Awesome Article Title

Atul Agrawal[1*†], Erik Tamsen[2†], Faidon-Stelios Koutsourelakis[1] and Jörg F. Unger[2]

[1*]Data-driven Materials Modeling, Technische Universität München, Boltzmannstraße 15, Garching, 85748, Germany.
[2]Modeling and Simulation, Bundesanstalt für Materialforschung und -prüfung, Unter den Eichen 87, Berlin, 12205, Germany.

*Corresponding author(s). E-mail(s): atul.agrawal@tum.de;
Contributing authors: erik.tamsen@bam.de;
p.s.koutsourelakis@tum.de; joerg.unger@bam.de;
†These authors contributed equally to this work.

Abstract

The abstract serves both as a general introduction to the topic and as a brief, non-technical summary of the main results and their implications. Authors are advised to check the author instructions for the journal they are submitting to for word limits and if structural elements like subheadings, citations, or equations are permitted.

Keywords: keyword1, Keyword2, Keyword3, Keyword4

1 Introduction

The Introduction section, of referenced text Campbell and Gear (1995) expands on the background of the work (some overlap with the Abstract is acceptable). The introduction should not include subheadings.

Springer Nature does not impose a strict layout as standard however authors are advised to check the individual requirements for the journal they are planning to submit to as there may be journal-level preferences. When preparing your text please also be aware that some stylistic choices are not supported in full text XML (publication version), including coloured font. These will not be replicated in the typeset article if it is accepted.

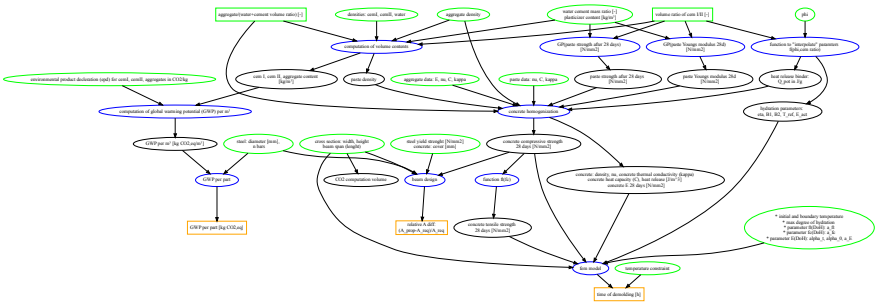

Fig. 1 Testing doit workflow. Figure is created by script. Macro for path is created by other script

2 Models

2.1 Notes on Early Age Concrete Model

Plan is do collect notes, information on the early age concrete model I am implementing. Currently the plan is to include temperature and humidity and couple them the respective mechanical fields. I will start with the temperature field.

2.2 Modeling of the temperature field

Temperature is generally described as

$$\rho C \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + \frac{\partial Q}{\partial t} \tag{1}$$

λ is the effective thermal conductivity in $\mathrm{Wm^{-1}K^{-1}}$. C is the specific heat capacity. ρ is the density. ρC is the volumetric heat capacity in $\mathrm{Jm^{-3}K^{-1}}$. Q is the volumetric heat, due to hydration, it is also called the latent heat of hydration, or the heat source in $\mathrm{Jm^{-3}}$. For now we assume the density, the thermal conductivity and the volumetric heat capacity as constant, however there are models that make them dependent on the temperature, moisture and/or the hydration.

2.2.1 Degree of hydration α

The degree of hydration α is defined as the ratio between the cumulative heat Q at time t and the total theoretical volumetric heat by complete hydration Q_∞,

$$\alpha(t) = \frac{Q(t)}{Q_\infty}, \tag{2}$$

by assuming a linear relation between the degree of hydration and the heat development. Therefore the time derivative of the heat source $\dot{Q}$ can be rewritten in terms of α,

$$\frac{\partial Q}{\partial t} = \frac{\partial \alpha}{\partial t} Q_\infty. \tag{3}$$

There are formulas to approximate total potential heat based on composition, approximated values range between 300 and 600 J/g of binder for different cement types, e.g. Ordinary Portland cement Q_∞ = 375–525 or Pozzolanic cement Q_∞ = 315–420.

2.2.2 Affinity

The heat release can be modeled based on the chemical affinity A of the binder. The hydration kinetics can be defined as a function of affinity at a reference temperature $\tilde{A}$ and a temperature dependent scale factor a

$$\dot{\alpha} = \tilde{A}(\alpha) a(T) \tag{4}$$

The reference affinity, based on the degree of hydration is approximated by

$$\tilde{A}(\alpha) = B_1 \left(\frac{B_2}{\alpha_{\max}} + \alpha \right) (\alpha_{\max} - \alpha) \exp\left(-\eta \frac{\alpha}{\alpha_{\max}} \right) \tag{5}$$

where B_1 and B_2 are coefficients depending on the binder. The scale function is given as

$$a = \exp\left(-\frac{E_\mathrm{a}}{R} \left(\frac{1}{T} - \frac{1}{T_\mathrm{ref}} \right) \right) \tag{6}$$

An example function to approximate the maximum degree of hydration based on w/c ratio, by Mills (1966)

$$\alpha_{\max} = \frac{1.031 w/c}{0.194 + w/c}, \tag{7}$$

this refers to Portland cement.

2.2.3 Time derivative

For a start I use a simple backward difference, backward Euler, implicit Euler method and approximate

$$\dot{T} = \frac{T^{n+1} - T^n}{\Delta t} \quad \text{and} \tag{8}$$

$$\dot{\alpha} = \frac{\Delta \alpha}{\Delta t} \quad \text{with} \quad \Delta \alpha = \alpha^{n+1} - \alpha^n \tag{9}$$

2.2.4 Formulation

Using (3) in (1) the heat equation is given as

$$\rho C \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + Q_\infty \frac{\partial \alpha}{\partial t} \tag{10}$$

Now we apply the time discretizations (8) and (9) and drop the index $n+1$ for readability (8)

$$\rho C T - \Delta t \nabla \cdot (\lambda \nabla T) - Q_\infty \Delta \alpha = \rho C T^n \tag{11}$$

Now, we use (9) and (4) to get a formulation for $\Delta\alpha$

$$\Delta \alpha = \Delta t \tilde{A}(\alpha) a(T) \tag{12}$$

2.2.5 Computing $\Delta\alpha$ at the Gauss-point

As $\Delta\alpha$ is not a global field, rather locally defined information.

2.2.6 Solving for $\Delta\alpha$

To solve for $\Delta\alpha$ we define the affinity in terms of α_n and $\Delta\alpha$

$$\tilde{A} = B_1 \exp\left(-\eta \frac{\Delta\alpha + \alpha_n}{\alpha_{\max}}\right) \left(\tfrac{B_2}{\alpha_{\max}} + \Delta\alpha + \alpha_n\right) (\alpha_{\max} - \Delta\alpha - \alpha_n). \tag{13}$$

Now we can solve the nonlinear function

$$f(\Delta\alpha) = \Delta\alpha - \Delta t \tilde{A}(\Delta\alpha) a(T) = 0 \tag{14}$$

using an iterative Newton-Raphson solver. For an effective algorithm we require the tangent of f with respect to $\Delta\alpha$

$$\frac{\partial f}{\partial \Delta\alpha} = 1 - \Delta t a(T) \frac{\partial \tilde{A}}{\partial \Delta\alpha} \quad \text{with} \tag{15}$$

$$\begin{aligned} \frac{\partial \tilde{A}}{\partial \Delta\alpha} = B_1 \exp\left(-\eta \frac{\Delta\alpha + \alpha_n}{\alpha_{\max}}\right) \Big[& \alpha_{\max} - \tfrac{B_2}{\alpha_{\max}} - 2\Delta\alpha - 2\alpha_n \\ & + (\tfrac{B_2}{\alpha_{\max}} + \Delta\alpha + \alpha_n)(\Delta\alpha + \alpha_n - \alpha_{\max})(\tfrac{\eta}{\alpha_{\max}}) \Big] \end{aligned} \tag{16}$$

The choice of a good starting value for the iteration seems to be critical. For some reason values close to zero can make to algorithm not converge, or to find negative values, which is non physical. When a starting values of eg. 0.2 is chosen, it seem to be stable. There is room for improvement here.

2.2.7 Macroscopic tangent

To incorporate the heat term in the this in the global temperature field, we need to compute to tangent of the term $Q_\infty \Delta\alpha$. Therefore the sensitivity of $\Delta\alpha$ with respect to the temperature T needs to be computed $\dfrac{\partial \Delta\alpha}{\partial T}$

$$\frac{\partial \Delta\alpha}{\partial T} = \Delta t \tilde{A}(\alpha) \frac{\partial a(T)}{\partial T}, \text{ with} \tag{17}$$

$$\frac{\partial a(T)}{\partial T} = a(T) \frac{E_\mathrm{a}}{RT^2} \tag{18}$$

2.3 Coupling Material Properties to Degree of Hydration

2.3.1 Compressive and tensile strength

Both compressive and tensile strength can be approximated using an generalized exponential function,

$$X(\alpha) = \alpha(t)^{a_x} X_\infty. \tag{19}$$

This model has two parameter, X_∞, the value of the parameter at full hydration, $\alpha = 1$ and a_x the exponent, which is a purely numerical parameter, difficult to estimate directly from a mix design, as the mechanisms are quite complex. The first parameter could theoretically be obtained through experiments. However the total hydration can take years, therefore usually only the value after 28 days is obtained. For now we will assume X_∞ to be a fitting parameter as well. Hopefully a functional relation of the standardized X_28 values and the ultimate value can be approximated. To write (19) in terms of the compressive strength f_c and the tensile strength f_t

$$f_\mathrm{c}(\alpha) = \alpha(t)^{a_\mathrm{c}} f_{\mathrm{c}\infty} \tag{20}$$

$$f_\mathrm{t}(\alpha) = \alpha(t)^{a_\mathrm{t}} f_{\mathrm{t}\infty} \tag{21}$$

$$\tag{22}$$

The publication assumes for their "C1" mix values of $f_{\mathrm{c}\infty} = 62.1$ MPa , $a_{fc} = 1.2$,$f_{\mathrm{t}\infty} = 4.67$ MPa , $a_{fc} = 1.0$.

2.3.2 Young's Modulus

The publication proposes a new model for the evolution of the Young's modulus. Instead of the generalized model (19), the model assumes an initial linear increase of the Young's modulus up to a degree of hydration α_t.

$$E(\alpha < \alpha_t) = E_\infty \frac{\alpha(t)}{\alpha_t} \left(\frac{\alpha_t - \alpha_0}{1 - \alpha_0} \right)^{a_E} \tag{23}$$

$$E(\alpha \geq \alpha_t) = E_\infty \left(\frac{\alpha(t) - \alpha_0}{1 - \alpha_0} \right)^{a_E} \tag{24}$$

Values of α_t are assumed to be between 0.1 and 0.2. For the mix "C1" $\alpha_t = 0.09$, $\alpha_0 = 0.06$, $E_\infty = 54.2$ MPa, $a_E = 0.4$.

2.4 Fitting of model parameters

As an initial example I will use the concrete applied in the "Cost Action TU1404".

2.4.1 Task 1 Adiabatic temperature

Vol therm al capacity 2.4 x 10^6 in J/()m3 K)
therm conductivity 1.75 w/(mK)
Initial temperature 20 degree C
Temperature data given for two initial values (temp and time/hours) Fig 2
results: activation energy 4029-5402 K**-1

2.4.2 Task 2 temperature development in a massive cube

400 mm edge cube
20 degree ambient temp
CEM I (table 4) 52.5R and other stuff...
isothermal calorimetry data 20,30,40,50,60 degree c (fig 5)
Values used by team 2 for massive cube: q pot 500 J/g
Ea/R= 5653 1/K
B1 = 0.0002916 1/s
B2 = 0.0024229
alpha max = 0.875
eta = 5.554

3 Methods (Title TBD)

Here is an empty file as example to start the calibration section. Feel free to create as many sections as necessary :D.

AA: 30 Nov, 2022: !!Below is just initial documentation. .Please dont analyze the content yet. More will be added here.!!

3.1 Model Calibration

AA: Dump from past notebook

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.

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

$$p(\boldsymbol{b} \mid \boldsymbol{x},\ \varphi) = \mathcal{N}(\boldsymbol{b} \mid\ \boldsymbol{f}_\varphi(\boldsymbol{x}), \boldsymbol{S}_\varphi(\boldsymbol{x})) \tag{25}$$

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

$$\begin{aligned} p(\boldsymbol{b}^{(1:N)}, \varphi \mid \mathcal{D}_N) &\propto p(\hat{\boldsymbol{y}}_c^{(1:N)} \mid \boldsymbol{b}^{(1:N)})\ p(\boldsymbol{b}^{(1:N)} \mid \hat{\boldsymbol{x}}^{(1:N)}, \varphi) \\ &= \textstyle\prod_{i=1}^N p(\hat{\boldsymbol{y}}_c^{(i)} \mid \boldsymbol{y}_c(\boldsymbol{b}^{(i)}))\ p(\boldsymbol{b}^{(i)} \mid \hat{\boldsymbol{x}}^{(i)},\ \varphi) \end{aligned} \tag{26}$$

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.

Additional Notes on Calibration

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

$$\varphi^* = \arg\max_{\varphi} \log p(\mathcal{D}_N \mid \varphi) \tag{27}$$

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 $p(\boldsymbol{b}^{(1:N)} \mid \mathcal{D}_N, \varphi)$. The marginal log-likelihood/evidence is given by:

$$\log p(\mathcal{D}_N \mid \varphi) = \sum_{i=1}^N \log \int p(\boldsymbol{b}^{(i)}, \mathcal{D}_N \mid \varphi) d\boldsymbol{b}^{(i)} \tag{28}$$

$$= \sum_{i=1}^N \log \int p(\mathcal{D}_N \mid \boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)} \mid \boldsymbol{x}^{(i)};\ \varphi) d\boldsymbol{b}^{(i)} \tag{29}$$

$$\geq \sum_{i=1}^N \langle \log \frac{p(\mathcal{D}_N \mid \boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)} \mid \boldsymbol{x}^{(i)};\ \varphi)}{q_i(\boldsymbol{b}^{(i)})} \rangle_{q_i(\boldsymbol{b}^{(i)})} = \mathcal{F}(q_{1:N}, \varphi) \tag{30}$$

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

$$q_i^{opt}(\boldsymbol{b}^{(i)}) = p(\boldsymbol{b}^{(i)} \mid \mathcal{D}_N, \varphi) \propto p(\hat{\boldsymbol{y}}_c^{(i)} \mid \boldsymbol{b}^{(i)}) p(\boldsymbol{b}^{(i)} \mid \boldsymbol{x}^{(i)}, \varphi) \tag{31}$$

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:

$$\frac{\partial \mathcal{F}}{\partial \varphi} = \sum_{i=1}^{N} \left\langle \frac{\partial \log p(\boldsymbol{b}^{(i)} \mid \boldsymbol{x}^{(i)};\ \varphi)}{\partial \varphi} \right\rangle_{q_i} \tag{32}$$

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

$$\frac{\partial \mathcal{F}}{\partial \varphi} \approx \sum_{i=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \frac{\partial \log p(\boldsymbol{b}_m^{(i)} \mid \boldsymbol{x}^{(i)};\ \varphi)}{\partial \varphi} \tag{33}$$

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 Neal and Hinton (1998) 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 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:

$$\frac{\partial \mathcal{F}}{\partial \varphi} \approx N \frac{1}{M} \sum_{m=1}^{M} \frac{\partial \log p(\boldsymbol{b}_m^{(j)} \mid \boldsymbol{x}^{(j)};\ \varphi)}{\partial \varphi} \tag{34}$$

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.

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

3.2 Optimization under uncertainty

3.2.1 Non-differentiable objective/constraints with design variables as argument

AA: !!under active development!! In lot of real world scenarios, complex computer simulators are used to build a relationship between parameters of the underlying theory to the experimental observations. Many a times the physics bases simulator/forward solver denoted by $\boldsymbol{y}(\cdot)$ is non-diffrentiable. For inference/optimization tasks involving these simulators, many active research areas are trying to tackle this. Cranmer et al (2020); Louppe et al (2019); Beaumont et al (2002); Marjoram et al (2003). Elaborating further, if the simulator is related to the objective of some optimization problem and design variables of the problem are direct input to the objective, then gradient based approaches are not directly applicable. The following optimization is desired:

$$\boldsymbol{x}^* = \min_{\boldsymbol{x}} \mathcal{O}(\boldsymbol{y}_o(\boldsymbol{x})) \tag{35}$$

For simplifying the explanation, constraints are omitted for now. In contrast to Eq. **??**, the design variables $\boldsymbol{x}$ are direct/explicit input to the solver. In such a setting, we advocate the use of Variational Optimization. Bird et al (2018); Staines and Barber (2012, 2013).

3.2.2 *Variational optimization*

Variational optimization are general optimization techniques that can be used to form a differentiable bound on the optima of a non-differentiable function. Given the objective $\mathcal{O}(\boldsymbol{y}(\boldsymbol{x}))$ with a simulator $\boldsymbol{y}(\cdot)$ to minimize (for simplicity lets call it $f(\boldsymbol{x})$ for now), these techniques are based on the following observation:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \leq \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\theta)}[f(\boldsymbol{x})] = U(\boldsymbol{\theta}) \tag{36}$$

where $q(\boldsymbol{x} \mid \theta)$ is a proposal distribution with parameters $\boldsymbol{\theta}$ over input values/design variables $\boldsymbol{x}$. In plain words, the minimum of a collection of values is always less than their average. Instead of minimizing f with respect to $\boldsymbol{x}$, we can minimize the upper bound U with respect to $\boldsymbol{\theta}$.

Why optimizing w.r.t $\boldsymbol{\theta}$ is same as optimizing w.r.t $\boldsymbol{x}$?
$U(\boldsymbol{\theta})$ needs to be (locally) convex ((locally) concave in case of maximization problem) in $\boldsymbol{\theta}$ and should be smooth.
Theorem 1 : If $f(\boldsymbol{x})$ is convex function and $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ an expectation affine distribution, then $U(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})}[f(\boldsymbol{x})]$ is convex in $\boldsymbol{\theta}$.(for proof check sec 1.2 in Staines and Barber (2012))
The (Multivariate) normal distribution is expectation affine, so provided a convex $f(\boldsymbol{x})$ is provided, the method should work as expected.

Under mild restrictions outlined by Staines and Barber (2012), the bound $U(\boldsymbol{\theta})$ is differential w.r.t $\boldsymbol{\theta}$, and using the log-likelihood trick its gradient can be rewritten as:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})}[f(\boldsymbol{x})] \\ &= \nabla_{\boldsymbol{\theta}} \int q(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int \nabla_{\boldsymbol{\theta}} q(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x}) d\boldsymbol{x} \\ &= \int q(\boldsymbol{x} \mid \boldsymbol{\theta}) \nabla_{\theta} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x}) d\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x})\right] \end{aligned} \tag{37}$$

The Eq.45 is the score function estimatorGlynn (1990), which also appears in the context of reinforcement learning. In the reinforcement learning context, it is classically known as the REINFORCE estimates. Williams (1992).

Effectively, this just means that if the score function $\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})$ of the proposal is known and if one can evaluate $f(\boldsymbol{x})$ for any $\boldsymbol{x}$, then one can construct approximations of Eqn. 45 which can in turn be used to minimize $U(\boldsymbol{\theta})$ with stochastic gradient descent.

For samples $x^1, \ldots, x^S$ from $q(\boldsymbol{x} \mid \boldsymbol{\theta})$ the following Monte Carlo based unbiased estimator to the upper bound gradient can be used:

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S} \sum_{i=1}^{S} f\left(x_i\right) \frac{\partial}{\partial \theta} \log q\left(x_i \mid \theta\right) \tag{38}$$

It is well known that the gradient estimator suffers from high variance which can depend on number of sample, nature of the simulator/solver etc. A common solution for this problem is to use a baseline Williams (1992) which makes use of the fact that:

$$\mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta}) f(\boldsymbol{x})\right] = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})} \left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})(f(\boldsymbol{x}) - B)\right] \tag{39}$$

for any constant B. The choice of the B does not bias the gradient estimator, but can control the variance if chosen properly.

For estimators using multiple samples as in the case presented above, we propose the use of baseline B_i for the $i-th$ term based on the other samples $j \neq i: \quad B_i = \frac{1}{S-1} \sum_{j \neq i} f(x_j)$ as discussed in Kool et al (2022). Doing this, we obtain the following for the estimator:

$$\frac{\partial U}{\partial \theta} \approx \frac{1}{S}\sum_{i=1}^{S}\frac{\partial}{\partial \theta}\log q\left(x_i \mid \theta\right)\left(f(x_i) - \frac{1}{S-1}\sum_{j\neq i} f(x_j)\right) \tag{40}$$

$$= \frac{1}{S-1}\sum_{i=1}^{S}\frac{\partial}{\partial \theta}\log q\left(x_i \mid \theta\right)\left(f(x_i) - \frac{1}{S}\sum_{j=1}^{S} f(x_j)\right) \tag{41}$$

The form in Eqn. 41 is convenient as it allows to construct a fixed baseline which is to be computed once per gradient step. Its crucial to stress on the fact that this amounts to no further computational budget as the simulators/solver would be solved for S number of times and the baseline uses the same dataset. For proof of the unbiasedness of the estimator in Eqn.41, the reader is directed to the Appendix section in Kool et al (2022).

Extending for Inequality constraints

Consider the problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad \text{such that } \mathcal{C}_i(\boldsymbol{x}) \leq 0\ , \quad \forall i \in \{1, \ldots, I\} \tag{42}$$

where $\boldsymbol{x}$ is a d dimensional vector of design variables. The design space $\mathcal{X}$ is a bounded subset of $\mathbb{R}^d$, $f : \mathcal{X} \to \mathbb{R}$ is an objective function, I is the number of inequality constraints and $\mathcal{C}_i : \mathcal{X} \to \mathbb{R}$ is the i^{th} constraint. Let the inequality constraint be defined as $\mathcal{G}(\boldsymbol{x}) = c\max(\mathcal{C}(\boldsymbol{x}), 0)$ for a given scaling term c. One can cast this constrained problem to an unconstrained one using penalty based methods giving rise to an updated objective given by:

$$\mathcal{L}(\boldsymbol{x}, c) = f(\boldsymbol{x}) + \sum_i \mathcal{G}_i(\boldsymbol{x}) \tag{43}$$

This ensures that the penalty is not applied when the constraints are met and penalty linearly increases for the amount of violation of the constraint(s). So, similar to Eq. 36, the above can upper bounded by

$$\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, c) \leq \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\theta)}[\mathcal{L}(\boldsymbol{x}, c)] = U(\boldsymbol{\theta}) \tag{44}$$

with the gradients given by

$$\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}|\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log q(\boldsymbol{x} \mid \boldsymbol{\theta})\mathcal{L}(\boldsymbol{x}, c)\right] \tag{45}$$

The gradients can be estimated by Monte Carlo as follows:

$$\nabla_{\boldsymbol{\theta}} U(\boldsymbol{\theta}) \approx \frac{1}{S}\sum_{i=1}^{S} \nabla_{\boldsymbol{\theta}} \log q(x_i \mid \boldsymbol{\theta})\mathcal{L}(x_i, c) \tag{46}$$

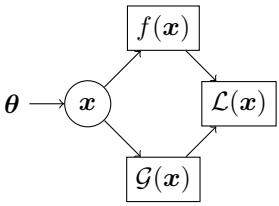

Fig. 2 *Stochastic computational graph for the constraint optimization problem:* The circle represents *stochastic nodes*, rectangle the *deterministic node* and no shape is for the *input nodes*.

To efficiently compute the gradient estimators, we propose to capitalize on the auto-differentiation capabilities of modern machine learning libraries which uses back-propagation algorithm. Since the current setup involves random variables leading to a need for gradient estimates, the standard back-propagation algorithm is modified under the hood. *Stochastic Computational Graphs* Schulman et al (2016) are used for the modification to automatically derive an unbaised estimator of the objective gradient. Furthermore, stochastic computational graphs also acts as powerful visualization method to study the interdependence between the variables. It is essentially a directed, acyclic graph, with three types of nodes: 1. Input nodes, which are set externally, including the parameters we differentiate with respect to. 2. Deterministic nodes, which are functions of their parents. 3. Stochastic nodes, which are distributed conditionally on their parents. The Stochastic Computational Graph for the objective given by Eqn. 44 is shown in Figure. 2. We propose to alleviate the dependence on the penalty parameter c by using the sequential unconstrained minimization technique (SUMT)algorithm Fiacco and McCormick (1990)[2]. Instead of using a fixed penalty parameter c as in the penalty-based method, SUMT adapts the penalty parameter at each iteration based on the current constraint violation. This adaptation of the penalty parameter helps to balance the need to satisfy the constraints with the need to make progress towards the optimal solution.

One can indeed argue that Augmented Lagrangian (AL) Fortin and Glowinski (2000) Methods can also be employed in the current setting, the choice of which optimization method to use, SUMT or AL, depends on the specific problem at hand and the trade-offs between accuracy and computational efficiency.

4 Numerical Experiments

4.1 Quadratic function

Lets consider a simple 2D quadratic function given by:

$$f(\boldsymbol{x}) = \frac{1}{2D}\left(x_1^2 + x_2^2\right) \tag{47}$$

[2] https://mat.uab.cat/ alseda/MasterOpt/const_opt.pdf

with $\boldsymbol{x} = (x_1, x_2)$. For the function, consider the problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad \text{such that } x_1 \geq 1 \tag{48}$$

Following from Eq.44, the upper bound can be given by the following for a Gaussian with the variational parameter θ:

$$U(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta})}\left[f(\boldsymbol{x}) + \max(1 - x_1, 0)\right] \tag{49}$$

with $\boldsymbol{\theta} = (\mu, \sigma)$. The gradients of the upper bound $U(\boldsymbol{\theta})$ can be approximated as discussed in Eq. 46 using Monte Carlo. With the learning rate $\eta = 0.1$, initial noise $\sigma = 5$ and ADAM optimizer to perform the optimization, we obtain the results discussed in Figure 5. Figure 3 discusses the results when the constraints are not considered and Figure 4 discusses the results with the constraints. As we can see from the figures that the noisy gradient estimates are able to drive that optimizer to the minimum. Especially in the case of Figure 4, two different starting point are studied. One which starts in a domain where the constraints are met (in red) and the next where the constraints are violated (in yellow). In both the cases, the optimizer moves towards the minimum while satisfying the constraints, thus converging close to $(x_1, x_2) = (1, 0)$. Its also noteworthy to observe that the variance σ^2 reduces as we near the minimum.

Open research question/Novelty :

1. Why not use finite differences to approximate gradients?: With the constraints, the augmented objective is C^0, so the gradients are not even defined at that point.
2. Then why not use Bayesian Optimization? AA: Bayesian optimization is difficult for ($dim \geq 10$). Also constraints are "difficult" in Bayesian optimization. The VO most probably will also struggle in high dimention. Have to check. But including constraints is not that difficult. Stelios: For the current toy problem, BO may be better (because objective has no random variable). But for the problem when the objective has implicit dependance on the design variable (through a random variable), BO makes no sense. The objective would not be known
3. To test in high dimention, Will it make sense to test the VO with the following?:

$$f(x) = \frac{1}{200}\sum_{i=1}^{100} x_i^2 \quad \text{s.t} \quad x_i \geq 1 \tag{50}$$

The $\boldsymbol{x}^*$ would be a unit vector.

4.2 Performance based concrete design

The interconnected graph in Figure. **??** can be represented in term of probabilistic graphs as discussed in Figure. 6

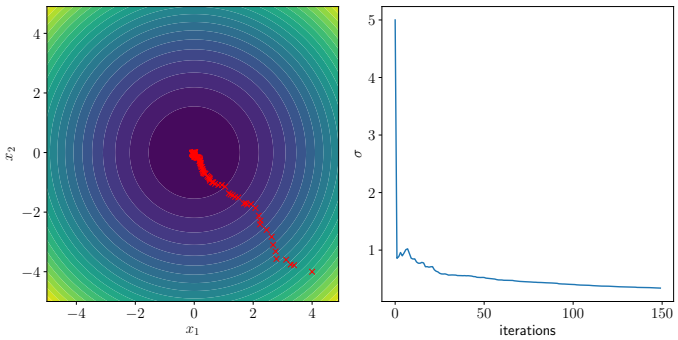

Fig. 3

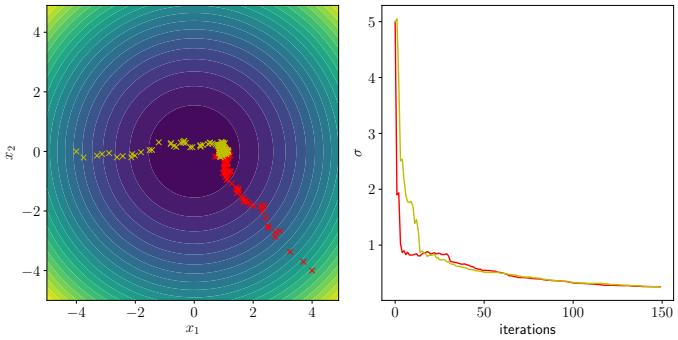

Fig. 4

Fig. 5 *Stochastic VO for constrained and unconstrained quadratic function*: (a) This is for case when constraints are not present. The left plot shows how the Gaussian mean μ move towards the minimum of objective despite noisy gradients, the right plots the learned σ values versus the gradient descent iterations (b) This is for a constraint on $\boldsymbol{x}(x_1 \geq 1)$. The left plot shows how the Gaussian mean μ moves towards the optimum (for two different starting values) while trying to satisfy the constraint and the right plots the learned σ values versus the gradient descent iterations

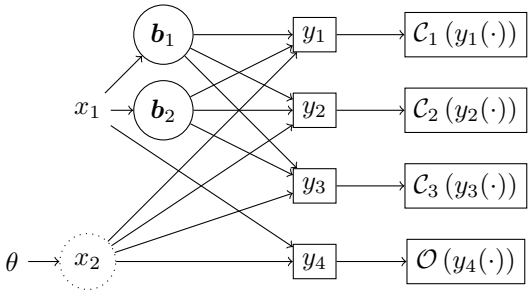

Fig. 6 *Stochastic computational graph for the constraint optimization problem for the performance based concrete design:* The circle represents *stochastic nodes*, rectangle the *deterministic node* and no shape is for the *input nodes* (design variables). The objective and the constraints are explicitly dependant on the design variable x_2 and they are not differentiable w.r.t it (Hence x_2 in dotted). So based on our discussions above, $x_2 \sim q(x_2 \mid \theta)$. Several other deterministic nodes are present between the random variables $\boldsymbol{b}_1$,$\boldsymbol{b}_2$ and the KPIs y_1, y_2, y_3, y_4 but they are ignored for brevity.

AA: This section will involve the calibration results and then the optimization results. The concrete model discussed in section 2 IMO should be in a form accessible to computational science community, with details elaborated in Appendix. Drawing from discussions in section 2, we can discuss the results here.

5 Example code from template

5.1 Tables

Tables can be inserted via the normal table and tabular environment.

Table 1 Caption text

Column 1	Column 2	Column 3	Column 4
row 1	data 1	data 2	data 3
row 2	data 4	data 5[1]	data 6
row 3	data 7	data 8	data 9[2]

Source: This is an example of table footnote. This is an example of table footnote.

[1]Example for a first table footnote. This is an example of table footnote.

[2]Example for a second table footnote. This is an example of table footnote.

Table 2 Example of a lengthy table which is set to full textwidth

	Element 1[1]			Element 2[2]		
Project	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote. This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]Example for a first table footnote.

[2]Example for a second table footnote.

In case of double column layout, tables which do not fit in single column width should be set to full text width. For this, you need to use `\begin{table*} ... \end{table*}` instead of `\begin{table} ... \end{table}` environment. Lengthy tables which do not fit in textwidth should be set as rotated table. For this, you need to use `\begin{sidewaystable} ...`

`\end{sidewaystable}` instead of `\begin{table*} ... \end{table*}` environment. This environment puts tables rotated to single column width. For tables rotated to double column width, use `\begin{sidewaystable*} ... \end{sidewaystable*}`.

5.2 Algorithms

Packages `algorithm`, `algorithmicx` and `algpseudocode` are used for setting algorithms in LaTeX using the format:

```
\begin{algorithm}
\caption{<alg-caption>}\label{<alg-label>}
\begin{algorithmic}[1]
. . .
\end{algorithmic}
\end{algorithm}
```

You may refer above listed package documentations for more details before setting `algorithm` environment. For program codes, the "program" package is required and the command to be used is `\begin{program} ... \end{program}`. A fast exponentiation procedure:

```
begin
  for i := 1 to 10 step 1 do
      expt(2, i);
      newline() od                Comments will be set flush to the right margin
where
proc expt(x, n) ≡
  z := 1;
  do if n = 0 then exit fi;
     do if odd(n) then exit fi;
        comment: This is a comment statement;
        n := n/2; x := x * x od;
     {n > 0};
     n := n − 1; z := z * x od;
  print(z).
end
```

Similarly, for `listings`, use the `listings` package. `\begin{lstlisting} ... \end{lstlisting}` is used to set environments similar to `verbatim` environment. Refer to the `lstlisting` package documentation for more details.

Table 3 Tables which are too long to fit, should be written using the "sidewaystable" environment as shown here

	Element 1[1]			Element[2]		
Projectile	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40
Element 5	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 6	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]This is an example of table footnote.

Algorithm 1 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
$y \Leftarrow 1$
if $n < 0$ **then**
    $X \Leftarrow 1/x$
    $N \Leftarrow -n$
else
    $X \Leftarrow x$
    $N \Leftarrow n$
end if
while $N \neq 0$ **do**
    if N is even **then**
        $X \Leftarrow X \times X$
        $N \Leftarrow N/2$
    else[N is odd]
        $y \Leftarrow y \times X$
        $N \Leftarrow N - 1$
    end if
end while

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
```

Supplementary information. If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

Acknowledgments. Acknowledgments are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are

submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
- Ethics approval
- Consent to participate
- Consent for publication
- Availability of data and materials
- Code availability
- Authors' contributions

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

Editorial Policies for:

Springer journals and proceedings:
https://www.springer.com/gp/editorial-policies

Nature Portfolio journals:
https://www.nature.com/nature-research/editorial-policies

Scientific Reports:
https://www.nature.com/srep/journal-policies/editorial-policies

BMC journals:
https://www.biomedcentral.com/getpublished/editorial-policies

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

Beaumont MA, Zhang W, Balding DJ (2002) Approximate bayesian computation in population genetics. Genetics 162(4):2025–2035

Bird T, Kunze J, Barber D (2018) Stochastic Variational Optimization. URL http://arxiv.org/abs/1809.04855, arXiv:1809.04855 [cs, stat]

Campbell SL, Gear CW (1995) The index of general nonlinear DAES. Numer Math 72(2):173–196

Cranmer K, Brehmer J, Louppe G (2020) The frontier of simulation-based inference. Proceedings of the National Academy of Sciences 117(48):30,055–30,062

Fiacco AV, McCormick GP (1990) Nonlinear programming: sequential unconstrained minimization techniques. SIAM

Fortin M, Glowinski R (2000) Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems. Elsevier

Glynn PW (1990) Likelihood ratio gradient estimation for stochastic systems. Communications of the ACM 33(10):75–84

Kool W, Hoof Hv, Welling M (2022) Buy 4 REINFORCE Samples, Get a Baseline for Free! URL https://openreview.net/forum?id=r1lgTGL5DE

Louppe G, Hermans J, Cranmer K (2019) Adversarial Variational Optimization of Non-Differentiable Simulators. In: Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics. PMLR, pp 1438–1447, URL https://proceedings.mlr.press/v89/louppe19a.html, iSSN: 2640-3498

Marjoram P, Molitor J, Plagnol V, et al (2003) Markov chain monte carlo without likelihoods. Proceedings of the National Academy of Sciences 100(26):15,324–15,328

Neal RM, Hinton GE (1998) A view of the em algorithm that justifies incremental, sparse, and other variants. Learning in graphical models pp 355–368

Schulman J, Heess N, Weber T, et al (2016) Gradient Estimation Using Stochastic Computation Graphs. URL http://arxiv.org/abs/1506.05254, arXiv:1506.05254 [cs]

Staines J, Barber D (2012) Variational Optimization. URL http://arxiv.org/abs/1212.4507, arXiv:1212.4507 [cs, stat]

Staines J, Barber D (2013) Optimization by variational bounding. In: ESANN

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning 8(3):229–256