Hydration Model Calibration & Optimization

Atul Agrawal

October 24, 2022

Notation:

- $\boldsymbol{x}$: concrete mix parameters (also optimization variables ultimately)
- $\boldsymbol{b}$: hydration model input parameters
- $\boldsymbol{y}_c$ or $\boldsymbol{y}_c(\boldsymbol{b})$: hydration model output(s) relevant for calibration
- $\boldsymbol{y}_o$ or $\boldsymbol{y}_o(\boldsymbol{b})$: hydration model output(s) relevant for optimization (e.g. KPIs)

Suppose N data-pairs $\mathcal{D}_N$ are available which consist of $\mathcal{D}_N = \{\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{y}}_c^{(i)}\}_{i=1}^N$. We would like to use those to infer the corresponding $\boldsymbol{b}^{(i)}$ but more importantly the relation between $\boldsymbol{x}$ and $\boldsymbol{b}$ which would be of relevance for downstream, optimization tasks.

1 Calibration

We postulate a probabilistic relation between $\boldsymbol{x}$ and $\boldsymbol{b}$ in the form of a density $p(\boldsymbol{b}|\boldsymbol{x},\ \varphi)$ where φ are associated parameters, e.g.:

$$p(\boldsymbol{b}|\boldsymbol{x},\ \varphi) = \mathcal{N}(\boldsymbol{b}|\ \boldsymbol{f}_\varphi(\boldsymbol{x}), \boldsymbol{S}_\varphi(\boldsymbol{x})) \tag{1}$$

Let also $p(\hat{\boldsymbol{y}}_c^{(i)}|\boldsymbol{y}_c(\boldsymbol{b}^{(i)}))$ the likelihood of each observation i. Then:

$$\begin{aligned} p(\boldsymbol{b}^{(1:N)}, \varphi|\mathcal{D}_N) &\propto p(\hat{\boldsymbol{y}}_c^{(1:N)}|\boldsymbol{b}^{(1:N)})\ p(\boldsymbol{b}^{(1:N)}|\hat{\boldsymbol{x}}^{(1:N)}, \varphi) \\ &= \textstyle\prod_{i=1}^N p(\hat{\boldsymbol{y}}_c^{(i)}|\boldsymbol{y}_c(\boldsymbol{b}^{(i)}))\ p(\boldsymbol{b}^{(i)}|\hat{\boldsymbol{x}}^{(i)},\ \varphi) \end{aligned} \tag{2}$$

Hence $\boldsymbol{b}^{(i)}$ (i.e. the latent model inputs for each concrete mix i) and φ would need to be inferred simultaneously. Most likely, point estimates for φ, say φ^*, would be sufficient at least as the first step.

2 Optimization (without constraints)

We define an objective function $o(\boldsymbol{y}_o)$ with respect to some output(s) of the hydration model. Since the latter depend implicitly on $\boldsymbol{x}$ (through $\boldsymbol{b}$), we wish to find the value of $\boldsymbol{x}$ that e.g. maximizes this objective. Since the relation between $\boldsymbol{x}$ and $\boldsymbol{b}$ is non-deterministic i.e. for a single $\boldsymbol{x}$ there could be multiple $\boldsymbol{b}$ and therefore multiple $\boldsymbol{y}_o(\boldsymbol{b})$, we consider the expected value of the objective, say:

$$V(\boldsymbol{x}) = \int o(\boldsymbol{y}_o(\boldsymbol{b}))\ p(\boldsymbol{b}|\boldsymbol{x}, \varphi^*)\ d\boldsymbol{b} \tag{3}$$

We denote with φ^* the point-estimates of φ found in the Calibration step above. For example if $o(\boldsymbol{y}_o)$ is the indicator function of some event relating to $\boldsymbol{y}_o$, then by maximizing $V(\boldsymbol{x})$, we maximize the probability of this event.

To carry out the optimization, derivatives of $V(\boldsymbol{x})$ would be needed. These can be expressed as:

$$\begin{aligned} \frac{\partial V}{\partial \boldsymbol{x}} &= \int o(\boldsymbol{y}_o(\boldsymbol{b}))\ \frac{\partial p(\boldsymbol{b}|\boldsymbol{x},\varphi^*)}{\partial \boldsymbol{x}}\ d\boldsymbol{b} \\ &= \int o(\boldsymbol{y}_o(\boldsymbol{b}))\ \frac{\partial\ \log p(\boldsymbol{b}|\boldsymbol{x},\varphi^*)}{\partial \boldsymbol{x}}\ p(\boldsymbol{b}|\boldsymbol{x}, \varphi^*)\ d\boldsymbol{b} \end{aligned} \tag{4}$$

Assuming the derivatives $\frac{\partial \log p(\boldsymbol{b}|\boldsymbol{x},\varphi^*)}{\partial \boldsymbol{x}}$ can be computed, the integral above can be approximated by Monte Carlo and sampling $\boldsymbol{b}^{(m)}$ from $p(\boldsymbol{b}|\boldsymbol{x}, \varphi^*)$ i.e. as:

$$\frac{\partial V}{\partial \boldsymbol{x}} \approx \frac{1}{M} \sum_{m=1}^{M} o(\boldsymbol{y}_o(\boldsymbol{b}^{(m)}))\ \frac{\partial\ \log p(\boldsymbol{b}^{(m)}|\boldsymbol{x},\varphi^*)}{\partial \boldsymbol{x}} \tag{5}$$

The Monte Carlo estimates can then be used in a stochastic gradient ascent scheme to identify the optimum $\boldsymbol{x}$. Note that **no derivatives of the hydration model** would be needed.

If we assume that the density $p(\boldsymbol{b}|\boldsymbol{x}, \varphi^*)$ is reparametrizable [1] (e.g. Gaussian) and one can write it as a differentiable transformation (ignoring φ^* in order to simplify the notation):

$$\boldsymbol{b} = \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{z}) \tag{6}$$

where $\boldsymbol{z}$ follow a known density, say $p(\boldsymbol{z})$, then with a change of variables:

$$V(\boldsymbol{x}) = \int o(\boldsymbol{y}_o(\boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{z})))\ p(\boldsymbol{z})\ d\boldsymbol{z} \tag{7}$$

and:

$$\frac{\partial V}{\partial \boldsymbol{x}} = \int \frac{\partial o}{\partial \boldsymbol{y}_o} \frac{\partial \boldsymbol{y}_o}{\partial \boldsymbol{b}} \frac{\partial \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{z})}{\partial \boldsymbol{x}}\ p(\boldsymbol{z})\ d\boldsymbol{z} \tag{8}$$

The gradient above can be estimated by sampling $\boldsymbol{z}^{(m)}$ from $p(\boldsymbol{z})$ and evaluating the other terms, i.e. as:

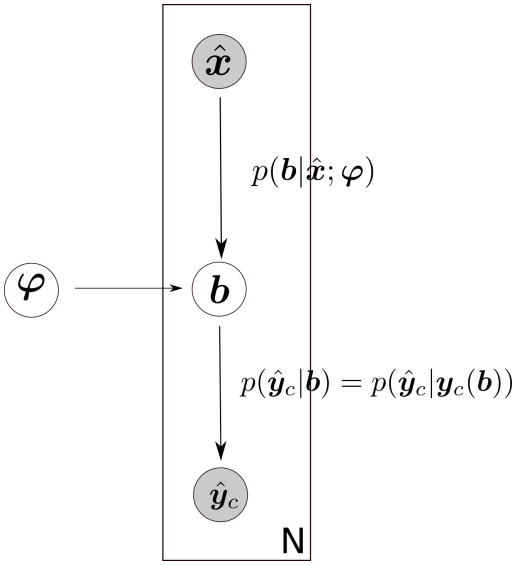
$$\frac{\partial V}{\partial \boldsymbol{x}} \approx \frac{1}{M} \sum_{m=1}^{M} \frac{\partial o^{(m)}}{\partial \boldsymbol{y}_o} \frac{\partial \boldsymbol{y}_o^{(m)}}{\partial \boldsymbol{b}} \frac{\partial \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{z}^{(m)})}{\partial \boldsymbol{x}} \tag{9}$$

where $\boldsymbol{b}^{(m)} = \boldsymbol{g}_{\boldsymbol{x}}(\boldsymbol{z}^{(m)})$, $\boldsymbol{y}_o^{(m)} = \boldsymbol{y}_o(\boldsymbol{b}^{(m)})$ and $o^{(m)} = o(\boldsymbol{y}_o^{(m)})$. While it is anticipated that the variance of these Monte Carlo estimates of the gradient of V will be smaller as compared to Equation (4), they would require a differentiable $o(.)$ (usually feasible) and a differentiable model $\boldsymbol{y}_o(\boldsymbol{b})$ (i.e. potentially adjoints).

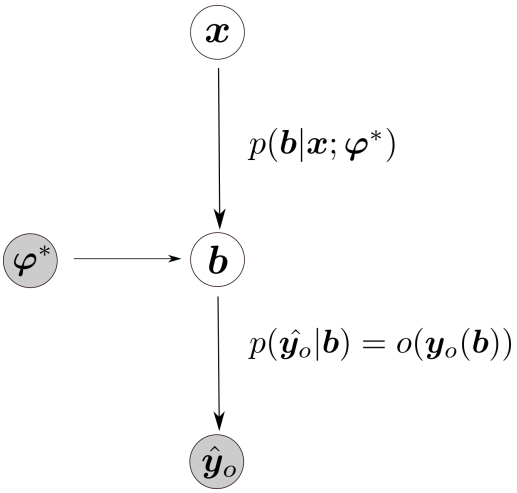
The calibration and optimisation frameworks can also be viewed as probabilistic graphs. For the current configuration, the probabilistic graph is given by Fig. 1. With slight notational abuse, same skeletal can be employed for calibration and the optimisation module. There is just a change of observed and unobserved variables. This is a generalised framework for any number of nodes and edges. In theory, if the conditionals are provided, calibration/optimisation can be performed for any level of hierarchy in the graph.

The variables and the relations associated are explained in points below:

- The variables associated are explained in the starting of the document.
- The probabilistic relation $p(\boldsymbol{b}|\boldsymbol{x};\varphi)$ between $\boldsymbol{b}$ and $\boldsymbol{x}$ can be given by a parametrised Normal distribution (example usecase in section 3.1). It can be a Gaussian process, Bayesian Neural Nets or anything similar also.
- In the example usecase in section 3.1), the probabilistic relation between the latent $\boldsymbol{b}$ and the observed $\hat{y}$ is given by a Normal distribution which needs a deterministic solver given by $\boldsymbol{y}(\boldsymbol{b})$. In the example, its just a simple linear function, it can be any black box physics based solver.
- Similarly, to elaborate the optimisation graph more, consider section 3.2. Here $\boldsymbol{x}$ is the optimisation variable with $\boldsymbol{\varphi}^*$ known from the calibration. The probabilistic relation between $\boldsymbol{b}$ and the $\hat{\boldsymbol{y}}$ is given by the objective function. It is the Mean Square error in the example ($o(\boldsymbol{y}_0(\boldsymbol{b})) = ||\hat{\boldsymbol{y}}_\mathbf{0} - \boldsymbol{y}_0(\boldsymbol{b})||_2$). Or the variable $\hat{y}$ can also represent the target domain $\mathcal{Y}$ if $o(\boldsymbol{y}_o(\boldsymbol{b}))$ is given by some indicator function $\mathbb{I}_\mathcal{Y}(\boldsymbol{y}_o)$. For example, for Hooke's law, one can have objective such as given certain strain, maximise the probability that the stress (given by $\boldsymbol{y}_o(\boldsymbol{b})$) is greater than $\hat{\boldsymbol{y}}_o$



(a) Probabilistic graph for the calibration module



(b) Probabilistic graph for the optimisation module

Figure 1: *The probabilistic graphs* : Hollow circle - latent/unobserved variables, grey circle-observed/known variables, $\longrightarrow$ - conditionals with data driven/physics based models. Note that optimisation can also be interpreted as a inference task, hence can be cast on the same skeletal.

Additional Notes on Calibration

To determine φ^*, we would need to perform the following:

$$\varphi^* = \arg\max_{\varphi} \log p(\mathcal{D}_N|\varphi)$$

Since this involves computations of deterministic values of the parameter φ, *Expectation-Maximisation* scheme can be employed. A lower bound to the log-evidence can be found out by Jensen's inequality. It says the optimal density which makes the lower bound tight is the posterior $p(\boldsymbol{b}^{(1:N)}|\mathcal{D}_N, \varphi)$. The marginal log-likelihood/evidence is given by:

$$\log p(\mathcal{D}_N|\varphi) = \sum_{i=1}^{N} \log \int p(\boldsymbol{b}^{(i)}, \mathcal{D}_N|\varphi) d\boldsymbol{b}^{(i)} \tag{10}$$

$$= \sum_{i=1}^{N} \log \int p(\mathcal{D}_N|\boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)}; \varphi) d\boldsymbol{b}^{(i)} \tag{11}$$

$$\geq \sum_{i=1}^{N} \langle \log \frac{p(\mathcal{D}_N|\boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)}; \varphi)}{q_i(\boldsymbol{b}^{(i)})} \rangle_{q_i(\boldsymbol{b}^{(i)})} = \mathcal{F}(q_{1:N}, \varphi) \tag{12}$$

- E-step: Fix φ and update $q_i(\boldsymbol{b}^{(i)})$. The optimal $q(\boldsymbol{b})$ is the conditional posterior i.e.

$$q_i^{opt}(\boldsymbol{b}^{(i)}) = p(\boldsymbol{b}^{(i)}|\mathcal{D}_N, \varphi) \propto p(\hat{\boldsymbol{y}}_c^{(i)}|\boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)}, \varphi) \tag{13}$$

  This ensures that the inequality is tight. Suppose one obtains M samples $\boldsymbol{b}_{1:M}^{(i)}$ of EACH OF THESE (conditional) posteriors using MCMC[1]. Note, this step can be parallelized.

- M-step: Given $\{q_i(\boldsymbol{b}^{(i)})\}_{i=1}^N$ compute derivatives of the ELBO w.r.t φ in order to update them. Note that:

$$\frac{\partial \mathcal{F}}{\partial \varphi} = \sum_{i=1}^{N} \left\langle \frac{\partial \log p(\boldsymbol{b}^{(i)}|\boldsymbol{x}^{(i)}; \varphi)}{\partial \varphi} \right\rangle_{q_i} \tag{14}$$

  If the latter expectation is approximated with the MCMC samples above, then:

$$\frac{\partial \mathcal{F}}{\partial \varphi} \approx \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \frac{\partial \log p(\boldsymbol{b}_m^{(i)}|\boldsymbol{x}^{(i)}; \varphi)}{\partial \varphi} \tag{15}$$

  and a stochastic gradient ascent algorithm would need to be employed. Note that one does not need to update all the $\boldsymbol{b}^{(i)}$-posteriors. It would suffice that e.g. only one is updated and the rest are kept the same (see R.Neal "A view of EM ...") i.e. new samples $\boldsymbol{b}_m^{(i)}$ are drawn from one or more q_i^{opt} while for the rest, the samples of the previous posteriors (even though φ has changed) are re-used in the estimate above. Of course this would increase the number of EM iterations that would be needed but each iteration would be cheaper. It is also possible

[1] Store the last sample from the previous E-step and use it as starting point of the MCMC in the next E-step

to use Monte Carlo to approximate the sum over i above. One could randomly select an index j, update only q_j^{opt}, i.e. draw MCMC samples from it and use a Monte Carlo estimate of the gradient as:

$$\frac{\partial \mathcal{F}}{\partial \varphi} \approx N \frac{1}{M} \sum_{m=1}^{M} \frac{\partial \log p(\boldsymbol{b}_m^{(j)} | \boldsymbol{x}^{(j)}; \varphi)}{\partial \varphi} \tag{16}$$

Of course this would increase the noise in the estimate of the derivative and probably require more EM iterations but each iteration would be cheaper.

Simplest first steps

- Generate synthetic data $\mathcal{D}_N$
  - Prescribe models:
    - $p(\boldsymbol{b}|\boldsymbol{x}, \varphi)$,
    - $\boldsymbol{y}_c(\boldsymbol{b})$
    - $p(\hat{\boldsymbol{y}}_c|\boldsymbol{y}_c(\boldsymbol{b}))$.
  - Select some value φ
  - Select $\{\boldsymbol{x}^{(i)}\}_{i=1}^N$
  - Draw $\boldsymbol{b}^{(i)} \sim p(\boldsymbol{b}^{(i)}|\boldsymbol{x}, \varphi),\ i = 1, \dots N$
  - compute $\boldsymbol{y}_c(\boldsymbol{b}^{(i)})$
  - Draw $\hat{\boldsymbol{y}}_c^{(i)} \sim p(\hat{\boldsymbol{y}}_c^{(i)}|\boldsymbol{y}_c(\boldsymbol{b}^{(i)})),\ i = 1, \dots N$
  - Then $\mathcal{D}_N = \{\boldsymbol{x}^{(i)}, \hat{\boldsymbol{y}}_c^{(i)}\}_{i=1}^N$
- Use $\mathcal{D}_N$ to infer $\boldsymbol{b}$ and find φ. Compare with the known values. As $N \to \infty$ you should recover ground truth.
- If the forward solver is prohibitively expensive, surrogate it with GP and propagate its uncertainty in the likelihood.
- With the learnt φ^*, perform the predictions/optimisation tasks downstream.
- Since there is FEM solver here, I think getting a closed form solution in E step would be difficult, so VI needs to be employed to get an approximate dist.

3 Numerical Experiment

3.1 *Calibration*

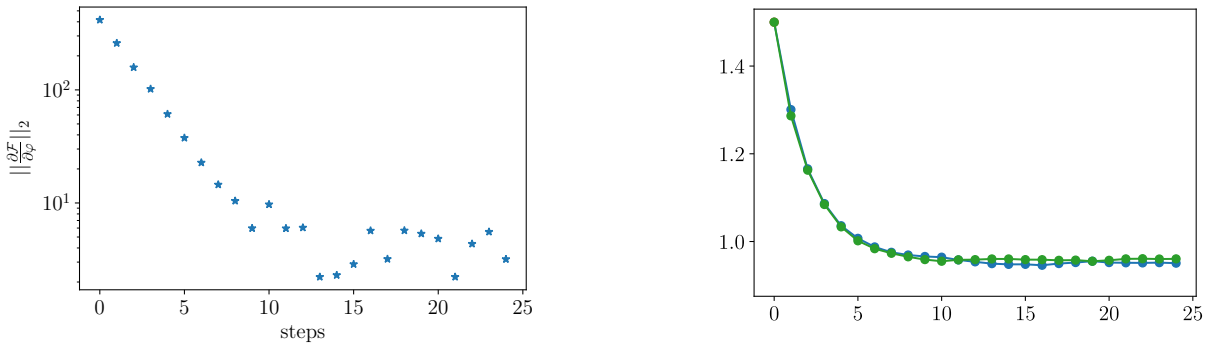
Simple linear function is chosen as a substitute to some expensive FEM model. The inputs $\boldsymbol{b}$ to the function are considered to be be latent and parameterised by φ. The models considered are as follows:

- $\boldsymbol{y}_c(\boldsymbol{b}) = \boldsymbol{A}\boldsymbol{b} + \boldsymbol{B}; \quad \boldsymbol{y}_c \in \mathbb{R}^{10}, \boldsymbol{b} \in \mathbb{R}^2 \quad$ with $\boldsymbol{A}, \boldsymbol{B}$ being coefficients.
- $p(\boldsymbol{b}|\hat{\boldsymbol{x}}; \varphi) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{\varphi}\hat{\boldsymbol{x}}, \Sigma_p) \quad \boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{\varphi} \in \mathbb{R}^2$
- $p(\hat{\boldsymbol{y}}_c|\boldsymbol{y}_c(\boldsymbol{b})) = \mathcal{N}(\hat{\boldsymbol{y}}_c|\boldsymbol{y}_c(\boldsymbol{b}), \Sigma_l)$

For a given $\boldsymbol{\varphi}$ and $\{\hat{\boldsymbol{x}}^{(i)}, \hat{\boldsymbol{y}}_c\}_{i=1}^N$, noisy synthetic data $\mathcal{D}_N$ is generated and the goal is to recover $\{q_{(i)}(\boldsymbol{b}^{(i)})\}_{i=1}^N$ and $\boldsymbol{\varphi}$ with the $E-M$ algorithm as described above.

Figure 2 discusses the results of the numerical experiment performed. The synthetic data was generated with $\boldsymbol{\varphi}$ beings $1's$. For random starting value of $\boldsymbol{\varphi}$ and $\boldsymbol{b}$, it is interesting to note that the parameters $\boldsymbol{\varphi}$ is learnt as per Fig. 2b. The ELBO gradients are given in the Fig. 2a Probabilistic estimate of the latent $\boldsymbol{b}$ are also inferred (Fig. 3). For all the observed data pairs, the inferred latents envelope the ground truth. Its worth mentioning the following points:

- More complex parametric model like a NN. GP etc can be employed for $p(\boldsymbol{b}|\boldsymbol{x}, \varphi)$.
- Currently a simple Random walk MCMC is used. If $\boldsymbol{y}_c(\boldsymbol{b})$ is computationally expensive, then either a cheaper surrogate needs to be used or the solver needs to be differentiable.
- Currently, point estimate of the parameter is learnt. If a probabilistic estimate needs to be made, variational inference needs to be performed. (EM is just a special case of VI with direc delta on the parameters)



(a) L_2 norm of the ELBO gradients

(b) Evolution of the parameters, Green and blue colors represent ϕ_1 and ϕ_2 respectively.

Figure 2: EM step diagnostics

3.2 *Optimisation*

Similar to the numerical experiment performed for the calibration task, the same simple linear model is chosen to demonstrate the capabilities of the optimisation algorithm. Here the parameters $\boldsymbol{\varphi}^*$ is

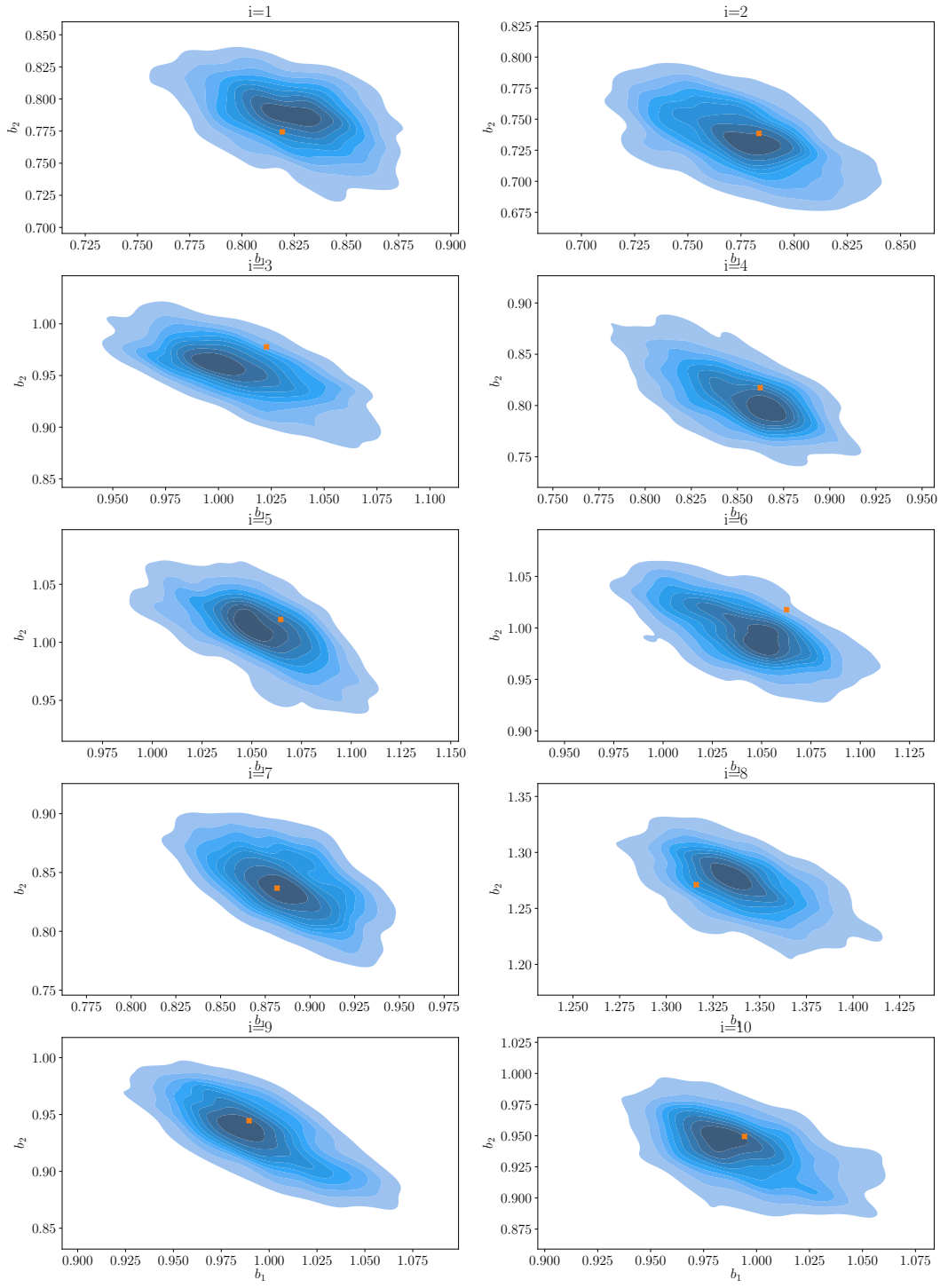
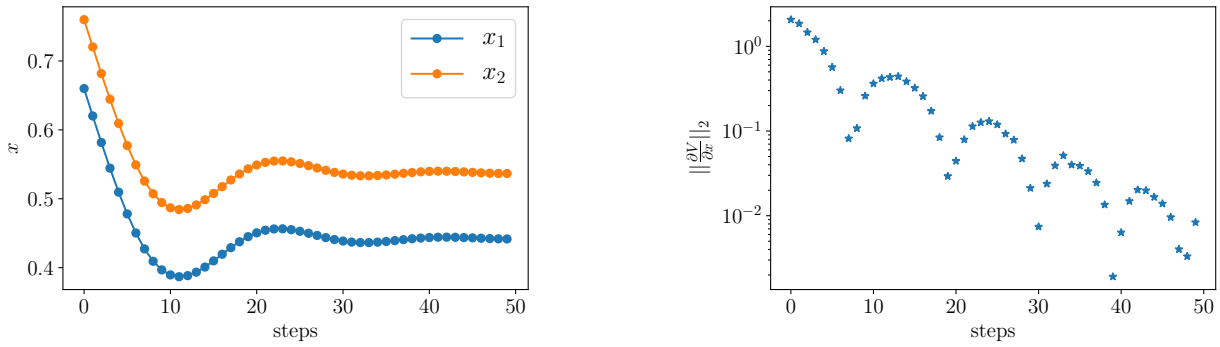


Figure 3: The latent $\boldsymbol{b}$ samples of the last EM step for N number of observed data. In orange are the "true" values for comparison.

known from calibration and the optimum $\boldsymbol{x}^*$ needs to be identified. The experiment is performed with the following configuration:

- $\boldsymbol{y}_o(\boldsymbol{b}) = \boldsymbol{A}\boldsymbol{b} + \boldsymbol{B}$; $\quad \boldsymbol{y}_c \in \mathbb{R}^{10}, \boldsymbol{b} \in \mathbb{R}^2 \quad$ with $\boldsymbol{A}, \boldsymbol{B}$ being coefficients.
- $p(\boldsymbol{b}|\boldsymbol{x}, \varphi^*) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{\varphi}^*\boldsymbol{x}, \Sigma_p) \quad \boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{\varphi}^* \in \mathbb{R}^2$
- Generating synthetic data : Take the $\boldsymbol{\varphi}^*$ value from the calibration step, fix $\boldsymbol{x}$, generate few realisations of $\boldsymbol{b}$ and then compute $\boldsymbol{y}_o(\boldsymbol{b})$ (lets call it $\hat{\boldsymbol{y}}_\mathbf{0}$)
- $o(\boldsymbol{y}_0(\boldsymbol{b})) = ||\hat{\boldsymbol{y}}_\mathbf{0} - \boldsymbol{y}_0(\boldsymbol{b})||_2$ with $\hat{\boldsymbol{y}}_\mathbf{0}$ being synthetically generated observed data.
- $\boldsymbol{x}^* = \arg\min V(\boldsymbol{x})$ (Eq. 3 - Eq. 5) is performed using ADAMS (Used hand coded optimiser as grads are approximated by Monte Carlo.)

The evolution of the design variable $\boldsymbol{x}$ is given in Fig. 4a and the Monte Carlo estimates of the gradients are given in Fig. 4b. It is important to note that the optimizer converged to an optimum value (The value was used to generate the observed synthetic data). It is noteworthy that the variance of the estimated gradients controls how the optimizer behaves near the optimum. For a tight variance near the optimum, better Monte Carlo estimates (need more samples M) are preferred. However it also amounts to a increased computational cost, which necessitates efficient surrogate to compute the objective.



(a) Evolution of the optimisation variable x

(b) Monte Carlo estimated gradients

Figure 4: Optimisation under uncertainty

3.3 Calibration and Optimisation of Concrete hydration model

A finite element based macro scale hydration model for early age concrete is performed [2]. Let the hydration model be given as a function $\boldsymbol{y} : \mathbb{R}^4 \rightarrow \mathbb{R}^D$ which maps the latent model parameter of the

[2]

hydration model given by $\boldsymbol{b}$ to the latent heat of hydration $\boldsymbol{Q}$, for D time steps. The observed data is assumed to be corrupted with a measurement noise $\boldsymbol{\varepsilon} \in \mathbb{R}^D$:

$$\hat{\boldsymbol{y}}_{\text{obs}} = \boldsymbol{y}(\boldsymbol{b}) + \boldsymbol{\varepsilon} \tag{17}$$

The model parameter $\boldsymbol{b}$ is related to the concrete mixture parameters given by $\boldsymbol{x}$. This can be assumed to be a probabilistic relation, parameterised by parameters $\boldsymbol{\varphi}$, which needs to be inferred by the EM scheme described above. Note that we cant do closed form updates in the $M-step$ as the relation between $\boldsymbol{b}$ and $\boldsymbol{x}$ is not always tractable.

The models used to perform the $E-M$ step is given by:

- $p(\boldsymbol{b}|\hat{\boldsymbol{x}};\varphi) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{W}\hat{\boldsymbol{x}} + \boldsymbol{\mathcal{B}}, \Sigma_p); \quad \boldsymbol{\varphi} = \{\boldsymbol{W}, \boldsymbol{\mathcal{B}}, \Sigma_p\}; \quad \boldsymbol{x} \in \mathbb{R}^1, \boldsymbol{W} \in \mathbb{R}^{4\times 1}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^4$
- $p(\hat{\boldsymbol{y}}_c|\boldsymbol{y}_c(\boldsymbol{b})) = \mathcal{N}(\hat{\boldsymbol{y}}_c|\boldsymbol{y}_c(\boldsymbol{b}), \Sigma_l)$

For N different experiments (each with different concrete mixture parameter), the observed data-pair can be given by $\mathcal{D} = \{\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{Q}}_i\}_{i=1}^N$. This is represented in Fig.5. It is assumed that $\boldsymbol{x}$ is $1-D$ which denotes the ratio of the two different types of cement. As per Fig. 6 and 7, we can claim that the $E-M$ step appears to be converged and hence the relation between the $\boldsymbol{b}$ and $\boldsymbol{x}$ is learnt successfully. The parametric map allows us to compute the heat of hydration $\boldsymbol{Q}$ for an unseen $\boldsymbol{x}$ (cement ratio). This is represented in Fig. 8 for the test dataset.

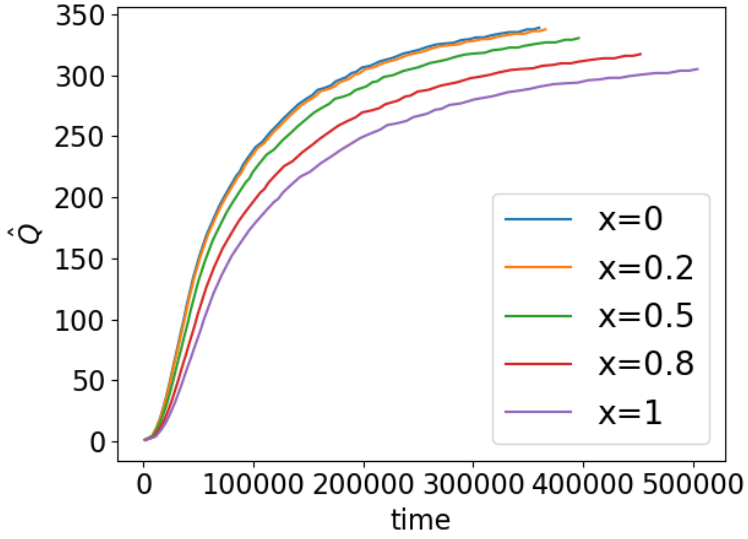


Figure 5: The available dataset $\mathcal{D} = \{\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{Q}}_i\}_{i=1}^N$ for the hydration model

3.3.1 optimisation

The task of optimization can be summarized as finding the optimum concrete mix parameters $\boldsymbol{x}$ (e.g, cement type ratio) to minimize certain (stochastic) objective such that some (stochastic) constraints are satisfied. Consider the following for the current formulation:

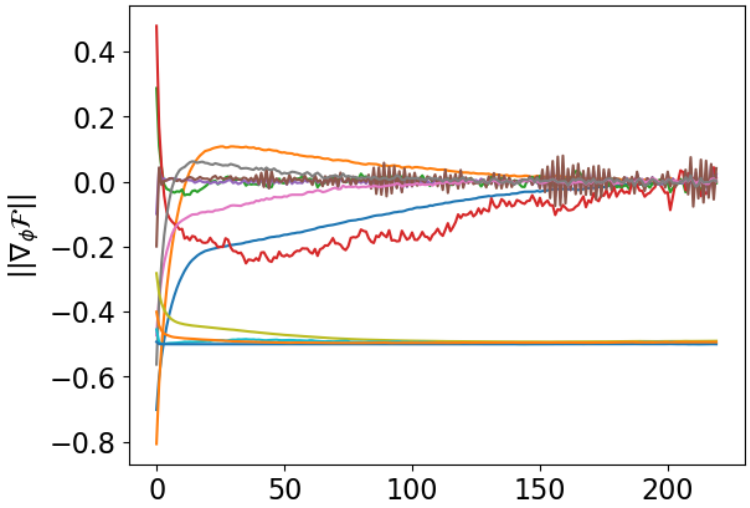


Figure 6: Evolution of the *ELBO* gradeints w.r.t individual parameters φ.

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}\sim p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi}^*)}[y_{o_1}(\boldsymbol{b})] \tag{18}$$

$$\text{s.t} \quad \mathbb{E}_{\boldsymbol{b}\sim p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi}^*)}[y_{o_i}(\boldsymbol{b})] \leq \alpha_i, \quad i = 1, \cdots, m \tag{19}$$

Here, $\boldsymbol{\varphi}^*$ denotes the learnt parameters between the latents $\boldsymbol{b}$ and $\boldsymbol{x}$, y_{o_1} denotes the solver output corresponding to the objective and y_{o_i} the solver output corresponding to the i^{th} constraint. Simplifying the notations a bit further, lets say $\mathcal{O}(\boldsymbol{x})$ denotes the objective and $\mathcal{C}_i(\boldsymbol{x})$ denotes the i^{th} constraint. The problem can be approached with penalty based methods [3] as follows:

$$\min_{\boldsymbol{x}} \mathcal{O}(\boldsymbol{x}) + \sum_{i=1}^{m} c_i \mathcal{C}_i(\boldsymbol{x}) \tag{20}$$

where $c_i > 0$ are penalty rates for violating constraints and the inequality constraint can be written as $\mathcal{C}_i(\boldsymbol{x}) = \max(\mathbb{E}_{\boldsymbol{b}\sim p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi}^*)}[y_{o_i}(\boldsymbol{b})] - \alpha_i, 0)$. Alternatively, the problem can also be approached with the Augmented Lagrangian Method A.

To test the scheme, a concrete column simulation with the hydration model is formulated [] (@Erik can add the description if need be) for which the Key performance Indicator (KPI) is the critical time (t_c) at which the max yield is reached and there is a temperature constraint which says that the max. temperature (T) at no point in time should increase a certain value. Following the notations above, we have:

[3] https://web.stanford.edu/class/ee364a/lectures/stoch$_p$rog.pdf

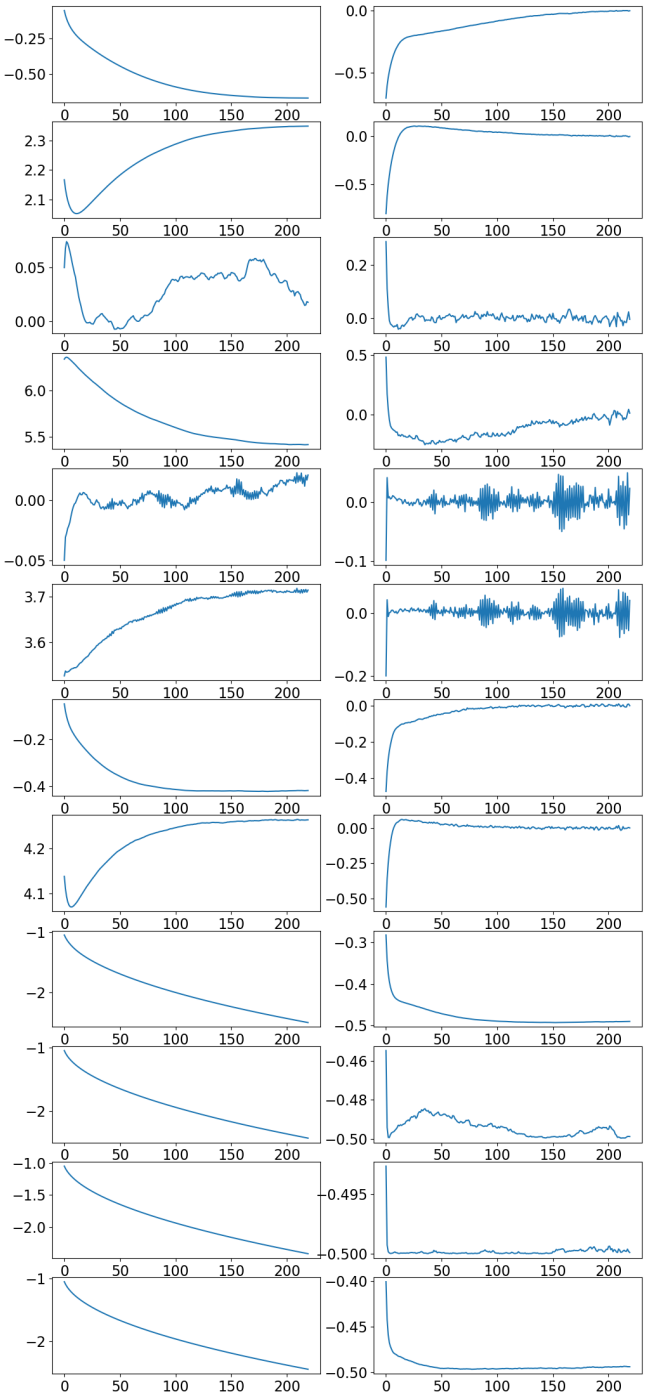


Figure 7: The evolution of parameters φ and its gradients. The left column is parameter evolution and right the gradients

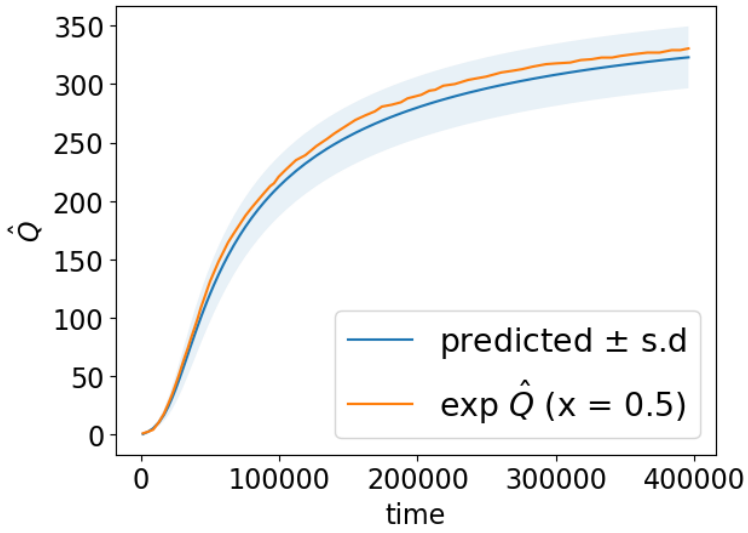


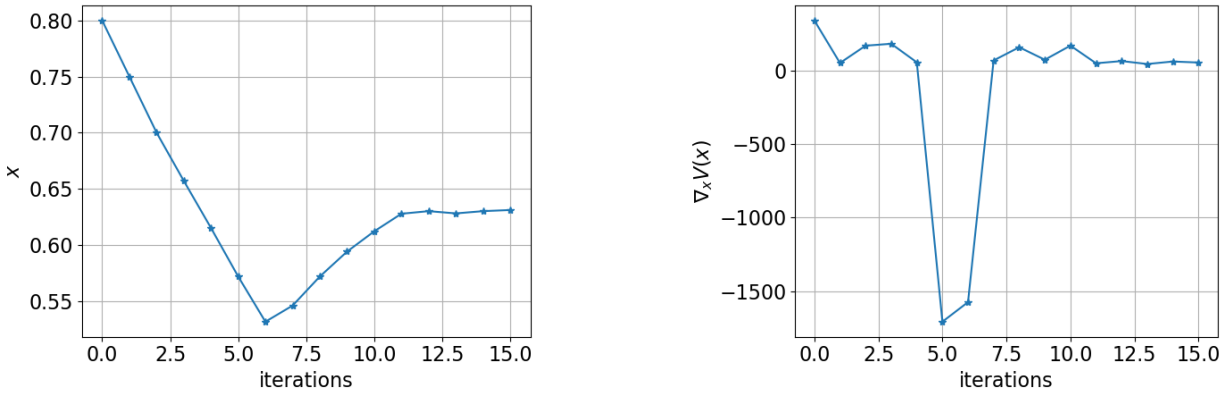
Figure 8: Evaluating the inferred parameters for the test dataset.The cem ratio $\boldsymbol{x} = 0.5$ is taken as test dataset.

$$\min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{b}\sim p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi}^*)}[t_c(\boldsymbol{b})] \tag{21}$$

$$\text{s.t} \quad \mathbb{E}_{\boldsymbol{b}\sim p(\boldsymbol{b}|\boldsymbol{x},\boldsymbol{\varphi}^*)}[T(\boldsymbol{b})] \leq \alpha \tag{22}$$

The problem is approached with penalty methods described above and score function gradient estimators (section.2) [1] are used to compute the expectation gradients. The implementation is performed in PyTorch library. Some initial results are depicted in Fig. 9 for $\alpha = 65^o$. As can be seen from the Fig. 9a, the constraints started to be violated close to $\boldsymbol{x} \sim 0.6$ which led to negative gradients, which reversed the descend of the optimization parameter (by observation $t_c \propto 1/\boldsymbol{x}$) and stabilized it to a value which also satisfies the constraints.

Its crucial to mention that computational cost of the forward solver can be a limiting factor in the optimization process. Efficient *offline* surrogates can be developed for that. It can serve two purposes: i. Alleviate the computational bottleneck, ii. Potentially reduce the estimated variance of the gradients as reparametrisation trick can be used then (surrogate is differentiable).



(a) Evolution of the optimisation variable x

(b) Monte Carlo estimated gradients

Figure 9: Optimisation under uncertainty for concrete column with hydration model

A Stochastic Optimization with inequality constraints: The Augmented Lagrangian Method

Consider the problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad \text{such that } c(\boldsymbol{x}) \leq 0 \tag{23}$$

We note that only noisy, Monte Carlo estimates of the aforementioned functions and their derivatives are available. One can define the Lagrangian:

$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda c(\boldsymbol{x}), \qquad \lambda > 0 \tag{24}$$

and perform the optimization:

$$\min_{\boldsymbol{x}} \max_{\lambda > 0} L(\boldsymbol{x}, \lambda) \tag{25}$$

One observes that if $c(\boldsymbol{x}) \leq 0$, then $\max_{\lambda>0} L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x})$ (for $\lambda = 0$) and if $c(\boldsymbol{x}) > 0$, then $\max_{\lambda>0} L(\boldsymbol{x}, \lambda) = +\infty$. The $\max_{\lambda}$ function is highly non-smooth w.r.t. $\boldsymbol{x}$ and we smooth it by adding a regularization with respect to a distance to a proximal point. In particular if λ^n is the estimate at iteration n we define the *augmented* Lagrangian (see here).

$$L^n(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda c(\boldsymbol{x}) - \frac{1}{2\rho}(\lambda - \lambda^n)^2 \tag{26}$$

This approach decomposes the constrained problem into a sequence of unconstrained problems where Lagrange multipliers are dynamically updated. We note that:

$$\frac{\partial L^n}{\partial \lambda} = 0 \rightarrow \lambda^{n+1} = max(\lambda^n + \rho c(\boldsymbol{x}),\ 0) \tag{27}$$

and:

$$\frac{\partial L^n}{\partial \boldsymbol{x}} = \nabla f(\boldsymbol{x}) + \lambda^{n+1} \nabla c(\boldsymbol{x}) \tag{28}$$

Hence we propose iterating between:

- updating $\boldsymbol{x}$ using stochastic gradient descent and Monte Carlo estimates of $\frac{\partial L^n}{\partial \boldsymbol{x}}$ above.
- updating λ as in Equation (27) where ρ is a fixed step-size (regularization term). This involves a Monte Carlo estimate of the constraint $c(\boldsymbol{x})$.

References

[1] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs, January 2016. arXiv:1506.05254 [cs].