The RANS approach (Reynolds-averaged Navier-Stokes) is the workhorse for CFD simulations in industry and according to NASA will remain to be important in the future\(^1\). In that approach, the effect of the turbulence on the mean flow is entirely modelled by a turbulence model, e.g. \( k - \epsilon \) solves transport equations for the turbulent kinetic energy \( k \) and the turbulent dissipation \( \epsilon \). The Boussinesq approximation uses these quantities to compute the eddy viscosity \( \nu_t \) and finally replaces the Reynolds stress by \(-\nu_t \overline{u_i u_j} = 2\nu_t S_{ij} - \frac{2}{3}k \delta_{ij}\). Due to the modelling the benefits of lower costs compared to LES and DNS comes at a price: uncertainty with respect to the solution. This lack of reliability of the simulation output makes the use of CFD as a predictive tool questionable. One type of uncertainty is the epistemic uncertainty (lack of knowledge) due to the uncertain closure coefficients in the used turbulence models, for which commonly excepted values are chosen (so-called nominal values).

In this project we apply the Markov-chain Monte-Carlo (MCMC) method to i.a. the Bayesian calibration of the closure coefficients of the \( k - \epsilon \) turbulence model. The problems we consider are low-dimensional (2–4 uncertain parameters), so that the curse of dimensionality is not an issue. The project will take you through the following steps:

1. **Analytical posterior**: You will calculate with pen and paper an analytical posterior exploiting the concept of conjugacy for the likelihood and the prior.

2. **Code MCMC**: You will code the metropolis MCMC algorithm in python and apply it to a simple parameter estimation problem.

3. **Uncertain closure coefficients**: You will propagate a given prior on the closure coefficients through a CFD code and study the output. The considered fluid dynamics problem is the turbulent flow in a straight pipe at \( Re = 44,000 \) and the quantity-of-interest (qoi) is the pressure drop over the pipe’s length.

4. **Calibration of a RANS turbulence model**: You will learn to use the python PyMC-package, which is one of the most used MCMC-libraries in science and industry. You will then apply the package to the calibration of the closure coefficients of a RANS turbulence model.

### Setting up the software

This project requires the programming language python and a suitable IDE to be installed on your computer. As IDE we suggested spyder, but of course, also other IDEs are allowed to be used. This step is essential for all following tasks. If you encounter any errors or problem with the code, don’t hesitate to send me an email.

### TASK

1. Install Python 2.7 and Spyder, e.g. using anaconda:
   
   https://www.continuum.io/downloads

---

\(^1\)NASA: CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences. 2014.
1 Analytical posterior

The Bayesian inference framework requires a statistical model of the relation between the (computer) model \( m(\theta) \) which takes parameters \( \theta \) as inputs and the validation data \( d \). For example, a simple model for unbiased measurement error is

\[
d = m(\theta) + \epsilon
\]

\[
\epsilon \sim \mathcal{N}(0, \sigma^2)
\]

in which \( d \) are the observations and the error \( \epsilon \) is distributed normally around 0 with variance \( \sigma^2 \). Given this statistical model the likelihood function reads

\[
L(d|\theta) \propto \exp\left\{-\frac{(d - m(\theta))^2}{2\sigma^2}\right\}.
\]

**TASK**

1. Show that, if the model is a linear map \( m(\theta) = \theta \) and the prior follows a normal distribution (conjugate prior)

\[
p(\theta) \propto \exp\left\{-\frac{(\theta - \theta_0)^2}{2\sigma_0^2}\right\},
\]

also the posterior is normally distributed

\[
p(\theta|d) \propto \exp\left\{-\frac{(\theta - \hat{\theta})^2}{2\hat{\sigma}^2}\right\}.
\]

What are the expressions for \( \hat{\theta} \) and \( \hat{\sigma}^2 \)?

2 Code MCMC

If the computer model \( m(\theta) \) isn’t linear, we can’t find a general analytical expression for the posterior and we need to use sampling strategies such as Markov-chain Monte-Carlo (MCMC). Let’s get started with the method for a simple test case. The file `samples_from_mvnm` contains 50 samples \( d \) from a normal distribution

\[
X \sim \mathcal{N}(\mu, \Sigma)
\]

\[
\rho_X \propto \exp\left\{\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right\}
\]

with diagonal covariance matrix

\[
\Sigma = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.
\]

**TASKS**

1. Code the Metropolis algorithm in python (pseudo code given above and introduced in the lecture). Use a (bivariate) normal for the proposal distribution. Use an (uninformative) uniform prior \( p(\mu) \sim \mathcal{U}(-\infty, \infty) \). You can use the script `mcmc_bivariate.py` as a starting point.

2. Identify the mean of the data \( \mu \) starting from \( \mu = [0.2, 0.4]^T \). Let the chain run for 10,000 iterations. For the proposal distribution use a bivariate normal distribution with \( \Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \). Try \( \sigma^2 = [0.0001, 0.01, 1] \) and calculate the acceptance/rejection ratio (How many proposals get accepted/rejected?) for each proposal variance.
Algorithm 1: Metropolis algorithm

1: Initial parameter $\theta_0$
2: Evaluate $p(\theta_0|d)$, needs to be $p(\theta_0|d) > 0$
3: for $k = 1$ to $N$ do
4: Draw sample $\theta^*$ from a symmetric proposal distribution $q(\theta_{k-1}, \cdot)$ (e.g. normal distribution)
5: Compute $\alpha = \frac{p(\theta^*|d)}{p(\theta_{k-1}|d)}$
6: Draw a sample $u \sim U([0, 1])$
7: if $u < \alpha$ then
8: Accept: Set $\theta_k = \theta^*$
9: else
10: Reject: Set $\theta_k = \theta_{k-1}$
11: end if
12: end for

3. Plot the traces both for each $\mu_i$. Calculate and plot the autocorrelation function of the traces. Discuss the convergence properties of the traces.

4. Use the function `plotParamRes` to plot the probability density functions $p(\mu_i|d)$ corresponding to each trace and discuss.

3 Uncertainty of closure coefficients

In this task we will deal with the propagation of uninformed prior uncertainty on the closure coefficients of the $k-\epsilon$ turbulence model for the case of turbulent flow in a straight pipe at $Re = 44,000$. We will assume a uniform distribution on the closure coefficients on a support around nominal values. The propagation of this requires multiple flow solws, therefore the CFD simulation is replaced by a surrogate model. This surrogate model was constructed beforehand using the sparse-grid method introduced in lecture 2 and you can use it as a black-box which emulates the expensive CFD runs. The inputs to the surrogate model are the closure coefficients $\kappa, C_\epsilon, C_\mu$ and $\sigma_k$ of the turbulence model. The output is the mean pressure drop over the pipe’s length $\Delta P$.

**TASKS**

1. The steady incompressible mean flow is governed by

   $$ U_j \partial_j U_i = -\partial_j P \delta_{ij} + \partial_j (\nu \partial_j U_i - \overline{u_i u_j}), $$

   $$ \partial_j U_j = 0, $$

   in which $U_i$ and $P$ are the mean velocity and mean pressure respectively, $\overline{u_i u_j}$ is the Reynolds stress and $\nu$ is the viscosity. Assume that the flow is fully developed and derive the governing 1D equation. Discuss why the pressure drop is linear over the pipe’s length $x$. (Hint: Use the Einstein convention for the indices, write down the components of the given equations and use the fact that the flow is fully developed.)

2. The script `prior_prop_turbulentpipe.py` shows how to use the surrogate model. Run the script.

3. Define an uniform distribution for each closure coefficient $\theta \sim U[\theta_{\min}, \theta_{\max}]$ on the min/max domain defined in the dict `domain` and sample from it (use e.g. `scipy.stats.uniform.rvs`). Propagate 1000 samples through the surrogate model. Calculate the mean and use the 5th and the 95th percentile as an error bound. Modify the script `prior_prop_turbulentpipe.py` accordingly.

4. Shrink the support of the uniform distribution to ±6% of the nominal values $\theta_o$, i.e. $\theta \sim U[\theta_o - 0.06 \cdot \theta_o, \theta_o + 0.06 \cdot \theta_o]$, and propagate 1000 samples. Calculate the mean and use the 5th and the 95th percentile as an error bound.
4 MCMC: Application to turbulence modelling

In this task we will perform a Bayesian calibration of the $k - \epsilon$ turbulence model for the case of turbulent flow in a straight pipe at $Re = 44,000$ given an empirical values of the pressure drop of $\Delta P = 0.42998$. We will utilise the surrogate model introduced in the previous section in order to reduce the computational costs, since the MCMC requires multiple CFD solves.

**TASKS**

1. The script `mcmc_turbulent_pipe.py` contains an implementation of the calibration using the PyMC library. Run the script.

2. The given script applies the Bayesian inference only to one closure coefficient. Modify the script so that it runs for all four coefficients with a uniform prior defined on the domain of the surrogate model given by `domain`. Let the Markov-chain run for length 10k and 100k iterations.

3. Plot the resulting Markov-chains, their autocorrelation function and perform a kernel density estimation to visualise their distribution. Discuss the convergence behaviour of the traces. Discuss the difference between the nominal values and the posteriors.

4. Chose new nominal values of the closure coefficients given the posteriors. Argue why you’ve chosen that value. Propagate the chosen values through the surrogate model and compare with the empirical pressure drop.