Supplementary Material

Model uncertainty and decision making: Predicting the Impact of COVID-19 Using the CovidSim Epidemiological Code

Wouter Edeling[1], Hamid Arabnejad[2], Robert C Sinclair[3], Diana Suleimenova[2], Krishnakumar Gopalakrishnan[3], Bartosz Bosak[4], Derek Groen[2], Imran Mahmood Qureshi Hashmi[2], Daan Crommelin[1, 5], and Peter V Coveney *[3, 6]

[1]Scientific Computing Group, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[2]Department of Computer Science, Brunel University London, London, UK
[3]Centre for Computational Science, University College London, London, UK
[4]Poznań Supercomputing and Networking Center, Poznań, Poland
[5]Korteweg-de Vries Institute for Mathematics, University of Amsterdam, Amsterdam, The Netherlands
[6]Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands

3rd November 2020

*Corresponding author; email: p.v.coveney@ucl.ac.uk

A Dimension-adaptive uncertainty propagation

The traditional forward uncertainty quantification methods present in EasyVVUQ (*e.g.* stochastic collocation and polynomial chaos), are subject to the curse of dimensionality. To illustrate the problem, consider first the standard stochastic collocation (SC) method, which creates a polynomial approximation of the code output q, as a function of the uncertain inputs $\boldsymbol{\xi} = (\xi_d, \cdots, \xi_d) \in \mathbb{R}^d$:

$$q(\boldsymbol{\xi}) \approx \tilde{q}(\boldsymbol{\xi}) = \sum_{j_1=1}^{m_1} \cdots \sum_{j_d=1}^{m_d} q(\xi_{j_1}, \cdots, \xi_{j_d})\ a_{j_1}(\xi_1) \otimes \cdots \otimes a_{j_d}(\xi_d) \tag{3}$$

Here, $\tilde{q}$ denotes the polynomial approximation of q, and $q(\xi_{j_1}, \cdots, \xi_{j_d})$ is the actual code output, evaluated at some location inside the stochastic domain of $\boldsymbol{\xi} \in \mathbb{R}^d$. Each input $\xi_i \in \boldsymbol{\xi}$ is assigned an independent probability density function $p(\xi_i)$, and the goal is to propagate these though CovidSim in order to examine the corresponding distribution of the output q. The basic building blocks for the SC method are one-dimensional quadrature and interpolation rules, which are extended to higher dimension through a tensor-product construction. In (3), $a_{j_1}(\xi_1) \otimes \cdots \otimes a_{j_d}(\xi_d)$ is the tensor product of one-dimensional Lagrange interpolation polynomials, used to interpolate the code outputs $q(\xi_{j_1}, \cdots, \xi_{j_d})$ to a (potentially) unsampled location $\boldsymbol{\xi}$. Unlike for instance the Monte Carlo method, the sample locations $(\xi_{j_1}, \cdots, \xi_{j_d})$ are not random. Instead, each ξ_{j_i} is a point drawn from a one-dimensional quadrature rule, used to approximate integrals weighted by the chosen input distribution $p(\xi_i)$. The order of the quadrature rule for the i-th input determines the number of points m_i, and due to the tensor product construction the total number of code evaluations for d inputs equals $M = m_1 \cdot m_2 \cdots m_d$, or $M = m^d$ if all inputs receive the same quadrature order (see Figure 4 for an example). The exponential increase with d, known as the curse of dimensionality, renders the SC method intractable beyond $d \sim 10$. Hence, even though our parameter analysis in the main article indicates that only roughly 6% of the inputs will be varied at some point, due to the large number of inputs this is far too much for such 'brute force' UQ methods.

Therefore, a dimension-adaptive version of the Stochastic Collocation (SC) method, based on the work of [14, 15], has been implemented in EasyVVUQ. It is reasonable to expect that the output q will not be equally sensitive to each input ξ_i. Hence, even though our input space is d-dimensional, a dimension-adaptive approach banks on the existence of a lower *effective* dimension. The basic idea is to start with a 0-th order quadrature rule for all inputs, and to adaptively rank order the inputs, keeping all ineffective inputs at a low (possible 0-th) order, while increasing the order of those that are effective, see again Figure 4 for an example in two dimensions.

The dimension-adaptive approach is explained in detail in [15], here we only provide a general outline. Let Λ be the set containing all selected quadrature-order multi indices (the gray squares of Figure 4), which is initialized as $\Lambda := \{(0,, \cdots, 0)\}$. Let the *forward neighbours* of any multi index l be defined by the set $\{l + e_i \mid 1 \leq i \leq d\}$, where e_i is the elementary basis vector in the i-th direction, *e.g.* $e_2 = (0, 1, 0, \cdots, 0)$. The forward neighbours of the set Λ are then the forward neighbours for all $l \in \Lambda$, which are not already in Λ. Similarly, the *backward neighbours* of l are

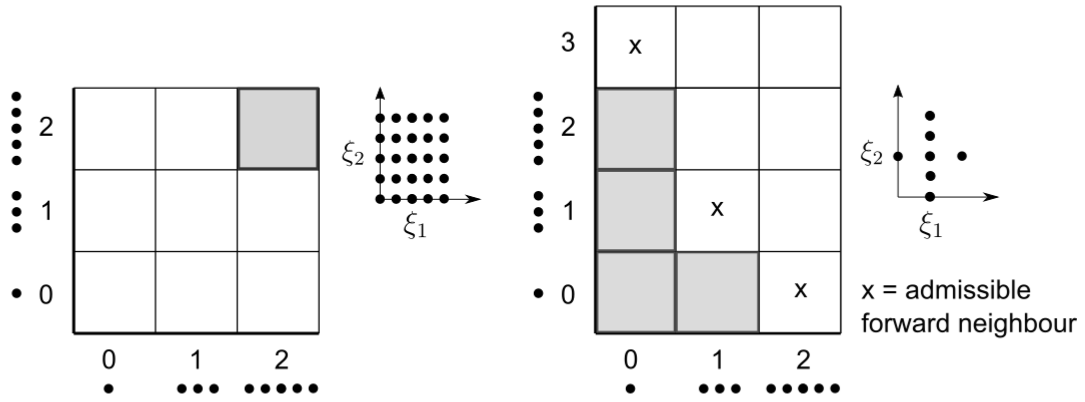


Figure 4 Two-dimensional example of building sampling plans with one-dimensional quadrature rules of different orders. Left shows a standard method, where for both inputs a 2nd-order quadrature rule is selected, leading to a dense sampling plan. Right displays a dimension-adaptive example at the 4-th iteration. The first iteration always contains the 0-th order rule for all inputs, *i.e.*, $\Lambda = \{(0,0)\}$ in this two-dimensional case. A possible sequence which results in the setup shown above could be $(0,0) \rightarrow (1,0) \rightarrow (0,1) \rightarrow (0,2)$, and the 5-th multi index to be added to Λ must be selected from one of the admissible forward neighbors. Note that $(1,2)$ is not an admissible forward neighbour, since its backward neighbour $(1,1)$ is not in Λ (the gray squares). The displayed sampling plan is built from a linear combination of tensor products, using the quadrature orders in Λ.

given by $\{l - e_i \mid l_i > 0, 1 \leq i \leq d\}$. An index set Λ is said to be *admissible* if all backward neighbours of Λ are in Λ.

To adaptively refine the sampling plan, a 'look-ahead step' [6] is executed, where the computational model is evaluated at the new unique sample locations generated by those forward neighbours l where $\Lambda \cup \{l\}$ remains an admissible set, corresponding to the $\times$ symbols of Figure 4. For each admissible forward neighbour l, a local error measure is computed. As proposed in [25], we will base our error measure on the so-called hierarchical surplus, defined as the difference between the code output q and the surrogate prediction $\tilde{q}$, evaluated at new sample locations of an admissible forward neighbour l,

$$s\left(\boldsymbol{\xi}_j^{(l)}\right) := q\left(\boldsymbol{\xi}_j^{(l)}\right) - \tilde{q}_\Lambda\left(\boldsymbol{\xi}_j^{(l)}\right), \quad \boldsymbol{\xi}_j^{(l)} \in X_l \backslash X_\Lambda. \tag{4}$$

Here, X_Λ is the sampling plan generated by the 1D quadrature rules in Λ, and X_l is the sampling plan generated by $\Lambda \cup \{l\}$. Furthermore, $\tilde{q}_\Lambda$ is the polynomial surrogate constructed from points in X_Λ alone. A local error measure can now be defined as

$$\eta^{(l)} := \frac{1}{\#(X_l \backslash X_\Lambda)} \sum_{\boldsymbol{\xi}_j^{(l)} \in X_l \backslash X_\Lambda} \left\| s\left(\boldsymbol{\xi}_j^{(l)}\right) \right\|. \tag{5}$$

Note that other error measures, based on quadrature errors [15, 30], or Sobol sensitivity indices [6] can also be defined. The admissible forward neighbour with the highest error measure $\eta^{(l)}$ is added to Λ, which can cause new forward neighbour to become admissible, and the algorithm repeats.

Note that every index $\mathbf{l} = (l_1, \cdots, l_d) \in \Lambda$ constitutes a separate tensor product of 1D quadrature rules with orders given by l. Therefore, unlike the standard approach (3), the SC expansion in the adaptive case is constructed as a linear combination of tensor products, *i.e.*,

$$q(\boldsymbol{\xi}) \approx \tilde{q}(\boldsymbol{\xi}) = \sum_{\mathbf{l}\in\Lambda} c_{\mathbf{l}} \sum_{j_1=1}^{m_{l_1}} \cdots \sum_{j_d=1}^{m_{l_d}} q(\boldsymbol{\xi}_{\mathbf{j}}^{(\mathbf{l})})\; a_{j_1}^{(l_1)}(\xi_1) \otimes \cdots \otimes a_{j_d}^{(l_d)}(\xi_d), \tag{6}$$

where $q(\boldsymbol{\xi}_{\mathbf{j}}^{(\mathbf{l})}) = q(\xi_{j_1}^{(l_1)}, \cdots, \xi_{j_d}^{(l_d)})$, and m_{l_i} is the number of points generated by a one-dimensional rule of order l_i. The coefficients c_l are computed as

$$c_l = \sum_{k_1=0}^{1} \cdots \sum_{k_d=0}^{1} (-1)^{|\mathbf{k}|_1} \cdot \chi(\mathbf{l}+\mathbf{k}), \quad \text{where} \quad \chi(\mathbf{l}) = \begin{cases} 1 & \mathbf{l} \in \Lambda_{\mathbf{l}} \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

see [14] for details.

Since Eqn.(6) consists of a linear combination of tensor products, the choice of the quadrature rule chosen to generate the one-dimensional points significantly affects the total number of code evaluations. It is common practice to select a *nested* rule, which has the property that a rule of a given order contains all points generated by that same rule at lower orders. When taking linear combinations of tensor products built from nested 1D rules of different order, may points will overlap. This leads to a more efficient sparse sampling plan, especially in higher dimensions. For our calculations we employ the well-known Clenshaw-Curtis quadrature rule; see *e.g.* [6].

A.1 Ensemble execution

Hence, through the use of adaptive methods we make the uncertainty analysis of CovidSim tractable, but our analysis nevertheless required us to perform thousands of runs, each with its own unique set of input parameters. Specifically, we used the Eagle supercomputer at Posnan Supercomputing and Network Centre [32], which has a track record of reliably supporting large ensemble calculations. The workflows associated with these UQ/SA procedures are large, multi-faceted and iterative, and to handle and curate them efficiently we rely on the FabSim3 automation toolkit [17]. FabSim3 allows us to capture commonly used workflow patterns in single-line bash commands, and it automatically captures all the relevant input parameters, output data and variables of both the job submission environment and the local machine environment in which each simulation has been executed.

A.2 Sobol index calculation

Sobol indices are variance-based sensitivity measures of a function $q(\boldsymbol{\xi})$ with respect to its inputs $\boldsymbol{\xi} \in \mathbb{R}^d$ [34, 35]. Let $\mathbb{V}[q_{\mathbf{u}}]$ be a so-called partial variance, where the multi-index $\mathbf{u}$ can be any subset of $\mathcal{U} := \{1, 2, \cdots, d\}$. Each partial variance measures the fraction of the total variance in the output q that can be attributed to the input parameter combination indexed by u. The Sobol indices are defined as the normalised partial variances, *i.e.*,

$$S_{\mathbf{u}} := \frac{\mathbb{V}[q_{\mathbf{u}}]}{\mathbb{V}[q]} \tag{8}$$

where $\mathbb{V}[q] = \sum_{u \subseteq \mathcal{U}} \mathbb{V}[q_{\mathbf{u}}]$ is the is the total variance of q [35]. Since all partial variances are positive, the sum of all possible $S_{\mathbf{u}}$ equals 1.

To perform the Sobol sensitivity analysis, we employ the method described in [20], which is an adaptation of a method originally proposed in [2]. The general idea is to transform the adaptive SC expansion into a polynomials chaos expansion (PCE), to facilitate the computation of the Sobol indices. The PCE equivalent of (3) reads

$$q(\boldsymbol{\xi}) \approx \tilde{q}(\boldsymbol{\xi}) = \sum_{\mathbf{k} \in \mathcal{K}} \eta_{\mathbf{k}} \, \phi^{(k_1)}(\xi_1) \otimes \cdots \otimes \phi^{(k_d)}(\xi_d) \tag{9}$$

Here, the basis functions $\phi_{\mathbf{k}}$ are usually constructed to be orthonormal to the input density, and the response coefficients $\eta_{\mathbf{k}}$ are normally computed via a spectral projection technique, or via a regression method. Unlike (3), summation does not take place over the collocation points $\boldsymbol{\xi}_{\mathbf{j}}$. Instead, it takes place over multi indices $\mathbf{k} = (k_1, \cdots, k_d) \in \mathcal{K}$, determined by a selected truncation scheme (see below). The PCE method is a well-know technique, and we refer to *e.g.* [36, 42] for more details.

The PCE method is particularly suited for sensitivity analysis, since the Sobol indices can be calculated from the response coefficients $\eta_{\mathbf{k}}$ in a post-processing procedure [37]. The PCE mean and variance (when the ϕ_k are orthonormal), are given by

$$\mathbb{E}[\tilde{q}] = \eta_{\mathbf{0}} \quad \text{and} \quad \mathbb{V}[\tilde{q}] = \sum_{\substack{\mathbf{k} \in \mathcal{K} \\ \mathbf{k} \neq \mathbf{0}}} \eta_{\mathbf{k}}^2. \tag{10}$$

[36]. Similarly, the partial variances can be computed with

$$\mathbb{V}[\tilde{q}_{\mathbf{u}}] = \sum_{\mathbf{k} \in \mathcal{K}_{\mathbf{u}}} \eta_{\mathbf{k}}^2 \quad \text{where} \quad \mathcal{K}_{\mathbf{u}} = \{\mathbf{k} \mid k_i > 0 \text{ when } k_i \in \mathbf{u}, \;\; j = 0 \text{ when } j \notin \mathbf{u}\}. \tag{11}$$

The multi index set $\mathcal{K}_{\mathbf{u}}$ can be interpreted as the set of all multi indices corresponding to varying only the inputs indexed by $\mathbf{u}$. That is, if for instance $\mathbf{u} = (1, 3)$, $\mathcal{K}_{\mathbf{u}}$ is the subset of $\mathcal{K}$, with all indices $\mathbf{k}$ where $k_1 > 0$ and $k_3 > 0$, with all other $k_j = 0$. Note that with (10) and (11), the Sobol indices (8) are readily available, provided we have the PCE coefficients $\eta_{\mathbf{k}}$.

To compute the PCE coefficients from our anisotropic sparse grid, we can transform the Lagrange basis to a PCE basis on the level of the one-dimensional basis functions [20]. Applying this transformation $\mathcal{T}$ to (6) yields

$$\mathcal{T}[\tilde{q}] = \sum_{\mathbf{l}\in\Lambda} c_{\mathbf{l}}\, \mathcal{T}\left[\sum_{j_1=1}^{m_{l_1}} \cdots \sum_{j_d=1}^{m_{l_d}} q(\boldsymbol{\xi}_{\mathbf{j}}^{(\mathbf{l})})\ a_{j_1}^{(l_1)}(\xi_1) \otimes \cdots \otimes a_{j_d}^{(l_d)}(\xi_d)\right], \tag{12}$$

and so we have to apply the transformation separately to each tensor product. Equating a tensor product of (12) to a corresponding PCE expansion (9) yields

$$\sum_{j_1=1}^{m_{l_1}} \cdots \sum_{j_d=1}^{m_{l_d}} q(\boldsymbol{\xi}_{\mathbf{j}}^{(\mathbf{l})})\ a_{j_1}^{(l_1)}(\xi_1) \otimes \cdots \otimes a_{j_d}^{(l_d)}(\xi_d) = \sum_{\mathbf{k}\in\Lambda_{\mathbf{l}}} \eta_{\mathbf{k}}^{(\mathbf{l})} \phi^{(k_1)}(\xi_1) \otimes \cdots \otimes \phi^{(k_d)}(\xi_d), \tag{13}$$

where the PCE truncation is $\Lambda_{\mathbf{l}} := \{\mathbf{k} \mid \mathbf{k} \leq \mathbf{l}\}$ [20]. By using the orthogonality property of the PCE basis functions (and the independence of the input distributions), we can find an expression for each coefficient $\eta_{\mathbf{k}}^{(\mathbf{l})}$ as

$$\eta_{\mathbf{k}}^{(\mathbf{l})} = \sum_{j_1=1}^{m_{l_1}} \cdots \sum_{j_d=1}^{m_{l_d}} q(\boldsymbol{\xi}_{\mathbf{j}}^{(\mathbf{l})})\ v_{k_1}^{(l_1,j_1)} \otimes \cdots \otimes v_{k_d}^{(l_d,j_d)}, \tag{14}$$

where each univariate transformation coefficient $v_{k_i}^{(l_i,j_i)}$ is given by

$$v_{k_i}^{(l_i,j_i)} = \int a_{j_i}^{(l_i)} \phi_{k_i}\ p(\xi_i)\mathrm{d}\xi, \quad i = 1\cdots, d. \tag{15}$$

This is integrated over the support of $p(\xi_i)$ using Gaussian quadrature. To generate the orthonormal ϕ_{k_i} we use the Chaospy package [9].

Once in possession of the $\eta_{\mathbf{k}}^{\mathbf{l}}$, we can compute the statistics and the Sobol indices corresponding to an adaptive sparse grid. The first two moments are given by

$$\mathbb{E}[\tilde{q}] = \sum_{\mathbf{l}\in\Lambda} c_{\mathbf{l}} \cdot \eta_{\mathbf{0}}^{(\mathbf{l})} \quad \text{and} \quad \mathbb{V}[\tilde{q}] = \sum_{\substack{\mathbf{l}\in\Lambda \\ \mathbf{l}\neq\mathbf{0}}} \left[\sum_{\mathbf{k}\in\mathcal{K}_{\mathbf{l}}} c_{\mathbf{k}} \eta_{\mathbf{l}}^{(\mathbf{k})}\right]^2 \tag{16}$$

where $\mathcal{K}_{\mathbf{l}} := \{\mathbf{k} \mid \mathbf{l} \in \Lambda_{\mathbf{k}}, \forall \mathbf{k} \in \Lambda\}$. The expression for the variance is obtained by i) inserting (6) in $\mathbb{E}\left[\tilde{q}^2\right] - \mathbb{E}[\tilde{q}]^2$; ii) grouping all terms with like $\mathbf{k}$ in $\mathbb{E}\left[\tilde{q}^2\right]$, which is what $\mathcal{K}_{\mathbf{l}}$ indicates; and iii) using the orthogonality of the $\phi_{\mathbf{k}}$ to remove all cross terms $\phi_{\mathbf{k}}\phi_{\mathbf{j}}$, $\mathbf{j} \neq \mathbf{k}$. The statistics (16) represent a more general version than those given in (10), and will revert to these equations when given the combination coefficients $c_{\mathbf{l}}$ corresponding to a standard, non-adaptive SC grid. The partial variances $\mathbb{V}[\tilde{q}_{\mathbf{u}}]$, and by extension the Sobol indices, are computed in the same way as before, namely by summing individual variance contributions indexed by the set $\mathcal{K}_{\mathbf{u}}$ shown in (11).

B Parameter distributions

Table 2 contains the 19 parameters which were included in the final UQ campaign. All were prescribed with uniform distributions with ranges displayed in Table 2, along with their default values as found in the Report 9 parameter input files [31].

The ‘Relative spatial contact rates by age power’ is not a direct input parameter to CovidSim. It is part of a parametrization for the ‘Relative spatial contact rates by age array’ input, which is defined for a number of age groups with the default values of $[0.6, 0.7, 0.75, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0.75, 0.5]$. There is a clear structure to this array, and it does not make sense to vary each entry independently form the others. Therefore, since these values lie between 0 and 1, we apply a simple power law to the default values, where ‘Relative spatial contact rates by age power’ is the exponent that we vary. This is implemented via a custom EasyVVUQ encoder, see [39] for the software.

C Uncertainty amplification factor

The aim here is to find a ‘robustness score’ of a computational model, under uncertainty in the input parameters. A simple (dimensionless) measure for variability in some random variable X is the coefficient of variation (CV), defined as the standard deviation over the mean, *i.e.*,

$$CV(X) = \frac{\sigma_X}{\mu_X}, \quad \text{if} \ \ \mu_X \neq 0. \tag{17}$$

Any forward uncertainty propagation method approximates the first two moments of the output $q \in \mathbb{R}^N$, and so $CV(q) \in \mathbb{R}^N$ is readily available. Assuming we can (analytically) compute the first two moments of each input $\xi_i \in \boldsymbol{\xi}$, $i = 1, \cdots, d$, $CV(\xi_i) \in \mathbb{R}$ is also easily computed. Although $\boldsymbol{\xi}$ may contain inputs defined on vastly different scales, since the CV is a dimensionless quantity, this will not pose a problem. We propose to use the ratio of $CV(Q)$ and $CV(\xi)$ as a relative measure of variability between the input and the output. To do so we first have to account for the fact that in general, $d \neq N$. Here we choose to average over all points:

$$CVR := CV(\bar{q})/CV(\bar{\boldsymbol{\xi}}) = \left(\frac{1}{N}\sum_{n=1}^{N}\frac{\sigma_{q_n}}{\mu_{q_n}}\right) \Bigg/ \left(\frac{1}{d}\sum_{i=1}^{d}\frac{\sigma_{\xi_i}}{\mu_{\xi_i}}\right). \tag{18}$$

The basic idea of (18) is to say something about the robustness of the code to input uncertainty, given the fact that in all likelihood the choice of input distributions can be at least partly ambiguous. We have for instance prescribed an input distribution for ‘Relative household contact rate after closure’ with end points located at 20% of the default value (see Table 2). Although this was within the range suggested by expert opinion, the number of 20% is still just a user-specified choice, and it might as well have been for instance 15%. It therefore makes sense to look at the relative

Parameter name	default	min	max
Relative household contact rate after closure	1.50	1.20	1.80
Household level compliance with quarantine	0.50	0.50	0.90
Relative spatial contact rate given social distancing	0.25	0.15	0.35
Delay to start household quarantine	1.00	0.50	3.50
Length of time households are quarantined	14.00	11.50	16.50
Delay to start case isolation	1.00	0.50	3.50
Duration of case isolation	7.00	4.50	9.50
Symptomatic infectiousness relative to asymptomatic	1.50	1.00	2.00
Proportion symptomatic	0.66	0.40	0.80
Latent period	4.59	3.00	6.00
Household attack rate	0.14	0.10	0.19
Relative spatial contact rates by age power	1.00	0.25	4.00
Residual place contacts after household quarantine	0.25	0.20	0.30
Relative place contact rate given social distancing by place type2	0.75	0.60	0.90
Relative place contact rate given social distancing by place type3	0.75	0.60	0.90
Relative rate of random contacts if symptomatic	0.50	0.40	0.60
Relative level of place attendance if symptomatic1	0.25	0.20	0.30
Relative level of place attendance if symptomatic2	0.50	0.40	0.60
Relative level of place attendance if symptomatic3	0.50	0.40	0.60

Table 2 The parameters, with their default values and uncertain range, which were included in the final UQ campaign. Variables ending with a number are part of a vector with the same name.

input-to-output uncertainty. Thus, given a user-specified average input perturbation of say 20% ($CV(\bar{\xi}) = 0.2$), Eqn. (18) tell us to what extent the code (which is a nonlinear mapping from the input to the output), amplifies this assumed uncertainty. Relative damping of uncertainty is also possible, corresponding to $CVR < 1$.

D Additional results

This section contains additional results which were omitted from the main paper due to size constraints.

D.1 Parameter refinement

The dimension-adaptive method iteratively builds a sampling plan, using a linear combination of points from quadrature rules of different order, as the locations on which to evaluate CovidSim, see Section A. All parameters are initialized with quadrature order zero, and refinement is achieved by anisotropically increasing the quadrature order of (combinations of) parameters within a given iteration of the algorithm, based on a suitable error metric.

Consider Figure 5, which shows the colour-coded refinement per iteration. Specifically, each column shows the quadrature orders that were used to refine the sampling plan. The first column is fully white, as all parameters are initialised to a zero-order rule. In the second column one parameter is refined to first order, and from there on different (combinations) of parameters are refined. Clearly, during roughly the first 50 iterations, the algorithm refines many *combinations* of important parameters to a first-order quadrature rule, before the first parameter is refined to second order. That is, it focuses on interaction effects between different parameters, and in doing so it creates a relatively dense sampling plan in the hypercube spanned by the important parameters.

D.2 Additional Sobol indices

Figure 3 of the main article only showed the Sobol indices for the three most influential inputs for scenario $\mathcal{S}_1$ and $\mathcal{S}_2$. Instead, Figure 6 displays the first-order Sobol indices for all 19 input parameters and both scenarios. The results for $\mathcal{S}_1$ and $\mathcal{S}_2$ are fairly similar, as for instance the three most dominant parameters are the same. For the less influential parameters the ranking start to differ between $\mathcal{S}_1$ and $\mathcal{S}_2$. By definition, the contribution of the least influential parameters are clumped together near zero, making it poorly visible. Consider therefore Figure 7 as well, which shows a bar chart depicting their time-averaged values. For this set, ‘Relative place contact rate given social distancing by place type3’, ‘Proportion symptomatic’, ‘Delay to start household quarantine’ and

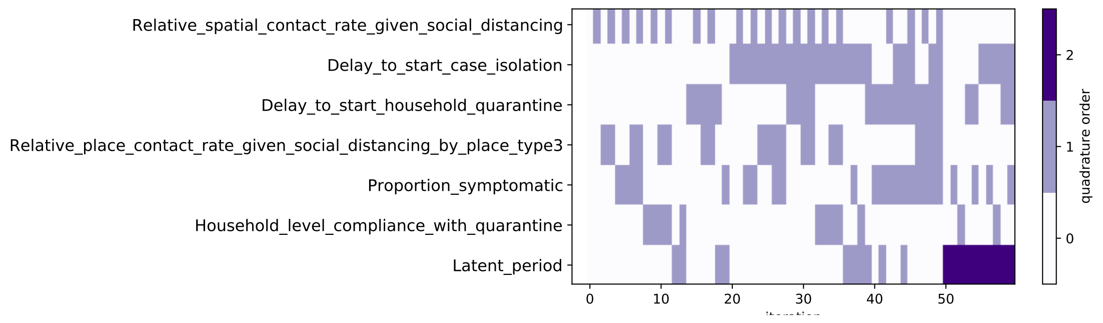
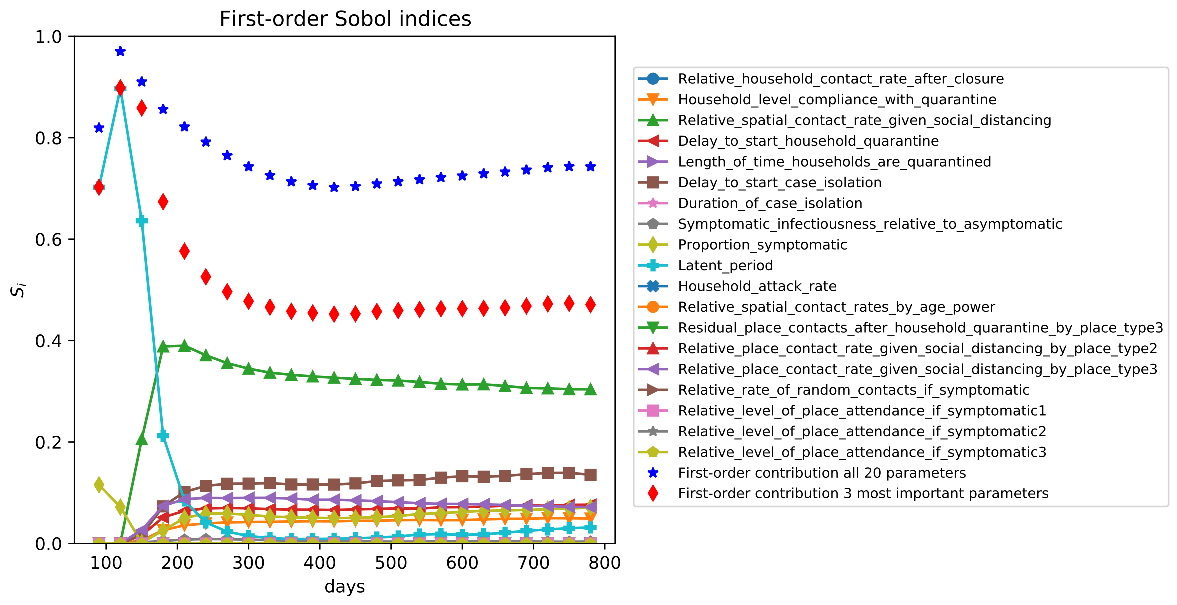
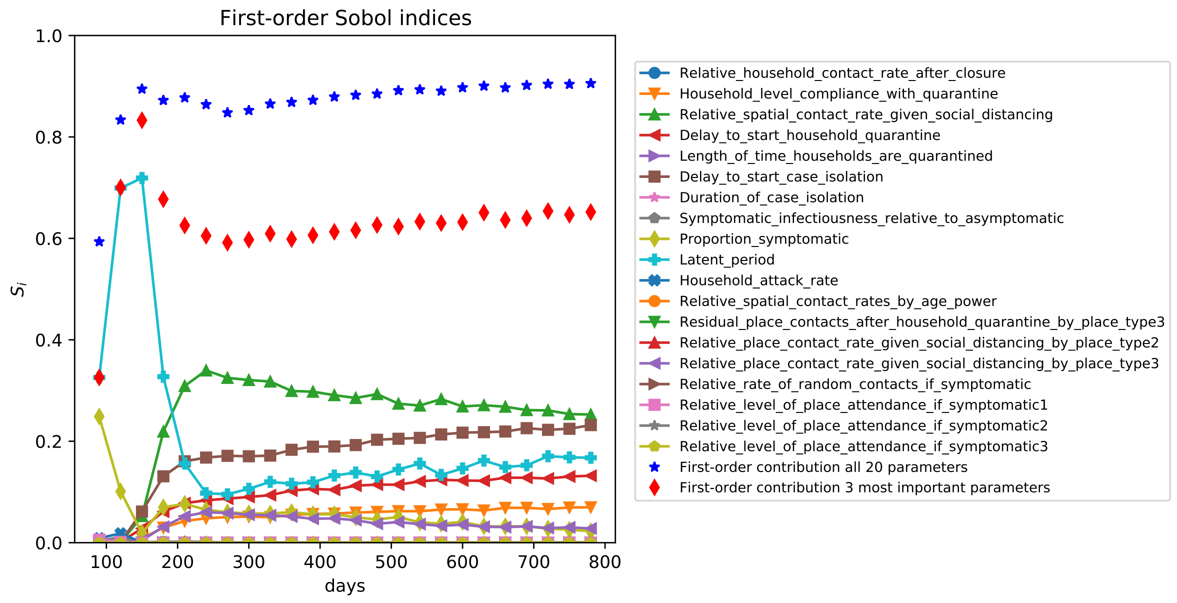


Figure 5 Colour-coded refinements per iteration of the dimension-adaptive algorithm. For the sake of clarity, not all iterations are shown. These results were obtained for $\mathcal{S}_1$.

‘Household level compliance with quarantine’ are dominant for both $\mathcal{S}_1$ and $\mathcal{S}_2$, although the order does differ.

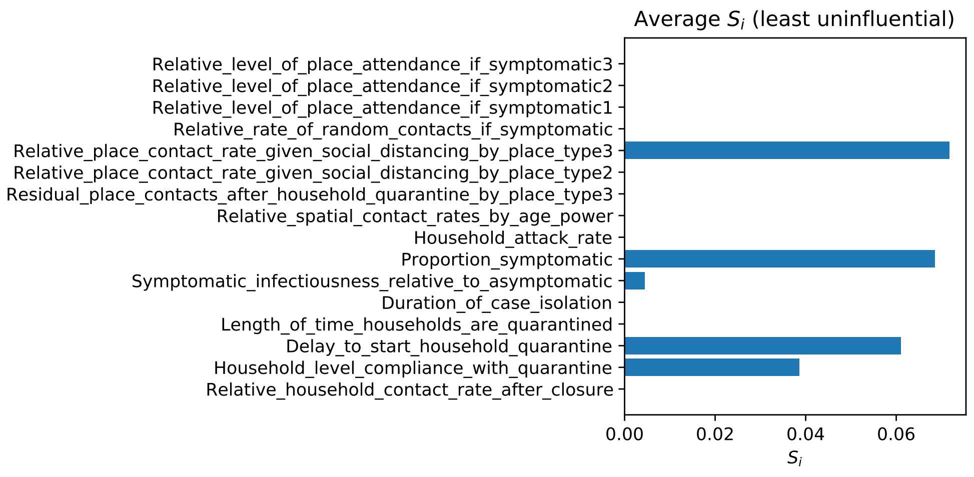


(a) Scenario $\mathcal{S}_1$: $R_0 = 2.4$, ICU on/off triggers 60/15.

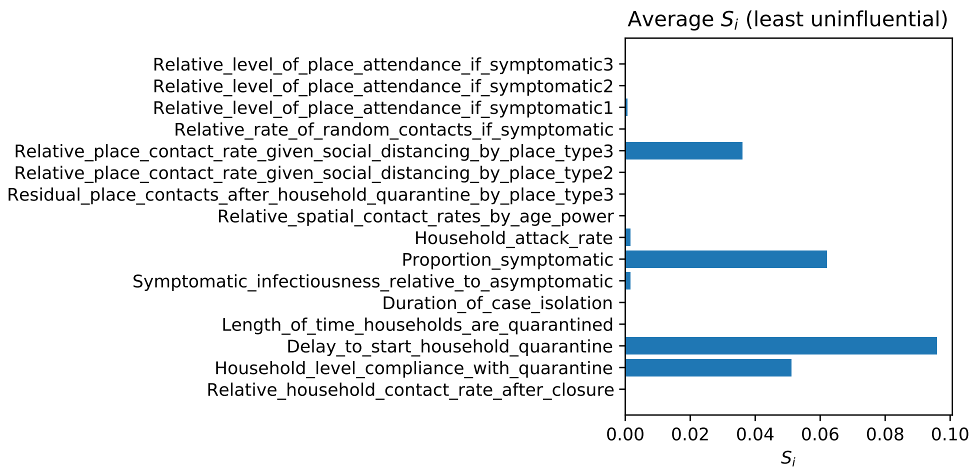


(b) Scenario $\mathcal{S}_2$: $R_0 = 2.6$, ICU on/off triggers 400/300.

Figure 6 The first-order Sobol indices for all parameters and two different scenarios S, plotted against time at one month intervals. It shows the fraction of the variance that each parameter is responsible for, over time. In addition, we show the sum of all 19 first-order indices (blue stars). The sum of the 3 most dominant parameters is also shown (red diamonds).



(a) Scenario $\mathcal{S}_1$: R_0 = 2.4, ICU on/off triggers 60/15.



(b) Scenario $\mathcal{S}_2$: R_0 = 2.6, ICU on/off triggers 400/300.

Figure 7 The time-averaged first-order Sobol indices for the 16 least influential parameters, for both scenarios. Parameters which were never refined do not contribute to the variance.