Computer Science in Ocean and Climate Research

Lecture 9: Ensemble Computations

Prof. Dr. Thomas Slawig

CAU Kiel
Dep. of Computer Science

Summer 2020

Contents

Ensemble Computations

- What is it?

  Performing a series of runs of a model with a set of parameters, forcing data, initial values ...
- Why are we studying this?

  Typical task in climate modeling for model tests, evaluation and assessment
- How does it work?

  Running the model several times with different settings, storing and analyzing results

  Optionally in parallel

  Best way: automatized, with scripts
- What if we can use it?

  Parameter studies

  Sensitivity analysis

  Uncertainty analysis

Contents

Climate models are parametrized

- Mathematical formulation – semi-discrete version (space discretized, time not):
- **Initial value problem (IVP)** for a system of **ordinary differential equations (ODEs)**:

$$\begin{aligned} \dot{y}(t) &= f(y(t), p, t), \quad t \geq t_0, \\ y(t_0) &= y_0. \end{aligned}$$

  for the unknown function $y : t \mapsto y(t)$,
- ... where we now explicitly mention model parameters $p \in \mathbb{R}^m$.
- Moreover, we have the initial values $y_0 \in \mathbb{R}^n$.
- The solution y depends on both: model parameters $p \in \mathbb{R}^m$ and initial values $y_0 \in \mathbb{R}^n$.
- It is an important task to study this dependency qualitatively and quantitatively.

Example: Zero-dimensional Energy Balance Model (EBM)

- Only variable: (global mean) temperature $y = y(t)$ as function of time:

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

with

$$c_1 = \frac{1}{4C}, c_2 = \frac{\sigma\epsilon}{C}$$

and

- the thermal coupling constant $C = 9.96 \times 10^6$,
- the emissivity $\epsilon = 0.62$,
- the Boltzmann constant $\sigma = 5.67 \times 10^{-8}$ (natural constant, makes no sense to vary),
- the solar constant $S = 1367$,
- and the albedo $\alpha = 0.3$,

$\rightsquigarrow$ $p = (C, \epsilon, S, \alpha)$,

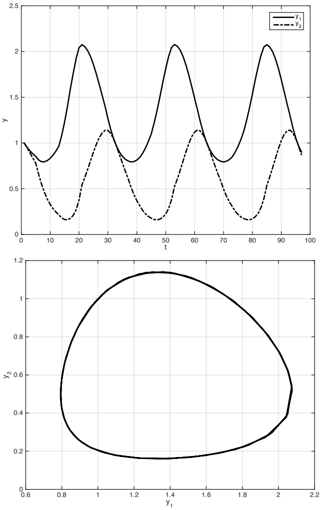
- .... and the given initial value y_0.

Example: Predator-prey model

- ODE system:

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

- Parameters: $p = (\alpha, \beta, \gamma, \delta, \lambda, \mu)$.
- Initial values $x(0) = x_0, y(0) = y_0$.
- Spatially distributed version (1-D), additional parameters:
- diffusion coefficient κ,
- spatial resolution h (or $\#$ of spatial points).



Time-discrete version of an ODE

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

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

- General form of a **multi-step method**:

$$y_{k+1} = y_k + \Delta t \, \Psi(f, p, y_k, \ldots, y_{k-m+1}, t_k, \ldots, t_{k-m+1}, \Delta t).$$

  $+$ initial values.

- Both now depending on the parameters p (since model function f depends on them).
- (Nearly) all climate models have this structure.

$\rightsquigarrow$ In the fully (space+time) discrete model, there are additional numerical parameters:

- step-size Δt,
- (optionally:) parameters of the time integrators ϕ, Ψ.

Contents

Parameters in an ensemble run for a 3-D and time-dependent model?

- Constant model parameters:
  - Global physical, biological parameters
    Example (Metos3D): growth parameter in the marine ecosystem, assumed to be the same for the whole ocean.
- Forcing data:
  - time-dependent: Example: solar "constant", incoming radiation
  - space+time dependent: Example: ice cover (if not computed by the model)
    Example: ocean circulation data (for marine ecosystem), ocean surface data (for atmosphere or ocean model alone)
- Initial values:
  - Example: Fields for temperature, velocity of water and air
- Numerical parameters:
  - Examples: spatial grid-size, temporal step-size, floating point accuracy (single/double precision)
- Numerical schemes:
  - Different discretization, e.g. advection scheme.

Contents

Parameter studies or sensitivity analysis: scalar parameters

- Vary considered parameter $p \in \mathbb{R}$ in a given interval $[p_{min}, p_{max}]$.
- Typical: equidistant grid in the parameter interval:

$$p_0 := p_{min}, p_i := p_{i-1} + \Delta p, i = 1, \dots, N \quad \text{for } \Delta p := \frac{p_{max} - p_{min}}{N} \text{ and } N \in \mathbb{N}.$$

- Takes $N+1$ model runs for each parameter $\rightsquigarrow$ high effort $\rightsquigarrow$ value N is usually small.
- Other parameters fixed.
- Repeated procedure for every parameter.

$\rightsquigarrow$ Effort: $\mathcal{O}(N^n)$, where $n =$ number of parameters.

- For time-and space-dependent data not possible, too many parameters.

Parameter studies or sensitivity analysis: time-/space-dependent fields

- Consider special (given or “interesting”) time-/space-dependent data fields only.
- ... or linear combination of special “typical” time-/space-dependent data fields, i.e. for a space- and -time dependent parameter

$$p(x,t) = \sum_{i=1}^{n} c_i \hat{p}_i(x,t),$$

where $\hat{p}_i$ are some given time-/space-dependent data fields,
c_i are the coefficients that are now varied.

- Effort: As above $\mathcal{O}(N)$ per coefficient.

$\rightsquigarrow$ several parameters: $\mathcal{O}(N^n)$, where $n =$ number of coefficients.

Generation of multi-dimensional parameters: Latin Hypercube sampling

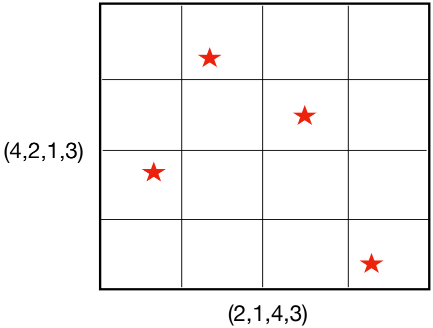
- When considering several parameters $p \in \mathbb{R}^n$ together, the effort becomes very high.

$\rightsquigarrow$ How to distribute m points in an n-dimensional parameter space?

$\rightsquigarrow$ **Latin Hypercube sampling**:

- Split the interval in each dimension into m equidistant subintervals.
- For every dimension $j = 1, \dots, n$, define a permutation of the subintervals:

$$\Pi_j(1, \dots, m) := (\pi_{1j}, \dots, \pi_{mj}) \, .$$



- Latin Hypercube points $p_i = (p_{ij})_{j=1}^n \in [0,1]^n$ are defined as

$$p_{ij} = \frac{\pi_{ij} - 1 + s_{ij}}{m}, s_{ij} \in [0,1], \quad i = 1, \dots m, j = 1, \dots, n.$$

Contents

Technical realization

- Typical situation: model is written for a single parameter run.
- Main question: Recompilation necessary for parameter change?
- Usually, parameters are defined in configuration/input file $\rightsquigarrow$ recompilation not necessary.
- Distributed parameters are read from file(s).
- Model code shall not be changed.
- Algorithm:
  - Define parameter values $p_i \in [p_{min}, p_{max}]$ (as above) to be studied.
  - For each of them:
    - Write/change configuration file
    - Start model run
    - Save output (might be overwritten $\rightsquigarrow$ different output files for every parameter).
  - Evaluate output.

$\rightsquigarrow$ Need loop around the model run.

- How to realize this?
- Shell scripts, script languages (as python).

Using scripts to perform ensemble model runs

- Script languages can be used to
  - Define the parameter values $p_i \in [p_{min}, p_{max}]$ in a vector/list.
  - For each of them:
    - Write/change configuration text file where parameters and mayvbe output file names are set.
    - Execute shell command to run the model
    - Eventually: copy output file (to be identified later).
  - Evaluate results.
- For some numerical parameters (spatial grid-size) and different numerical schemes $\leadsto$ recompilation necessary:
  - $\leadsto$ Script has to include compilation process ...
  - ... but in this case, usually not that many different runs are necessary.
  - $\leadsto$ Changes can be done by hand.

Example: Manipulating text files with python

- Modify configuration text file for a model run with given parameter and output file.
- Have to find the corresponding line in the file and modify it.
- Example: read lines of a file:

```
f = open('example.nml','r')
inlines = f.readlines()
for line in inlines:
    ...
```

- Replace line that defines parameter `alpha`:

```
if (line[0:5] == 'alpha'):
    line = 'alpha = ' + ...    # new parameter value
    outlines.append(line)
```

- Perform system command:

```
import os
os.system('run_model.exe')
```

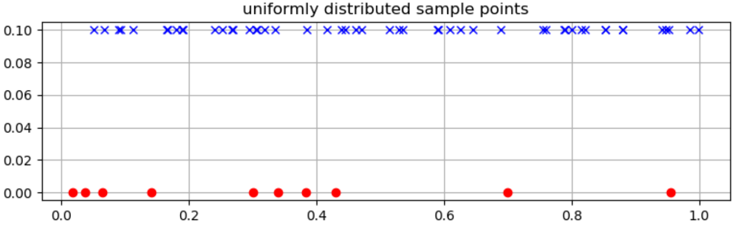
Contents

Uncertainty analysis: Ensembles with given probability distribution

- An alternative to an equidistant grid in the parameter interval (as above) is to generate parameters that are randomly chosen in the interval:

$$p_i := \text{uniform}(p_{min}, p_{max}), \quad i = 1, \ldots, N,$$

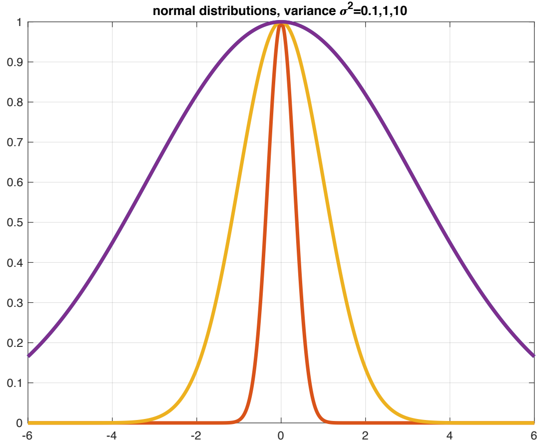
  where *uniform* denotes the uniform distribution on the given parameter interval.
- Functions that generate uniformly distributed random numbers are available in all scripting languages.



10 and 50 uniformly distributed points in $[0, 1]$.

Uncertainty analysis: Ensembles with given probability distribution

- Sometimes a certain parameter value is “typical” and we want to study small perturbations that are considered to be random.
- Then not every parameter value has the same probability, ...
- ... but small perturbations have higher probability than big ones.
- A typical way to generate such kind of parameters are using the **normal distribution**.
- They are determined by the expectation μ (to be set to the “typical” value) ...
- ... and a variance σ^2 defining the spread.

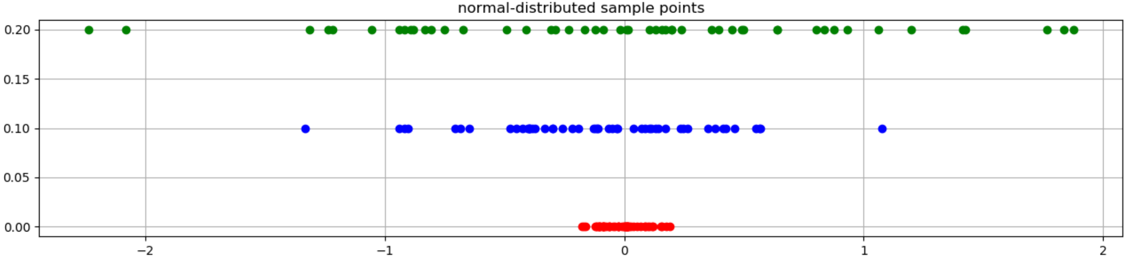


Density functions of normal distribution with $\mu = 0$ and different variances.

Normal-distributed ensembles with different variances

- Function that generate normal samples are also available in programming libraries or scripting languages (e.g., the python `scipy.stats` module):

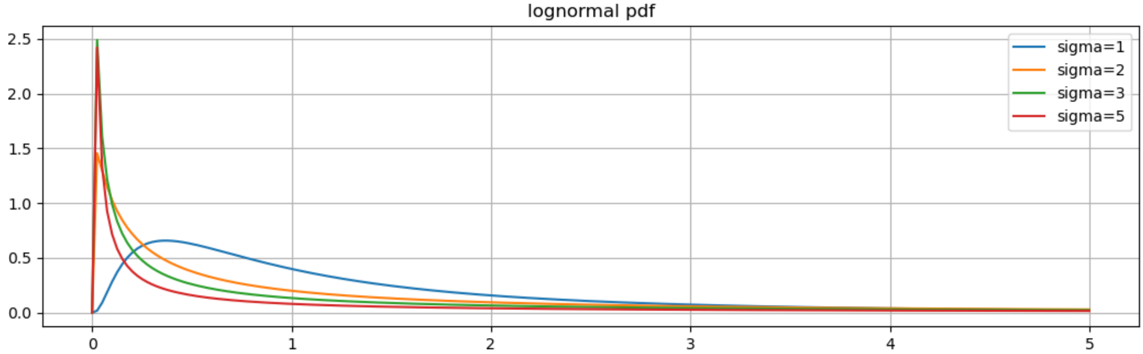
```
from scipy.stats import norm
u = norm.rvs(0,0.1,50)
plt.plot(u,np.zeros((n,1)),'ro')
```



50 normal-distributed points with $\mu = 0$ and $\sigma^2 = 0.1, 0.5, 1$.

Log-normal distribution for positive parameters

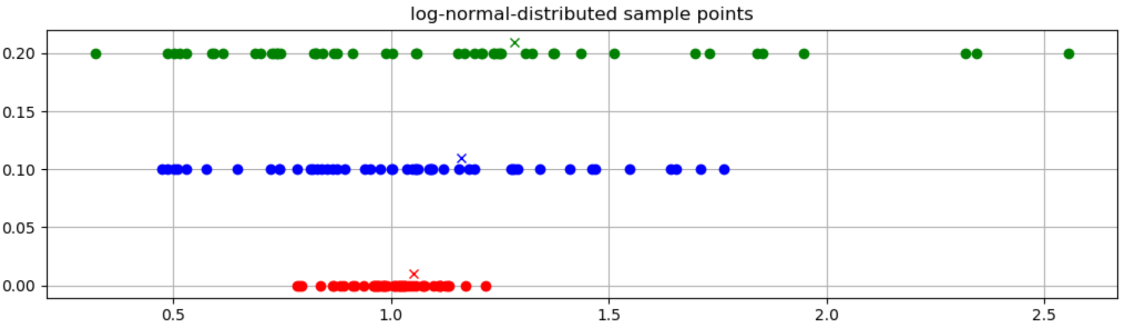
- Normal distribution has non-zero probabilities for negative values.
- If parameters are always positive: consider log-normal distribution (i.e. exponential of normal-distributed samples.



Densities of log-normal distribution with different variances of the underlying normal distribution.

Log-normal-distributed ensembles

- Function that generate log-normal samples are also available in programming libraries or scripting languages.

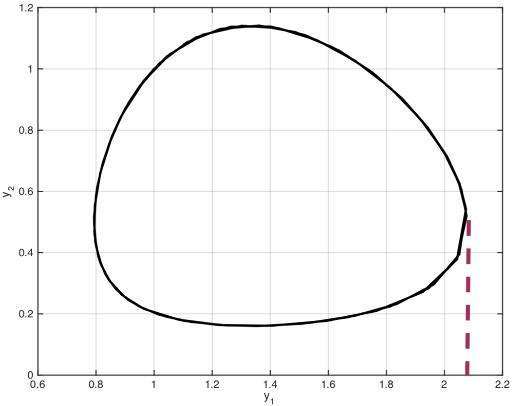


50 log-normal-distributed points with $\mu = 0$ and $\sigma = 0.1, 0.3, 0.5$.
Expectation ($\times$) of the log-normal distribution is $e^{\mu + \frac{\sigma^2}{2}}$.

Evaluation of the result of an uncertainty ensemble run

- After the ensemble run with a given parameter (= input) distribution ....
- ... we evaluate the distribution of the model results (= output).
- For example: One characteristic or interesting variable.
- Example predator-prey model (without spatial distribution):

  Maximum value of prey.
- How does this depend on the input parameter (e.g., α)?
- How big is the spread (variance)?



Evaluation of the result of an uncertainty ensemble run

- As input of the ensemble run, we have a parameter sample $(p_i)_{i=1}^N$.
- As output, we have an ensemble $(y_i)_{i=1}^N$ of the considered output variable y (assumed to be a scalar here).
- We can now estimate the expectation of the output using the mean

$$\bar{y} := \frac{1}{N} \sum_{i=1}^{N} y_i$$

- ... and the variance by computing the value

$$\frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y})^2.$$

- Using the estimator for the variance, we can now compare the input variance and the output variance to see if the model increases the uncertainty.
- This is only a first example for uncertainty analysis.

What is important

- Ensemble runs are used to study sensitivity and uncertainty of the model output w.r.t. changes in parameters of the model or the simulation.
- Parameters may be model parameters, forcing or initial data as well as also numerical parameters or even numerical schemes, for example time integrators.
- Parameters may be scalars or spatially and/or temporally distributed fields.
- For distributed fields, the problems is often reduced to coefficients of these fields. This results in considering scalar parameters again.
- We can consider equally distributed values in a parameter interval, or values generated by probability distributions.
- The evaluation of ensemble runs can be automated using scripting languages, leaving the original model unchanged.
- Results of uncertainty ensemble runs can be investigated by applying estimators for the expectation and the variance of the output values.