal distribution.
- Picture: pdf with

$$z = (0.5, 0.2), \Sigma = \begin{pmatrix} 2 & 0.3 \\ 0.3 & 0.5 \end{pmatrix}.$$



Using a prior estimate for the parameters

- Often, a certain value p_0 is known/given for the parameters (default/standard value).

⇝ Deviation from this value shall not get too big in the parameter optimization.

⇝ Add a second term to the cost function, e.g.,

$$\min_{p \in \mathbb{R}^n} \frac{1}{2}\|y - z\|^2_{\Sigma^{-1}} + \frac{1}{2}\|p - p_0\|^2_{\Sigma_p^{-1}}, \quad \alpha > 0.$$

where p_0 is called the **prior** and Σ_p the parameter covariance matrix.

- Stochastic interpretation (as above): optimal parameter p is now the value where the probability for $y = z$ and $p = p_0$ (in the mean/expectation) is maximized, taking into account:
  - the given spread/variances of both data and parameters,
  - the interdependency of the different measurements ...
  - ... and of the different parameters.

Contents

General form of an optimization method

- We need a method to solve the optimization problems above.
- Sometimes we have additional bounds on the parameters.

Algorithm (General descent method):

1. Choose initial guess $p_0 \in \mathbb{R}^n$.
2. For $k = 0, 1, \dots$:
    1. Choose a search direction $d_k \in \mathbb{R}^n$.
    2. Choose a step-size $\rho_k > 0$ that reduces the cost function f.
    3. Set $p_{k+1} = p_k + \rho_k d_k$.

   until a stopping criterion is satisfied.

Search directions:

- $d_k = -\nabla f(p_k)$: negative gradient of the cost.
- $d_k = -\nabla^2 f(p_k)^{-1} \nabla f(p_k)$: Newton method, using Hessian matrix (2nd derivatives).
- d_k: Quasi-Newton method, more efficient approximation of Newton direction.

Contents

Derivative computation

- We have seen: (most) optimization algorithms use derivative information.
- Parameter vector $p \in \mathbb{R}^n \rightsquigarrow$ Derivative of the cost function

$$f(p) := \frac{1}{2}\sum_{k=1}^{m}(y_k - z_k)^2.$$

is a vector of partial derivatives (gradient):

$$\nabla f(p) = \left(\frac{\partial f}{\partial p_i}(p)\right)_{i=1}^{n}.$$

- Model output depends on parameters: $y = y(p)$.

$\rightsquigarrow$ Derivative (gradient) of the cost function has to be computed via the chain rule:

$$\frac{\partial}{\partial p_i}\left(\frac{1}{2}\sum_{k=1}^{m}(y_k - z_k)^2\right) = \sum_{k=1}^{m}(y_k - z_k)\frac{\partial y_k}{\partial p_i} \Rightarrow \nabla f(p) = y'(p)^\top (y - z), \quad y' = \left(\frac{\partial y_k}{\partial p_i}\right)_{ki}.$$

Ways to compute derivatives

1. Analytical derivative: there is a formula or the model is that simple that we can anayltically compute the derivative:

   Example EBM, steady state:

$$y = \sqrt[4]{\frac{S(1-\alpha)}{4\sigma\epsilon}} = \left(\frac{S(1-\alpha)}{4\sigma\epsilon}\right)^{\frac{1}{4}}.$$

   Derivative of y w.r.t. parameter $p = \epsilon$:

$$y'(\epsilon) := \frac{dy}{d\epsilon} = \frac{1}{4}\left(\frac{S(1-\alpha)}{4\sigma\epsilon}\right)^{-\frac{3}{4}} \frac{S(1-\alpha)}{4\sigma}\left(-\frac{1}{\epsilon^2}\right).$$

   Not that easy in the general, realistic case.

2. Symbolical computation using some online tools/software: works also only for simple functions.

3. Finite-difference derivative approximation

- Components of the gradient $\nabla f(p)$ can be approximated by

$$\frac{\partial f}{\partial p_i}(p) \approx \frac{f(p+he_i)-f(p)}{h}, \quad i=1,\dots,n,$$

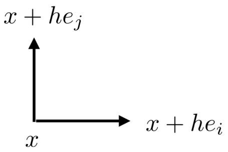
with $h>0$ fixed.

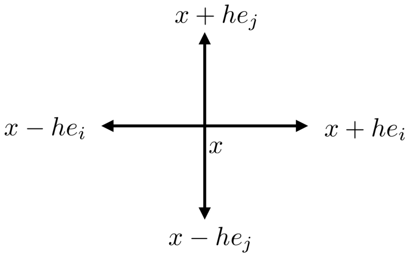
$\rightsquigarrow$ n additional evaluations of f.

- Central approximation for gradient:

$$\frac{\partial f}{\partial p_i}(p) \approx \frac{f(p+he_i)-f(p-he_i)}{2h}.$$

$\rightsquigarrow$ $2n$ additional evaluations of f.





4. Algorithmic/automatic differentiation (AD)

- Realistic case: cost function given as a computer program

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f : p \mapsto f(p).$$

⇝ f is in fact a concatenation of elementary functions of the used programming language:

$$f = F_k \circ \cdots \circ F_1$$

- Define the intermediate variables

$$x_0 := p, \quad x_i := F_i(x_{i-1}), i = 1, \ldots, k, \quad f(p) = x_k.$$

⇝ Derivative of f to be computed by the chain rule:

$$x_0' := I, \quad x_i' := F_i'(x_{i-1})x_{i-1}', i = 1, \ldots, k, \quad f'(p) = x_k'.$$

- The F_i are now elementary functions of the used language.

⇝ Their derivatives can be computed exactly.

- This can be done algorithmically (and efficiently).

Simple example: AD using source transformation

$$F(x) = \sqrt{x} \Rightarrow F'(x) = \frac{1}{2\sqrt{x}}.$$

```
real function f(x,y)
real x,y
y=sqrt(x)
f=y
return
end
```

original Fortran function code

```
r2_v = sqrt(x)

if ( x .gt. 0.0e0 ) then
   r1_p = 1.0e0 / (2.0e0 *  r2_v)
else
   call ehufSO (9,x, r2_v, r1_p,g_ehfid,32)
endif
g_y = r1_p * g_x
y = r2_v
--
g_f = y
```

part of algorithmically generated derivative code

Derivative of f to be computed by the chain rule:

$$x_i' := F_i'(x_{i-1})x_{i-1}', i = 1, \ldots, k.$$

Algorithmic/automatic differentiation (AD)

- Works for long and operational climate and weather forecast models.
- Two methods: source transformation (example above) ...
- ... or operator overloading:

```
module ADClass
    use ISO_FORTRAN_ENV
    implicit none

    type, public :: AD
        private
        real(real64) :: val, der
    end type AD
```

```
! times operators:
elemental type(AD) function times(advar, x)
    type(AD), intent(in) :: advar, x
    times%val = advar%val * x%val
    times%der = x%val * advar%der + advar%val * x%der
end function
```

- Different software tools for Fortran, C, C++, Matlab® ...
- The obtained derivatives are exact (in contrast to the finite-difference derivatives).

What is important

- Parameter optimization is an important method to improve model results.
- The model output is compared to available data.
- We try to find the parameters that provide the best model-to-data fit.
- For this purpose, a cost function that measures the misfit is defined.
- There are several options for the cost function.
- Some of them take into account data uncertainties and allow for a stochastic interpretation.
- We apply iterative optimization algorithms that approximate the optimal parameters.
- Many of these algorithms use derivative information.
- Derivatives can be computed via finite-difference approximations ...
- ... or applying software tools for Algorithmic/Automatic Differentiation.