Uncertainty and sensitivity analysis of models for single panel algae photobioreactors

Martin Stefanov Stefanov

August 2013
Uncertainty and sensitivity analysis of models for single panel algae photobioreactors

Name course: Thesis project Biomass Refinery and Process Dynamics
Number: BRD-80436
Study load: 36 ects
Date: September 2011 - August 2013
Student: Martin Stefanov Stefanov
Registration number: 87-04-29-800-110
Study programme: MBT (Biotechnology)
Supervisor(s): Ir. P.M. Slegers and Dr.ir. A.J.B. Van Boxtel
Examiners: Dr.ir. L.G. Van Willigenburg
Group: Biomass Refinery and Process Dynamics
Address: Bornse Wieland 9
6708 WG Wageningen
The Netherlands
Tel: +31 (317) 48 21 24
Fax: +31 (317) 48 49 57
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE OF CONTENTS</td>
<td>3</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>5</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>7</td>
</tr>
<tr>
<td>MODEL</td>
<td>11</td>
</tr>
<tr>
<td>1. Functional form</td>
<td>11</td>
</tr>
<tr>
<td>2. Conceptual description</td>
<td>12</td>
</tr>
<tr>
<td>METHODS</td>
<td>17</td>
</tr>
<tr>
<td>1. Preparation for uncertainty and sensitivity analysis</td>
<td>17</td>
</tr>
<tr>
<td>1.1 Improvement of the simulation time</td>
<td>17</td>
</tr>
<tr>
<td>1.2 Constrained optimization</td>
<td>18</td>
</tr>
<tr>
<td>2. Uncertainty analysis</td>
<td>19</td>
</tr>
<tr>
<td>2.1 Numerical error</td>
<td>20</td>
</tr>
<tr>
<td>2.2 Input, parameter, and sub – model errors</td>
<td>21</td>
</tr>
<tr>
<td>3. Sensitivity analysis</td>
<td>21</td>
</tr>
<tr>
<td>RESULTS AND DISCUSSION</td>
<td>25</td>
</tr>
<tr>
<td>1. Preparation for uncertainty and sensitivity analysis</td>
<td>25</td>
</tr>
<tr>
<td>1.1 Improvement of the simulation time</td>
<td>25</td>
</tr>
<tr>
<td>1.2 Constrained optimization</td>
<td>28</td>
</tr>
<tr>
<td>2. Uncertainty analysis</td>
<td>30</td>
</tr>
</tbody>
</table>
Abstract

Algae biorefineries are promising facilities for sustainable production of biodiesel, ω-3 fatty acids, proteins, carbohydrates, vitamins, and pigments. The feasibility of these facilities should be determined by life cycle analysis (LCA) studies. However, their conclusiveness so far has been undermined by the lack of biomass productivity data for photobioreactors (PBRs) as a function of location, year, algae species, and design. Biomass productivity model for flat panel PBRs to address this deficiency has been developed. Yet, predictions from models deviate from reality because of uncertainty in the inputs, parameters, sub-models, and numerical schemes.

First, the speed of the original programming codes was improved. Also, a number of optimal sets of variables were determined by constrained optimization for a test case, i.e PBRs in The Netherlands, in the year 2009, made of glass, and operating with P.tricornutum. A new vectorized code in MATLAB improved the speed of the original codes by 15 times. Together with a coarse discretization mesh which improved the speed of the new code by 6.3 times, it allowed elaborate uncertainty and sensitivity analyses within reasonable time. The constrained optimization provided the variable sets which lead to the highest productivities and are of interest for uncertainty and sensitivity analyses.

Second, the predictive power of the model for single panel PBRs was validated a priori by interval uncertainty analysis for the test case. The true biomass productivities were found to be between −45 and +105 % of the model predictions for the optimal PBR designs defined by the PBR slope, surface azimuth angle, light path, and biomass concentration. The error bounds were within a factor of 3.72 from each other which is significant improvement to the typical factor of ~25 used in LCA studies.

Third, the uncertainties which have the highest impact on the accuracy were determined by one-at-a-time and a Sobol variance decomposition sensitivity analyses. The former analyses indicated the model was nonlinear in the uncertain factors. The individual errors accounted for 92% of the total error determined by interval uncertainty analysis. However, the model is linear in some errors as was confirmed by the latter analysis. The first and total order Sobol coefficients of the three most important errors were equal to their first order coefficients, i.e. the model was linear in them. Thus, 82 % of the total biomass productivity error can be accounted for by the uncertainty in the parameters; spectrally averaged absorption coefficient, maximum growth rate, and functional cross section of the photosynthetic apparatus. Furthermore, the variance of the biomass productivity is expected to be reduced by 97 % if that uncertainty is addressed.
Introduction

The world faces many energy management challenges. Much hope is put in the ability of alternative energy sources to compete economically with fossil fuels, meet the energy demand increase in the next few decades, and eventually overshadow fossil fuels. Alternatives that have achieved prominence are oil and biomass from algae [1, 2]. Algae can accumulate oil as much as 50% of their bodyweight and can grow 2 to 5 times faster than other energy crops. They do not compete for arable land, require small amount of water, and can be used as biorefinery feedstock because of their many value added components other than oil and biomass, e.g. carotenoids, ω-3 fatty acids, proteins, carbohydrates, vitamins, and pigments. In the process of biofuel and fine chemical production, algae can also utilize lower quality water, e.g. from effluent streams, and remove CO2 and NOx from flue gases.

Life cycle studies should reveal the feasibility of algae bio-refineries. However, the lack of accurate outdoor biomass productivity data undermines their validity. They are based on biomass productivities which vary by a factor of ~25 regardless of the PBR design, location, weather conditions, and algae species, see [3-7]. They use data from limited number of outdoor experimental installations, expected achievable values in the near future, theoretically maximal estimations, or a range of values [8]. How expectations and theoretical estimations would be met or how the few tested outdoor installations would perform at different locations, under variable weather conditions year to year, with different algae species is unclear.

The model of Slegers, et al, [9] for single panel algae PBRs creates an opportunity for a computational framework that can help LCA studies to identify the most promising PBR configurations and the maximal potential biomass productivities on a global scale. Such scale of analysis is necessary because at the present state of affairs an area the size of Portugal is needed to cover the fuel consumption of Europe [10]. Biomass productivity maps on the basis of light irradiation data from meteorological stations or satellite images can indicate the capabilities of existing bioreactor systems and algae species for variety of locations and years, see figure 1. Such maps can be updated continuously as the technology progresses and technical studies validate the existing models on the basis of large scale installations. Global scale maps to estimate the energy producing potential of photovoltaic technology already exist [11]. Despite their lack or rigour, such maps form a basis for the EU policy on sustainable energy.
Figure 1 A schematic representation of a sample iterative process for development and use of PBR models. The red loop reflects optimization process, whereas the blue loop the validation process. The purple elements reflect the common elements between the two. The symbols UA and SA are abbreviations of Uncertainty Analysis and Sensitivity Analysis.

However, there is a need for analysis of the predictive power of the model of Slegers, et al, [9] before it can be used for global mapping purposes. Predictions from computer simulations deviate from reality because of errors in the model inputs, parameters, sub – models, and numerical schemes. Errors arise because of uncertainty, i.e. lack of knowledge or intrinsic variability of the system and environment [12]. An uncertainty analysis can show what the accuracy is of the predicted biomass productivities. An additional sensitivity analysis can show which uncertainties lead to the highest variations in the predicted biomass productivities. These analyses are crucial 1) for assessing the confidence in the predictions as well as 2) for identifying the most important factors that need to researched and/or validated experimentally.

Moreover, knowledge of the optimal PBR design for specific location, year, algae species, and PBR material is necessary before uncertainty and sensitivity analysis can be performed. The PBR design is defined by the variables slope, surface azimuth angle, light path, and biomass concentration. Optimal variable sets were discovered in the work of Slegers, et al, [9] by graphical exploration of the variable space. The exploration was performed for the species P. tricornutum and T. pseudonana, the year 2009, and locations in The Netherlands, Algeria, and France. It indicated there are a variety of light paths and biomass concentrations which lead to the same optimal biomass productivities. Moreover, it indicated the productivity is highest for vertical reactors oriented towards East and West. Nevertheless, the variables were varied one or two at a time and at discrete intervals. Therefore, it is unclear if the discovered optimal points represent absolute maximums.
Lastly, there is a need to improve the speed of the programming code of Slegers, et al, [9] to facilitate uncertainty and sensitivity analyses. The model contains more than ten mathematical arguments and sub-models which can contain uncertainties. In addition, a single simulation with the original code in MATLAB® environment takes 15 [sec] on a computer with 1.9 MHz Intel Core Duo processor and 2GB RAM memory. This limits the use of the models in computationally intensive analyses like interval uncertainty analysis and Sobol variance decomposition sensitivity analysis. They employ constraint optimization algorithms and multi-dimensional Monte Carlo simulations which may take tens of thousands of simulations for several sets of optimal variables.

Thus, the first goal of this work was to prepare the model and code of Slegers, et al, [9] for uncertainty and sensitivity analyses. The model was formulated as a function to allow user-friendly view of the model inputs, parameters, and variables and their connection to the PBR design, location, year, algae species, and material. Then, the original code was transformed to fit the new input-output functional form so that it could be implemented in algorithms available in the Mathworks MATLAB® library. Subsequently, the simulation time was optimized by testing programming code versions differing in programming constructs, language, parallelism, and discretization of partial differential equations. Finally, a formal constrained optimization was carried out to supply the optimal points in the variable space for a test case. The test case was a PBR in The Netherlands, in the year 2009, made of glass, and operating with P.tricornutum.

The second goal of this work was to investigate the predictive power of the single panel PBR model of Slegers, et al, [9] for a test case to the benefit of LCA studies. A main point of interest was whether or not the model can produce more precise biomass productivities than the ones already used in LCA studies. The input, parameter, and sub – model errors were estimated by interval uncertainty analysis [12]. Interval analysis was used because it models uncertainty in the inputs, parameters, and sub – models with bounds which reflect all possible values. The bounds can be obtained from scarce experimental data, circumstantial evidence, and expert opinions. Then, they can be mapped to biomass productivity bounds which indicate the range of possible values for the ‘true’ biomass productivity. Uncertainty in the numerical schemes was neglected on the grounds of numerical evidence from variable precision simulations.

The last goal of this study was to investigate the impact of individual errors on the accuracy of the model of Slegers, et al, [9]. A main concern of this study was to aid LCA studies in pointing out bottlenecks and the potential benefits of addressing them. A one-at-time sensitivity analysis was performed because it can give a basic indication of the effect of each error on the model inaccuracy [13]. It can also discover if the model is linear in the errors, in which case it can indicate the
fractional contribution of each uncertainty to the model inaccuracy. However, if the model is non-linear, a sensitivity analysis that explores the entire uncertainty space is needed. A Sobol variance decomposition analysis was performed to that end because it can evaluate the fractional contribution of individual uncertainties on the variance of the model output [13].
Model

1. Functional form

Equations 1 – 6 represent the model of Slegers, et al, [9] for single panel photobioreactors as an input – output function. The model output is the specific biomass productivity for single panel PBR $P_{SP} [kg*year^{-1}*m^{-2}(panel\ area)]$. The design variables are the reactor slope $\theta [^\circ]$, surface azimuth angle $\gamma [^\circ]$, light path $d [m]$, and biomass concentration $C_x [kg*m^{-3}]$. It should be noted the slope $\theta [^\circ]$ and the surface azimuth angle $\gamma [^\circ]$ relate to the front side of the PBR, i.e. the side that is facing North and/or West or West and/or South. Figure 2 gives a visual account of the design variables for a sample location, year, algae species, and PBR material. The design variables and the specificity of the biomass productivity are determined by the nature of light irradiation. The light irradiation is constant over the front and back surfaces and depends on the PBR inclination and orientation. How much of the light is captured and used for growth depends on the light path (width) of the PBRs and the algae concentration.

\[
P_{SP} = f_{SP}(x_{SP}, p, I_b, I_d)
\]

\[
x_{SP} = \{\beta, \gamma, d, C_x\}
\]

\[
p_{SP} = \{\lambda, \varphi, \epsilon, f_{p/s}, T_m, \eta_{air}, \eta_{water}, \rho, \alpha, \mu_{max}, r_{max}, \theta_{max}, \sigma, f_{PAR}\}
\]

\[
P_{SP} \in R^{1 \times 1}
\]

\[
x_{SP} \in R^{1 \times 4}, p \in R^{1 \times 5}, I_b \in R^{1440 \times 365}, I_d \in R^{1440 \times 365}
\]

\[
\beta, \gamma, d, \epsilon, C_x, \lambda, \varphi, \mu_{max}, r_{max}, \theta_{max}, \sigma, f_{PAR} \in R^{1 \times 1}
\]

The inputs and parameters are determined by the specifications, i.e. location, year, algae species, and design materials. The inputs are direct $I_b [W*m^{-2}]$ and diffuse $I_d [W*m^{-2}]$ horizontal light irradiation measured on a minute scale for a year. It can be obtained from meteorological stations and satellite images for given location and year. The parameters longitude $\lambda [^\circ]$ and latitude $\varphi [^\circ]$ are determined by the location. The parameters wall transparency $T_m [-]$ and refractive index $\eta_{wall} [-]$ are determined by the material of the PBR. The parameter ground reflectivity $p [-]$ is determined by the material of the surroundings. The parameters light absorption $\alpha [m^2*kg^{-1}]$, growth rate $\mu [s^{-1}]$, maintenance rate $r_m [s^{-1}]$, maximum Chl a: Carbon ratio $\theta_{max} [g(Chl\ a)*g(C)]^{-1}$, and functional cross section of the photosynthetic apparatus $\sigma [g(C)*mol(photons)^{-1}*m^2*g(Chl\ a)]^{-1}$ are determined by the algae species. The parameters angle of incidence of diffuse and reflect light $\epsilon [^\circ]$, ratio between p- and s- polarized light $f_{p/s} [-]$, air refractive index $\eta_{air} [-]$, broth (water) refractive index $\eta_{water} [-]$, and fraction of photosynthetically active radiation $f_{par} [-]$ are fixed at standard values from literature.
2. Conceptual description

The models of Slegers et al, [9] evaluate the biomass productivity of single flat panel PBRs under realistic lighting conditions. The temperature, nutrient, and Carbon dioxide levels are assumed to be controlled at optimal levels for the algae species so that light is the only growth limiting factor, see fig. 3 A). Furthermore, a fractal, i.e. self-similar, design is assumed for the models to apply to large scale installations, see fig. 3 B). A panel row can be viewed as a single flat panel unit which consists of smaller panels with minimal air hold-up and tubing. Thus, the ideal mixing and chemo-stat assumptions can be extended to large scale installations. A wall of panels can be considered a single ideally mixed, chemo-stat unit, and described with the models of Slegers et, al, [9].

The light intensity experienced by a single panel PBR depends on the light intensity of the incoming light rays and their angle of incidence. The former is obtained by horizontal light irradiation data from meteorological stations for a specific location and year. The latter is determined by isotropic models for specific PBR location, orientation, and inclination. Isotropic models consider three types of light rays: direct, diffuse, and reflect. Figure 4 A) shows direct rays which irradiate only one side of the PBR. The direct light has the same angle of incidence over the irradiated PBR surface depending on the solar position. Figure 4 A) also shows diffuse light. It comes from all directions of the sky dome with the same intensity. Figure 4 B) shows the reflected light. It comes from the entire ground with
the same intensity. Both diffuse and reflect light are collectively described by effective rays with the same constant angle of incidence irrespective of the solar position. It should be noted that the irradiation on the top and side surfaces is neglected because they are much smaller than the front and back of the panels. In addition, they are covered by a supporting frame in real-life installations.

Figure 3 Schematic representation of algae cultivation system. Green signifies algae broth, blue water, purple nutrients, and black CO₂ (air) source. A) A controlled single panel PBR with the corresponding tubing and equipment. B) A fractal flat panel PBR in the limiting case of no gas hold-up and tubing.

Figure 4 Schematic representation of insolation of PBRs. Red signifies direct sunlight, orange diffuse light, blue the sky dome, green the PBRs, and brown the surrounding ground. The large orange arrow indicate the effective diffuse light rays, whereas the large orange arrow in red covering indicate the effective diffuse and reflect light ray. A) A PBR irradiated by direct and diffuse light. B) A PBR irradiated by reflect diffuse and direct light.

The amount of light that is reflected from the PBR depends on the angle of incident light and the material of the PBR wall. It is estimated by the Fresnel’s and the Snell’s laws. The latter determines the change in the angle of incidence of incoming light at the air/wall and wall/broth interfaces due to refraction. The former determines the amount of reflected light on the basis of these angles. Figure 5 A) shows the light path of solar rays. It should be noted that the light path and the amount of reflected light is different for direct and diffuse or reflect light. Moreover, the light path in the broth
changes direction due to scattering phenomena. It is assumed to follow Snell’s law in an infinitesimal region close to the wall but to penetrate perpendicular to the light panel wall.

The light and growth profiles inside the PBRs are determined by the Lambert – Beer law and the PI – curve growth models of Geider, et al [14]. The Lambert – Beer law determines the local light intensity for the light penetrating the PBRs and for the specific concentration and absorption capacity of the algae. The PI – curve growth models determines the local growth rates for the local light intensity and for the specific growth characteristics of the species. Figure 5 B) shows the local light intensity profiles follow an exponential decay curve. Additionally, the growth rate model follows a saturation curve which attenuates the exponential curve of the local light intensity.

![Diagram of light reflection and penetration]

**Figure 5** Schematic representations of light reflection and penetration. The orange solid arrows signify light rays, the orange interrupted arrows signify light penetrating the broth, and the orange signifies the light intensity inside the PBR. The red signifies the growth rate at the local light intensity. It should be noted the light penetrating from the right side is stronger. A) The light path of sample solar rays. B) Sample light intensity and growth profiles inside a single PBR.

Overall, the productivity for single panel PBRs $P_{sp} [kg*year^{-1}*m^{-2}(panel area)]$ is expressed by the double integral in eq. 1 and its discretized version in eq. 2. The horizontal solar irradiance $I [W*m^{-2} (height \times width)]$, measured in 1 minute intervals for a year, is averaged for every 10 minutes. The averaged solar irradiance $\bar{I} [W*m^{-2} (height \times width)]$ is used to calculate the photon concentration per unit light path (in the broth) $I_{SP, PFD} [\mu mol photots*m^{-2}(light path)]$. Then, the growth rate $\mu_{sp} [kg*sec^{-1}*m^{-2}(light path)]$ is calculated in 100 intervals along the light path. Finally, the productivity is estimated by a variation of the midpoint rule in time and the trapezium rule in space. It is estimated for all 10 minute time intervals $I [-]$ in a year and all 100 intervals along the light path signified with...
symbols $\Delta t$ and $\Delta t$. Originally, the panels were selected to be 1 metre tall and 1 meter wide. These boundary conditions do not influence the specific productivity but are necessary for the simulations.

$$P_{SP}(t, z) = \int_0^{m \Delta t} \int_0^{i \Delta t} \mu_{SP} \left( I_{SP,PFD}(I(t), z) \right) * dt * dz$$  \hspace{1cm} (7)

$$P_{SP}(t, z, y) = \frac{\Delta t \cdot \Delta z}{2} * ...$$

$$\sum_{j=1}^{m} \sum_{i=1}^{l} \left( \mu_{SP} \left( I_{SP,PFD}(I((t-1) \cdot \Delta t \rightarrow t \cdot \Delta t), (j-1) \cdot \Delta z) \right) \right)$$

$$+ \mu_{SP} \left( I_{SP,PFD}(I((t-1) \cdot \Delta t \rightarrow t \cdot \Delta t), j \cdot \Delta z) \right)$$  \hspace{1cm} (8)
Methods

1. Preparation for uncertainty and sensitivity analysis
   1.1 Improvement of the simulation time

The simulation time of a programming code depends on the programming language [12]. The original code of Slegers, et al. [9] is written in MATLAB. Programming in higher language like MATLAB, PYTHON, and JAVA is easier than programming in lower level languages like C, C++, and FORTRAN. However, higher level languages are slower to run than lower languages because it takes compilers more time to translate them to an executable machine language. A pragmatic approach used in this work to circumvent the trade-off between the ease of programming and the simulation time is to program in MATLAB, and when needed, convert parts of the code in C with the help of the Mathworks MATLAB® coder [15].

The simulation time of a programming code also depends on the used commands and constructs [12]. It could be sub-optimal if the used commands and constructs have higher capabilities and complexity than necessary. In this work, the official performance guidelines for MATLAB were used [16]. They recommend vectorizing arrays, pre-allocating arrays before accessing them within loops, storing and accessing data in columns, avoiding unnecessary variables, using functions instead of scripts, limiting logical statements, and eliminating global variables. It should be noted that vectorizing arrays, i.e. using matrices instead of for loops, is the most important factor that determines the simulation time. Unless the operational memory of the computer becomes a limiting factor, all variables should be defined in matrix form. It should also be noted that the Mathworks MATLAB coder® transform codes in MATLAB to codes in C with efficient constructs. Vectorizing is a major factor the speed of codes in C as well but in this case for loops are preferable.

The amount of processing power that is employed in simulations is determined by the level of parallelism. Most modern computers have a few central processing units (CPUs) that work independent of each other to handle large workloads. The Mathworks MATLAB® program normally employs only one CPU. This is also true when it is calling external applications in C. It evokes more CPUs only for specific operations that can be split into smaller ones, e.g. element by element multiplications in matrix multiplication. This is the default multithread parallelism where one instance of MATLAB automatically generates and executes simultaneous instruction streams [17].

The default multithread parallelism in the Mathworks MATLAB® program is carried out ad hoc for each operation among matrices. Such approach is inefficient for simulations where all operations can
be divided into independent streams with different settings among the available CPUs, e.g. Monte Carlo analysis. A more efficient approach for such case is to use distributed parallelism [17]. The instruction streams are generated manually or automatically and separate MATLAB instances execute them independently on different CPUs within a computer or cluster of computers. In this work, the distributed parallelism was implemented manually on a single computer. A code was developed which splits the simulations into fractions and selects a fraction depending on the CPU number entered into the code. Then, for each separate CPU, a Mathworks MATLAB® instance with its own independent interface, was initiated and the code was run on it. Lastly, a code to combine the results was run on a Mathworks MATLAB® instance when all of them carried out the simulations. More complete and elaborate treatise on parallel computing is available in the appendix.

Lastly, the simulation time of codes for partial differential equations depends on the discretization mesh [12]. The finer the mesh, the more accurate the numerical solution is. However, the number of computations grows geometrically with refining the mesh. A practical approach to balance between speed and numerical accuracy is to evaluate the discretization error of a number of meshes for one set of variables for a test case and select the fastest mesh with tolerable discretization error, e.g. 1 – 5 %. In this work, the reference value for evaluating discretization errors was produced by employing a highly accurate numerical algorithm at a very fine mesh for a test case defined by a set of optimal variables. More information on the discretization error is provided in section 2.1 from this chapter, i.e. Methods.

1.2 Constrained optimization

The purpose of a constrained optimization is to find the best set of variables \( x \) that maximize the output \( y \) of a function, given a set of inputs \( u \), parameters \( p \), and constraints \( x^L \) and \( x^U \) on the variables. A general constrained optimization problem is defined by eq. 9 – 13. The specific equations for the PBRs are presented in section 1.2 of the Results and Discussions chapter. The active set algorithm of the \textit{fmincon} function from the Mathworks MATLAB Optimization Toolbox® was used to solve bounded constrained optimization problems in this work [18]. The active-set algorithm is based on Lagrange multipliers which convert constrained optimization problems to unconstrained ones. It is the fastest algorithm from the ones offered in the \textit{fmincon} function, i.e. trust-reflective, interior-point, and sequential programming ones. It is a small scale algorithm, i.e. it needs to operate on full matrices. This does not cause memory problems because the functions in this work have only one scalar output and four variables. Also, it cannot recover from \textit{Inf} and \textit{Nan} outputs but such results are impossible for the models and the selected constraints in this work.
Given \( y = f(x, p, u) \) \hspace{1cm} (9)
where \( y \in \mathbb{R}^{n_y}, p \in \mathbb{R}^{n_p}, u \in \mathbb{R}^{n_u} \) \hspace{1cm} (10)
find \( \max(y(x)) \) \hspace{1cm} (11)
for \( x^{UB} > x > x^{LB} \) \hspace{1cm} (12)
where \( x, x^{UB}, x^{LB} \in \mathbb{R}^{n_x} \) \hspace{1cm} (13)

The initial search points for the optimization algorithm were selected with the help of Sobol quasi-random generator. Numbers from quasi-random sequences are not truly “random” because they are pre-selected. However, they come from a uniform distribution and have a low discrepancy, i.e. they are evenly distributed in the variable space. The latter property is useful not only in evenly exploring the variable space but also in variance decomposition analyses discussed in section 3.2 of this chapter. An example of quasi-random numbers in two dimensions is shown in fig. 6.

![Sobol quasi-random numbers](image)

**Figure 6** Sobol quasi-random numbers (1024) in two dimensions for variables within zero and one.

2. **Uncertainty analysis**

Errors are quantitative measures of the discrepancy between reality and scientific computing. They can be split into several types depending on their source [19]. The input and parameter errors reflect incorrect specification of the data which describes the system and its environment. The sub-model errors reflect the predictive power of mathematical relationships, assumptions, and conceptualizations. The numerical errors reflect programming mistakes as well as differences between the exact solution of a model and the one produced by discretization, iteration, and round off schemes. Errors arise because of uncertainty. Its modelling, quantification, and propagation, i.e. analysis, can be used to estimate the errors. In this work, the numerical errors were neglected on the
basis of evidence from variable precision simulations; whereas the input, parameters, and sub-model errors were estimated by interval uncertainty analysis.

2.1 Numerical error

The numerical error can be neglected on the grounds of comparisons between variable accuracy simulations. Detailed and exact methods are available for estimation of the numerical errors [12]. However, they were not considered in this work because they are time consuming, impractical, and unlikely to bring significant improvement to the results with highly accurate simulations. The programming errors were checked for by simulations with the original and new programming files for quasi-random points in the design variable space. The round-off errors were checked for by simulations with single, i.e. 8 significant digits, and double, i.e. 16 significant digits, precision for the optimal design variables. The discretization errors were checked for by performing simulations with a standard and a highly accurate scheme for the optimal design variables. The highly accurate scheme employed the fourth order accurate Simpson’s method, e.g. eq. 14 – 15, instead of the original first order accurate midpoint and second order accurate trapezium rules. Moreover, the discretization mesh was substantially smaller than the one used in the standard scheme. Details on the exact discretization can be found in section 2.1 in the chapter Results and Discussion. Iterative errors were neglected because the used algorithms in this work are native Mathworks MATLAB® functions with embedded stopping criteria for difference between sequential iterative steps of 0.001 %.

\[
h_{SP}(z) = \int g_{SP}(t, z) \, dt = \ldots
\]

\[
\ldots = \frac{\Delta t}{3} \left( g_{SP/MP}(t = 0, z) + g(t = t_{end}, z) + \ldots \right)
\]

\[
y = \int h_{SP}(y) \, dz = \frac{\Delta z}{3} \left( f_{SP}(z = 0) + f_{SP}(z = z_{end}) + \ldots \right)
\]

where \( g \) is the model output rate of change per unit space and time \([kg*minute^{-1}*m^{-1}]\), \( h \) the model output rate of change per unit space \([kg*m^{-1}]\), \( n [-] \) the number of points in time without the initial and final one which define the discretization mesh in time, \( o [-] \) the number of points without the initial and final ones which define the discretization mesh in space.
2.2 Input, parameter, and sub-model errors

The input, parameter, and sub-model errors arise because of uncertainty. Its modelling, quantification, and propagation, i.e. analysis, can be used to estimate the errors. Uncertainty is the lack of knowledge, i.e. epistemic, and the inherent variability of the system and its environment, i.e. aleatory, that cause simulation results to deviate from reality [19]. Epistemic uncertainty arises from imperfect understanding of phenomena and/or deficient experimental data. Aleatory uncertainty arises from randomness of observable properties and the relationships among them. More elaborate and complete uncertainty treatment from the point of view of evolutionary epistemology exist [20]. However, the most workable one in the context of systems engineering is the one used here [21].

An interval analysis models uncertainty by bounded sets of possible values which are mapped to a set of bounded possible outputs [21]. Aleatory uncertainty is quantified into errors from specific experimental data, whereas epistemic uncertainty is quantified from weakly related experimental data or expert opinions. The errors are propagated by solving a constrained optimization problem. The optimization problem for a generalized input – output function with errors is defined by eq. 16 – 20. The specific equations for the PBRs are presented in section 2.2 of the Results and Discussions chapter. The optimization problem was solved with the MATLAB® fmincon function with an active set algorithm and with quasi-random initial search points.

\[
\text{Given } y = f(x, p, u, E) \\
\text{where } y \in R^{ny}, x \in R^{nx}, p \in R^{np}, u \in R^{nu} \\
\text{find } \max(y(E)) & \min(y(E)) \\
\text{for } E^{UB} > E > E^{LB} \\
\text{where } E, E^{UB}, E^{LB} \in R^{nE}
\]

where \(E\) is the fractional deviation of the parameters and/or models from their true value within the limits \(E^{LB}\) and \(E^{UB}\).

3. Sensitivity analysis

Uncertainty analysis can only address the output indeterminacy. It cannot point out which uncertainties contribute the most to it. This is the domain of sensitivity analyses [13]. A one-at-a-time and a Sobol variance decomposition analyses were performed in this work. The former can indicate the linearity of the model and give a basic measure of the effect of each error. The latter can rank the importance of uncertainties as well as measure their impact on the output error if the input errors are mitigated for both linear and non-linear models.
In one-at-a-time analysis, the effect of each uncertainty on the output error is studied independently [13], e.g. see eq. 21–28. The input, parameter, and sub-model errors are varied within their bounds and their effect on the model output is studied. This analysis gives information on the linearity of the model. It can show if the individual errors add up to the total error when all uncertainties are considered, or if there are non-linear effects in the models and the uncertainties affect each other. For a model linear in the errors, the one-at-a-time analysis can give the fractional contribution of the individual uncertainties to the output error. Even if the model is nonlinear, one-at-a-time analysis can give an indication of the importance of uncertainties. The individual errors are estimated by solving a constrained optimization problem, defined by eq. 21–28 with the help of the MATLAB® fmincon function with an active set algorithm and with quasi-random initial search points. The specific equations for the PBRs are presented in section 3 of the Results and Discussions chapter.

\[
\Delta Y_1 = f(\Delta E_1, \Delta E_{2-n} e = 0), \Delta Y_2 = f(\Delta E_2, \Delta E_{1,3-n} e = 0), ...
\]

\[
\Delta Y_n = f(\Delta E_n, \Delta E_{1-n} e_{n-1} = 0)
\]

\[
\Delta Y_t = f(\Delta E_{1-p})
\]

\[
L = \frac{\sum_{i=1}^{n} \Delta Y_i}{Y_t}
\]

Given \( y = f(x, p, u, E_n) \)

where \( y \in R^n, x \in R^n, p \in R^n, u \in R^n \)

find \( \max(y(E_n)) \) & \( \min(y(E_n)) \)

for \( E_n UB > E_n > E_n LB \)

\[
\text{where } E_n, E_n UB, E_n LB \in R^1
\]

where \( L \) is the ratio between the sum of individual output errors and the total output error, \( n^e \) the number of errors, \( \Delta Y_i \) is the output error when only a specific error \( \Delta E_i \) is taken into account and \( \Delta Y_t \) the total output error when all model input and model form errors are taken into account.

In Sobol variance decomposition analysis, the effect of the entire input uncertainty space on the entire output uncertainty space is studied [13]. The first or higher order Sobol coefficients indicate the expected change in the variance of the output mean when an error or a combination of errors is removed, see eq. 29–30. Their sum adds up to one, see eq. 31. The total Sobol coefficients indicate the expected change in the output variance when an error is removed, see eq. 32. The total Sobol coefficient is a measure of importance. It indicates by how much the output variability is expected to be reduced if an error is removed. The total Sobol coefficient of an error is equal to all Sobol coefficients than involve that error, see eq. 33.

\[
S_i = \frac{V(E_{-\{i\}}(V(E_i))}{V(Y)}
\]
\[ S_{i,j,...,n} = \frac{V(\bar{E}_{E-i,j,...,n}(Y|E_i,E_j,...,E_n))}{V(Y)} \]  

(30)

\[ 1 = \sum_{i} S_i + \sum_{j>i}^{n-1} S_{ij} + S_{ij,...,n} \]  

(31)

\[ S_{Ti} = \frac{\bar{E}(V_{E-i}(Y|E_i))}{V(Y)} \]  

(32)

\[ S_{Ti} = S_i + \sum_{j} S_{ij} + ... S_{i,j,...,n} \]  

(33)

where \( E_i \) is fractional deviation of the parameters and/or models from their true value, \( \bar{E} \) expected (mean) value, \( S_i \) the first order Sobol sensitivity coefficient, \( S_n \) the total sensitivity coefficient, \( V \) variance, \( Y \) the model output.

For linear models, the total Sobol coefficients equal the first order ones. What is more, the expected change of the output variance when several errors are removed is equal to the sum of their respective total Sobol coefficients. For non-linear models the sum of all total order coefficients is higher than one. Also, the expected change in the output variance when several errors are removed is not equal to the sum of their respective total Sobol coefficients. However, the sum of total Sobol coefficients indicates the maximal expected change in the model output variance when the respective errors are removed. Additional information on the Sobol coefficients and analysis is available in the appendix.

The method of Saltelli, et al. [22] was used to estimate the first order and total Sobol coefficients. A quasi-random matrix with columns twice the number of errors is generated. As a rule of thumb, a sample size of at least \( 2^{10} \) or 1024 points is sufficiently large to eliminate sampling errors. The number of rows \( N \) reflects the sample size for the numerical estimation. It is split into two matrices \( A \) and \( B \), defined in eq. 34 – 35. A set of Matrices \( C \) equal to the number of uncertainty factors are created. The matrices \( C_i \), defined in eq. 36, are identical to the matrix \( B \) apart from the \( i \)th column which is taken from matrix \( A \). Each row of each matrix is used to determine an output response from the models as can be seen in eq. 37 – 40. Lastly, the Sobol coefficients are determined by equations 41 – 42. The specific equations for the PBRs are presented in section 3 of the Results and Discussions chapter.

\[
A = \begin{pmatrix}
E_{1}^{(1)} & E_{2}^{(1)} & \ldots & E_{1}^{(1)} & \ldots & E_{n}^{(1)} \\
E_{1}^{(2)} & E_{2}^{(2)} & \ldots & E_{1}^{(2)} & \ldots & E_{n}^{(2)} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
E_{1}^{(N)} & E_{2}^{(N)} & \ldots & E_{1}^{(N)} & \ldots & E_{n}^{(N)}
\end{pmatrix}
\]  

(34)
\[
B = \begin{pmatrix}
E^{(1)}_{n+1} & E^{(1)}_{n+2} & \cdots & E^{(1)}_{n+i} & \cdots & E^{(1)}_{2n} \\
E^{(2)}_{n+1} & E^{(2)}_{n+2} & \cdots & E^{(2)}_{n+i} & \cdots & E^{(2)}_{2n} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
E^{(N-1)}_{n+1} & E^{(N-1)}_{n+2} & \cdots & E^{(N-1)}_{n+i} & \cdots & E^{(N-1)}_{2n} \\
E^{(N)}_{n+1} & E^{(N)}_{n+2} & \cdots & E^{(N)}_{n+i} & \cdots & E^{(N)}_{2n}
\end{pmatrix}
\]

\[
C_i = \begin{pmatrix}
E^{(1)}_{n+1} & E^{(1)}_{n+2} & \cdots & E^{(1)}_{i} & \cdots & E^{(1)}_{2n} \\
E^{(2)}_{n+1} & E^{(2)}_{n+2} & \cdots & E^{(2)}_{i} & \cdots & E^{(2)}_{2n} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
E^{(N-1)}_{n+1} & E^{(N-1)}_{n+2} & \cdots & E^{(N-1)}_{i} & \cdots & E^{(N-1)}_{2n} \\
E^{(N)}_{n+1} & E^{(N)}_{n+2} & \cdots & E^{(N)}_{i} & \cdots & E^{(N)}_{2n}
\end{pmatrix}
\]

\[
y_A = f(A), \quad y_A \in R^N
\]

\[
y_B = f(B), \quad y_B \in R^N
\]

\[
y_{C_i} = f(C_i), \quad y_{C_i} \in R^N
\]

\[
f_0^2 = \left( \frac{1}{N} \star \sum_{j=1}^{N} y_A^{(j)} \right)
\]

\[
S_i = \frac{y_A \cdot y_{C_i} - f_0^2}{y_A \cdot y_A - f_0^2}
\]

\[
S_{Ti} = \frac{y_B \cdot y_{C_i} - f_0^2}{y_A \cdot y_A - f_0^2}
\]
Results and Discussion

1. Preparation for uncertainty and sensitivity analysis

1.1 Improvement of the simulation time

The simulation time was optimized by testing code versions differing in programming language, commands and constructs, and parallelism. The code of Slegers, et al, [9] in MATLAB was reorganized into a input – output function. All recommended commands and constructs apart from vectorization were implemented. Then, a fully vectorized and a non – vectorized version of the code were created. Both of these versions were transformed into external functions in C with the help of the Mathworks MATLAB coder®. The four codes and the original one were then implemented on 1 CPU, 2 CPUs with the default multithread parallelism, and on 2 CPUs with a custom distributed parallelism. The implementation on 1 CPU was carried out by disabling the other CPUs by the process task manager in Microsoft Windows® 7. The simulation times of individual codes were taken as the average of 10 simulations. They were carried out on a computer with 1.9 MHz Intel Core Duo processor and 2GB RAM memory. The in - silico experiments were performed for the situation in Table 1. It reflects optimal variables from the work of Slegers, et al, [9] for the test case in this work.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The Netherlands</td>
</tr>
<tr>
<td>Year</td>
<td>2009</td>
</tr>
<tr>
<td>PBR material</td>
<td>Glass</td>
</tr>
<tr>
<td>Microorganism</td>
<td>P.tricornutum</td>
</tr>
<tr>
<td>Biomass concentration</td>
<td>2.5 [kg*m⁻³]</td>
</tr>
<tr>
<td>Light path</td>
<td>0.05 [m]</td>
</tr>
<tr>
<td>Slope</td>
<td>90° [deg]</td>
</tr>
<tr>
<td>Surface azimuth angle</td>
<td>-90° [deg]</td>
</tr>
<tr>
<td>Height</td>
<td>1 [m]</td>
</tr>
<tr>
<td>Width</td>
<td>1 [m]</td>
</tr>
<tr>
<td>Interval in time</td>
<td>10 [min]</td>
</tr>
<tr>
<td>Method in time</td>
<td>Modified midpoint rule</td>
</tr>
<tr>
<td>Interval in space</td>
<td>5*10⁻⁴ [m]</td>
</tr>
<tr>
<td>Method in space</td>
<td>Trapezium rule</td>
</tr>
</tbody>
</table>

Figure 7 and Table 2 show the in – silico experiments which were used to estimate the simulation time of codes differing in programming language, commands and constructs, and parallelism. The results on 1 CPU are not affected by parallelization. Vectorization was beneficial for codes in MATLAB and not beneficial to codes in C as expected. However, the codes in C were not faster than the codes in MATLAB. The extra time it takes MATLAB to call external applications might have negated the faster machine code compilation. In addition, the codes in C may include sub-optimal commands and
constructs because it was generated automatically with the help of the Mathworks MATLAB coder®. It should be noted the fully vectorized codes in MATLAB and C, together with the partially vectorized original code, could not be applied to a very fine mesh, see Table 3. Vectorization improves the simulation time but does so at the expense of operational memory because numerous matrices with large amount of data are operated on. The non-vectorized code in C is useful for such cases because it is almost as fast as the vectorized one but without the requirement for large amount of operational memory.

Figure 7 Simulation time of the developed codes for the models of Slegers, et al [9] for the situation described in Table 1.

Table 2 Simulation time of the developed codes for the models of Slegers, et al [9] for the situation described in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>MATLAB</th>
<th>Original</th>
<th>MATLAB+C</th>
<th>MATLAB+C</th>
<th>MATLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(non-vectorized)</td>
<td>(partially vectorized)</td>
<td>(vectorized)</td>
<td>(non-vectorized)</td>
<td>(vectorized)</td>
</tr>
<tr>
<td><strong>1 CPU</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28.16</td>
<td>15.36</td>
<td>4.43</td>
<td>3.71</td>
<td>1.68</td>
</tr>
<tr>
<td><strong>2 CPUs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Multithreaded)</td>
<td>29.18</td>
<td>15.68</td>
<td>4.65</td>
<td>3.84</td>
<td>1.13</td>
</tr>
<tr>
<td>(Distributed)</td>
<td>15.79</td>
<td>8.66</td>
<td>2.51</td>
<td>2.04</td>
<td>1.07</td>
</tr>
</tbody>
</table>
Table 3: Capacity of the tested codes to execute a simulation for the situation in Table 1 at a fine discretization, i.e. 1 [min] time and 5*10^4 [m] in space. The green letter ‘V’ indicates success whereas the red letter ‘X’ indicates failure.

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>MATLAB</th>
<th>MATLAB+C</th>
<th>MATLAB</th>
<th>MATLAB+C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Vectorized</td>
<td>Vectorized</td>
<td>Non-vectorized</td>
<td>Non-vectorized</td>
</tr>
<tr>
<td>2 CPUs (Custom distributed parallelism)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

Parallelism positively affected the simulation speed of some codes. The default multithread parallelism on 2 CPUs improved the vectorized code in MATLAB as expected, and did not improve the non-vectorized one as well as the external codes in C. Yet, it also did not improve the performance of the partially vectorized original one. A closer inspection of the simulation time of all programming lines with the Mathworks MATLAB profiler® showed the part of the original code with matrices accounted for only a few percent of the total simulation time on 1 CPU. The efficiency of the matrices significantly improved the performance of the original code in comparison to the non-vectorized code in MATLAB on 1 CPU. Yet, this efficiency meant insignificant amount of time could be shed off on 2 CPUs. The distributed parallelism improved all simulations because it carried all instructions independently on each CPU. The improvement was less than 2, i.e. the number of CPUs, because the CPUs were on the same core with shared memory and had synchronize their action. Nonetheless, the default multithread parallelism performed just as well as the custom distributed one for the vectorized code in MATLAB. It appears the multithread parallelism spends minimal time in generating and combining the instructions ad hoc for each matrix operation.

Figures 8 – 9 show the speed and the accuracy of the vectorized code in MATLAB for various discretization intervals. The ‘true’ value was estimated by the non-vectorized code in MATLAB and C. It employed numerical integration with Simpson’s rule and a discretization of 1 [min] in time and 1000 intervals in space. The simulation time was similarly affected by the time and space discretization. However, this was not the case for the accuracy. The discretization in space was the stronger contributor. The discretization that was selected for all intents and purposes in this work was 60 [min] time intervals and 50 space intervals. It was chosen subjectively. The number of space steps was selected to be small but not excessively so because it may play a significant role for simulations with different parameter and variable settings. The size of the time interval was selected for the same reason and for minimizing the numerical error to within 3%.
Overall, the fully parallelized code in MATLAB and implemented in the custom parallel architecture was the fastest, see figure 7 and Table 1. It enabled uncertainty and sensitivity analyses which can take a few tens of thousands of simulations for one set of optimal variables. The fully parallelized code in MATLAB was 15 times faster than the original code implemented in the default multithread parallelism. The larger discretization used in this work in comparison to the work of Slegers, et al, [9] brought an additional 6.3 fold improvement, see fig. 8 – 9, for a total improvement of 94.5 times. Furthermore, the non-vectorized code in MATLAB and C, implemented with the custom distributed parallelism, allowed simulations at very fine discretization, i.e. 1 [min] intervals in time and 1000 intervals in space. The original codes were unable to perform simulation at very fine meshes because they are partially vectorized and the operational memory is a limiting factor. No exact estimation was made of the required number of simulations for the analyses in this work. Yet, they took several days of computational time. The same analyses with the original code and mesh would have taken a few months.

1.2 Constrained optimization

The design variables were optimized by solving two constrained optimization problems, defined by eq. 43 – 50. The first optimization problem was to find the optimal biomass concentration which maximizes the biomass productivity for fixed slope and surface azimuth angle at 90° [deg] and –90° [deg] respectively. The constraints for the biomass concentration were 1 and 13 $[kg*m^{-3}]$. The optimization was expected to reveal if various combinations of light paths and concentrations lead to the same optimal biomass productivities at fixed slope and surface azimuth angle. The second optimization problem was to find the optimal biomass concentration, slope, and surface azimuth angle. The constraints on the PBR slope were 0.1 and 90 [°], and for the PBR surface azimuth angle 0
and – 90° [deg]. The second optimization was expected to reveal if upright PBRs facing East and West were optimal and if the optimal set of light paths and biomass concentrations is different when the slope and surface azimuth angles are not fixed. The optimizations were performed for 8 Sobol quasi-random initial search points in the design variable space with the help of the Mathworks MATLAB® fmincon function which employs an active set algorithm.

\[
P_{SP} = f_{SP}(x_{SP}, p, I_b, I_d)
\]

\[
x_{SP} = \{\beta, y, d, C_x\}
\]

\[
p_{SP} = \{\lambda, \varphi, \epsilon, f_{p/5}, T_m, \eta_{air}, \eta_{wall}, \eta_{water}, \ldots\}
\]

where

\[
P_{SP} \in R^1 \times 1, x_{SP} \in R^1 \times 4, p \in R^1 \times 14, I_b \in R^{1440 \times 365}, I_d \in R^{1440 \times 365}
\]

Given

Find

\[
ma \ x(P_{SP}(x[3,4]))
\]

for

\[
[0.1, 13] > x_{SP}{[3,4]} > [0.01, 1] \text{ and } x_{SP}{[1,2]} = [90, -90]
\]

and find

\[
ma \ x(P_{SP}(x_{SP}))
\]

for

\[
[90, 0, 0.1, 13] > x_{SP} > [0.1, -90, 0.01, 1]
\]

Figure 10 shows the optimal biomass concentration for various light paths at fixed or optimal slope and surface azimuth angle. Figure 11 shows the optimal biomass productivities for optimal biomass concentration, slope, and surface azimuth angles as well as for the optimal biomass productivity at fixed slope and surface azimuth angles. There are various combinations of light paths and biomass concentrations that lead to the same optimal biomass productivities as indicated in the work of Slegers, et, al [9]. There is an inversely proportional relationship between the optimal biomass concentration and the optimal light path. The slope and surface azimuth angles of 90° [deg] and - 90° [deg] do not appear to be optimal in contrast to the results in the work of Slegers, et, al [9], see figures 12 and 13. However, the biomass productivities of upright PBRs facing East and West are very close to the optimal ones, i.e. within 2.5 %. This difference is in the order of the numerical error of 3 % for the discretization mesh used in this work and is negligible. Lastly, the optimal slope and surface azimuth angle may not be seemingly independent of the light path. The close biomass productivities confirm the observation of Slegers, et, al [9] that the biomass productivity is insensitive to the slope and surface azimuth angle. The stopping criteria of the optimization algorithm may be fulfilled before the truly optimal points are identified.
2. Uncertainty analysis

2.1 Numerical error

The codes of Slegers, et al, [9] were transformed to improve the simulation time. However, the new codes were not verified for programming errors. To check for programming errors, the new vectorized code in MATLAB and the new non-vectorized code in C were compared against the original codes. The comparisons were made for $2^8$ (128) points in the design variable space, limited by the design variable space constraints used for the optimizations in section 1.2 of this chapter, i.e. Results and Discussion. The new C code, verified against the original one, employed the original modified midpoint and the trapezium rules. The same version was used in section 1.1 of this chapter for evaluation of the simulation speed. It should be noted that the new C code used for highly
accurate simulations at a very fine discretization mesh employed Simpson’s rule for numerical integration. It was not used in the verification because it is a close derivative of the code in C with the original numerical integration schemes. Moreover, it is going to introduce discretization error in the comparisons because of the different numerical integration method from the original one.

Figures 14 – 15 show the absolute errors between the original code, the new vectorized code in MATLAB, and the new non–vectorized code in C. They are always in the range of 0.010 to 0.018 [kg*m^2*year^{-1}] regardless of the design variables and the biomass productivity. The optimal biomass productivity is in the range from 6.5 to 7 [kg*m^2*year^{-1}]. The programming errors are insignificant. Their relative impact becomes significant only for the near zero productivities. Moreover, the differences between the new codes and the original one may occur not only because of the programming mistakes but also because of round-off errors. The new codes implemented the models with different programming constructs and intermediate variables than the original one.

The discretization error was evaluated by performing simulations with the original numerical integration methods and a highly accurate one. The round-off errors were evaluated by performing single and double (default for Mathworks MATLAB®) precision simulations. The original integration methods standard scheme involved the original modified midpoint and trapezium rules for numerical integration. It utilized 60 [min] discrete intervals in time and 50 intervals in space. The highly accurate numerical method for evaluating the discretization error involved the Simpson’s rule. It utilized 1 [min] intervals in time and 1000 intervals in space. The double and single precision simulations to evaluate the round–off error utilized the original numerical integration methods and a discretization of 60 [min] time intervals and 50 intervals in space. The variable accuracy simulations were performed for six optimal points in the variable space, see Table 4. For three combinations of
the optimal light path and biomass concentrations, the slope and surface azimuth angles were fixed at 90° [deg] and -90° [deg] according to the work of Slegers, et al, [9]. For the same combinations of optimal light paths and biomass concentrations, the slope and surface azimuth angles were fixed at their optimal values determined in section 1.2 in his chapter. The three combinations of light paths and biomass concentrations have low – high, medium – medium, and high – low values.

Figures 16 – 17 show the simulations with variable accuracy for the variables listed in Table 4. The differences between the single and double precision simulations were less than 0.005 %. The difference between the highly accurate numerical integration method and the original ones was around 3 % regardless of the design variables as long as they are optimal. This is in agreement with the error determined in section 1.1 of this chapter where the discretization mesh was selected. The numerical error was considered small and was neglected in this work. Detailed and exact methods for estimation of the discretization and round – off errors exist [12]. However, they are not expected to bring much improvement in the estimations with variable precision simulations. It should also be noted that the numerical error may change in uncertainty and sensitivity analyses where the parameters and sub – model outputs are varied. The numerical error was not estimated for the entire parameter and sub – model space because of the computationally intensive simulations that are required. In this work, it was assumed the error is negligible in the entire uncertainty space.

Table 4 Optimal points in the design variable space where the model accuracy was estimated.

<table>
<thead>
<tr>
<th>Light path [m]</th>
<th>Biomass concentration [kg*m⁻³]</th>
<th>Slope [°]</th>
<th>Surface azimuth angle [°]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>4.38 / 4.35</td>
<td>78.77 / 90</td>
<td>-78.75 / -90</td>
</tr>
<tr>
<td>0.06</td>
<td>2.19 / 2.18</td>
<td>80.94 / 90</td>
<td>-74.02 / -90</td>
</tr>
<tr>
<td>0.09</td>
<td>1.46 / 1.45</td>
<td>81.69 / 90</td>
<td>-74.21 / -90</td>
</tr>
</tbody>
</table>
2.2 Input, parameter, and sub-model errors

Light irradiation is described by isotropic light models in the work of Slegers, et al, [9]. The accuracy of the isotropic light models and parameters was estimated on the grounds of the experimental study of Loutzenhiser, et al, [23]. A measurement error of ±3% was used for the measured horizontal light irradiation data. An error of ±5% was used for the ground reflectivity. A modelling error of ±15% was used for direct, diffuse, and reflect light, which corresponds to the mean absolute error in the total light irradiation. Splitting the experimental error of the total irradiance is necessary because the different types of light have different angle of incidence, and thus, fraction of reflected light at the panel wall.

The Fresnel equations and Snell’s law are used to model light reflection and refraction in the work of Slegers, et al, [9]. Their accuracy was estimated on the grounds of the work of Duffie and Beckman which is based on published experimental data [24]. No modelling errors were considered because they are highly accurate physical relationships. The same accounts for the effective angle of incidence of direct light. It is predicted by highly accurate models for the solar position. The angle of incidence of diffuse and reflect light vary within 1% for a fixed PBR slope. However, the effective angle of incidence of reflect light vary from 90 to 60°, whereas the angle of incidence of diffuse light vary from 55 to 60°, with varying the PBR slope from 0 to 90°. No error was used in this work for the effective angles of light incidence because the PBR slope in the cases of interest to this work were close or equal to 90°. No error was used for the refractive indexes of air, glass, and water (broth) as well as for the wall transparency because they have been measured with highly accurate methods and listed in reference physical textbooks with accuracy within ±0.5% [25].
The Beer–Lambert law was used to model light absorption in the work of Slegers, et al, [9]. Its accuracy was estimated on the grounds of the experimental work of Fernandez, et al, [26]. It showed the attenuation is exponential and not linear in biomass concentrations above 1.5 [kg*m⁻³]. In this work, a – 10% model form error was used on the basis of measured irradiance profiles inside a tubular PBR up to a concentration of 4.0 [kg*m⁻³] for P. tricornutum. A – 50% error was used for the spectrally averaged absorption coefficient on the basis of data from several batches of P. tricornutum. No errors for the fraction of the photosynthetically active radiation were selected because it varies within 1% as a rule of thumb [27].

Algae growth is modelled by PI – growth model of Geider, et al, [14] for balanced growth. This model is very accurate for cells which are adapted to the prevalent light and nutrient conditions and increase their numbers without significantly changing their composition. The only source of error in this model is the parameters which determine the adapted state of algae. The bounds were chosen to be at least within the listed standard deviations for P. tricornutum in the work of Geider, et al, [28] where an overview of published experimental data is given. An error of ±20% was used for the maximum growth rate and the maintenance rate. An error of ±8% was used for the maximum Chl a: Carbon ratio $\theta_{\text{max}}$. An error of 40% was used for the light absorption cross section of the photosynthetic apparatus. In principle, the light absorption cross section of the photosynthetic apparatus and the spectrally averaged absorption coefficient are related through the photosynthetic efficiency. However, they were allowed to vary independently in this work because it is focused on conservative estimations and the photosynthetic efficiency varies somewhat.

Interval analysis was used to estimate the biomass productivity errors, given the input, parameter, and sub-model errors in Tables 4 and 5. It was performed for the six optimum variable points listed in Table 3. Interval analysis was used to estimate the best and worst possible biomass productivities. The bounds of the model input and model form errors were propagated by solving a constrained optimization problem defined by eq. 51 – 58. The constrained optimizations were performed for 8 Sobol quasi-random initial search points in the variable space with the help of the Mathworks MATLAB® fmincon function which employs an active set algorithm.
Table 4. An overview of the inputs and parameters as well as the associated model input

<table>
<thead>
<tr>
<th>Input and parameters</th>
<th>Symbol</th>
<th>Units</th>
<th>Value(s)</th>
<th>Error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal direct light</td>
<td>$I_b$</td>
<td>[W*m$^{-2}$]</td>
<td>(Too large dataset to list)</td>
<td>[-3:3]</td>
</tr>
<tr>
<td>Horizontal diffuse light</td>
<td>$I_d$</td>
<td>[W*m$^{-2}$]</td>
<td>(Too large dataset to list)</td>
<td>[-3:3]</td>
</tr>
<tr>
<td>Latitude</td>
<td>$\varphi$</td>
<td>[°]</td>
<td>51.97° N</td>
<td>0</td>
</tr>
<tr>
<td>Longitude</td>
<td>$\lambda$</td>
<td>[°]</td>
<td>4.93° E</td>
<td>0</td>
</tr>
<tr>
<td>Ratio between p- and s-polarized light</td>
<td>$f_{p/s}$</td>
<td>[-]</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Angle of incidence diffuse and reflect light</td>
<td>$\epsilon$</td>
<td>[°]</td>
<td>60</td>
<td>[0:50]</td>
</tr>
<tr>
<td>Wall transparency</td>
<td>$T_m$</td>
<td>[-]</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Refractive index of air</td>
<td>$\eta_{air}$</td>
<td>[-]</td>
<td>1.25</td>
<td>0</td>
</tr>
<tr>
<td>Refractive index of glass</td>
<td>$\eta_{wall}$</td>
<td>[-]</td>
<td>1.25</td>
<td>0</td>
</tr>
<tr>
<td>Refractive index of water (broth)</td>
<td>$\eta_{water}$</td>
<td>[-]</td>
<td>1.25</td>
<td>0</td>
</tr>
<tr>
<td>Ground reflectivity</td>
<td>$\rho$</td>
<td>[-]</td>
<td>0.4</td>
<td>[-5:5]</td>
</tr>
<tr>
<td>Spectrally averaged light absorption coefficient</td>
<td>$\alpha$</td>
<td>[m$^2$*kg$^{-1}$]</td>
<td>7</td>
<td>[-50:0]</td>
</tr>
<tr>
<td>Maximum growth rate</td>
<td>$\mu$</td>
<td>[s$^{-1}$]</td>
<td>1.6</td>
<td>[-20:20]</td>
</tr>
<tr>
<td>Maintenance rate</td>
<td>$r_m$</td>
<td>[s$^{-2}$]</td>
<td>0.05</td>
<td>[-20:20]</td>
</tr>
<tr>
<td>Maximum Chl a: Carbon ratio</td>
<td>$\theta_{max}$</td>
<td>[g(Chl a)*g(C)$^{-1}$]</td>
<td></td>
<td>[-8:8]</td>
</tr>
<tr>
<td>Functional cross section of the photosynthetic apparatus</td>
<td>$\sigma$</td>
<td>[g(C)*mol(photons)$^{-1}$*m$^2$*g(Chl a)$^{-1}$]</td>
<td></td>
<td>[-40:40]</td>
</tr>
<tr>
<td>Fraction of photosynthetically active radiation</td>
<td>$f_{par}$</td>
<td>[-]</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. An overview of the models and the associated model form errors.

<table>
<thead>
<tr>
<th>Model</th>
<th>Symbol</th>
<th>Units</th>
<th>Error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isotropic light</td>
<td>$ILE$</td>
<td>[W*m$^{-2}$]</td>
<td>[-15:15]</td>
</tr>
<tr>
<td>Fresnel</td>
<td>$FE$</td>
<td>[W*m$^{-2}$]</td>
<td>0</td>
</tr>
<tr>
<td>Beer – Lambert</td>
<td>$BLE$</td>
<td>[W*m$^{-2}$]</td>
<td>[-10:0]</td>
</tr>
<tr>
<td>PI – curve</td>
<td>$PE$</td>
<td>[W*m$^{-2}$]</td>
<td>0</td>
</tr>
</tbody>
</table>
\[ P_{SP} = f_{SP}(x_{SP}, p, I_b, I_d, E) \]  
(51)

\[ x_{SP} = \{\beta, \gamma, d, C\} \]  
(52)

\[ P_{SP} = \{\lambda, \varphi, \epsilon, f_{p/50}, T_m, \eta_{air}, \eta_{water}, \ldots\} \]  
(53)

\[ E = \{E_{lb}, E_{ld}, E_c, E_{p}, E_{at}, \eta_{max}^\prime, \ldots\} \]  
(54)

\[ P_{SP} \in R^1, x_{SP} \in R^1, p \in R^1, I_b \in R^{1440 \times 365}, I_d \in R^{1440 \times 365} \]  
(55)

\[
\text{find} \quad \max(P_{SP}(E)) \quad \text{and} \quad \min(P_{SP}(E))
\]  
(56)

\[
E \leq \begin{cases} 
1.03, 1.03, 1.03, 1.05, 1.12 , & \ldots \\
1.2, 1.08, 1.4, 1.15, 1 & 
\end{cases}
\]  
(57)

\[
E \geq \begin{cases} 
0.97, 0.97, 1, 0.95, 1.08 , & \ldots \\
0.8, 0.92, 0.6, 0.85, 0.9 & 
\end{cases}
\]  
(58)

Figure 18 shows the biomass productivity error estimated by interval analysis for the six optimum variables listed in Table 3. The interval analysis showed that the true biomass productivities are in the interval between –45% and +105% of the predicted ones. The estimated biomass productivities vary by a factor of 3.72. This is a significant improvement to the factor of 25 used in life cycle studies [8]. Therefore, the models have a high predictive power and can yield more precise biomass productivities than the ones used in LCA studies. Moreover, the accuracy of the model was virtually the same for the six optimal variable sets. It appears to be independent of the variables as long as the productivity is at its optimum.

Figure 18 The total biomass productivity error estimated by interval analysis at the optimal variables in Table 4.
3. Sensitivity analysis

One-at-a-time and Sobol variance decomposition analyses were used to estimate the linearity of the model and the biomass productivity errors, given the input, parameter, and sub-model errors in Tables 4 and 5. They were performed for the six optimum variable points listed in Table 3. The errors were propagated by a constrained optimization problem defined by eq. 59 – 66. The Sobol sensitivity analysis was performed according to the procedure in section 1.3 in the chapter Methods. The exact equations are not listed here because the procedure for the specific case is considered self-evident. The numerical estimation was carried out with \(2^{10}(1024)\) quasi-random points in the uncertainty space.

\[
\text{For } n=1:11
\]
\[P_{SP} = f_{SP}(x_{SP}, p, I_b, I_f, E)\]  
(59)

\[
x_{SP} = \{\beta, \gamma, d, C_x\} \]  
(60)

\[
p_{SP} = \{\lambda, \phi, \epsilon, f_{p/s}, f_{m}, \eta_{air}, \eta_{wall}, \eta_{water}, \ldots\} \]  
(61)

\[
E = \{E_{1b}, E_{1f}, E_{e}, E_{p}, E_{a}, E_{\mu_{max}}, \ldots\} \]  
(62)

\[
\text{where } P_{SP} \in \mathbb{R}^{1 \times 1}, x_{SP} \in \mathbb{R}^{1 \times 4}, p \in \mathbb{R}^{1 \times 14}, I_b \in \mathbb{R}^{1440 \times 365}, I_f \in \mathbb{R}^{1440 \times 365} \]  
(63)

\[
\text{find } \max(P_{SP}(E(n))) \text{ and } \min(P_{SP}(E(n))) \]  
(64)

\[
E \leq \left\{1.03, 1.03, 1.05, 1.05, 1.12, \ldots\right\} \]  
(65)

\[
E \geq \left\{0.97, 0.97, 1.0, 0.95, 1.0, 0.8, \ldots\right\} \]  
(66)

Figure 19 compares the sums of all individual errors from the one-at-a-time sensitivity analysis and the total errors from the interval uncertainty analysis for the six optimum variable sets in Table 4. The individual errors accounted for 92% of the total error determined by interval uncertainty analysis. Thus, the model is nonlinear in the errors. However, an 8 % deviation from linearity is not significant. Figure 22 shows the ratio between the first and total order coefficients. The four most important factors according to the total Sobol coefficients in figure 21 are linear, i.e. the first and total coefficients are the same. The nonlinearity originates from the rest of the factors. Therefore, the individual errors in figure 20 for the four most important factors actually indicate their contribution to the total error. Thus, the uncertainty in the spectrally averaged absorption coefficient, the maximum growth rate, and the functional cross section of the photosynthetic apparatus account for 82 % of the total error. Furthermore, the variance of the biomass
productivity is expected to be reduced by 97% if that uncertainty is addressed. Lastly, it should be noted that Sobol coefficients cannot take negative values, which is contrast to the some results in figures 21 and 22. Such results occur when the Sobol coefficients are so small that the numerical error of the method of Saltelli, et al, [22] drives the coefficient estimations to negative values.

Figure 19 Ratio between the sum of the individual errors in OAT and the total errors in interval analysis.

Figure 20 The biomass productivity error for each model input or model form error when the rest are neglected and its variability for the six optimal configurations (error bars).

Figure 21 The total Sobol coefficients for all errors (bars) and their variability for the six optimal configurations (error bars).

Figure 22 The ratio between the total and first order Sobol coefficients (bars) and their variability for the six optimal configurations (error bars).
Further Research

This work needs to be extended to facilitate the use of the models of Slegers, et, al, [9] for technical and design studies. It focused on achieving that end for LCA studies. It estimated the accuracy of the model in predicting the performance of optimal PBR designs which have been determined by optimization of the model without uncertainty analysis. However, this may be of little use to technical and design studies. They are not concerned with the biomass productivity of the optimal PBR designs as much as with all possible PBR designs that could be optimal for a test case and should be tested experimentally. For example, figure 26 shows an interval uncertainty analysis performed on the relationship of the optimal light paths and biomass concentrations. It shows the truly optimal set of light paths and biomass concentrations is within a range of values around the relationship in figure 10 which was identified by constrained optimization and assumed perfectly accurate model.

In addition, there is a need to perform uncertainty and sensitivity analysis for models of various systems. The modelling framework of Slegers, et al, [9] has been extended in more recent works for multiple panel, open pond, single and multiple tubular PBRs [9, 29, 30]. These models can be used to discover the best PBR designs for various locations, years, algae species, and PBR materials. An additional uncertainty and sensitivity analyses would be able to indicate what the confidence is in selecting one type of production system over another and what needs to be researched to improve it. Such information can be used in global mapping of the potential of the technology and encourage policy makers and investors worldwide to participate in the advancement of the technology.

**Figure 23** The optimal biomass concentration bounds for various light paths at fixed or optimal PBR slopes and azimuth angles.
Conclusion

The first goal of this work was to prepare the programming code of Slegers, et al, [9] for single panel PBRS for uncertainty and sensitivity analyses purposes. The speed of the original code was improved by testing code versions differing in the programming constructs, language, parallelism, and discretization of partial differential equations. A new vectorized code in MATLAB with a coarse discretization and implemented through a distributed parallel architecture, was 94.5 times faster than the original code and allowed uncertainty and sensitivity analyses within a reasonable time. Furthermore, the optimal variables which define a single panel PBR design for a test case, i.e. The Netherlands, in the year 2009, made of glass, and operating with *P.tricornutum* were determined by constrained optimization. The global optimization of the variables indicated there are a number of light path and biomass concentration combinations that lead to the same optimal productivities. This coincides with the results of the one and two-at-a-time explorations of the variable space in the work of Slegers, et al, [9]. However, upright and East – West facing panels were not necessary optimal like it was indicated in the latter work.

The second goal of this work was to investigate the predictive power of the single panel PBR model of Slegers, et al, [9] for a test case to the benefit of LCA studies. It was discovered that the models can supply more precise productivities than the ones that have already been used in LCA studies. The true biomass productivities were found to be between – 45 and + 105 % of the model predictions for the optimal PBR designs defined by the PBR slope, surface azimuth angle, light path, and biomass concentration. The error bounds were within a factor of 3.72 from each other which is significant improvement to the typical factor of ~25 used in LCA studies.

The third goal of this study was to investigate the impact of individual errors on the accuracy of the model of Slegers, et al, [9]. It was discovered that the uncertainty in three parameters was responsible for 82 % of the total error and 97 % of its variance. The accuracy of the model can be improved if the uncertainty in the parameters spectrally averaged absorption coefficient, maximum growth rate, and functional cross section of the photosynthetic apparatus is eliminated. The model was linear in these parameters and improvement in the accuracy by addressing the uncertainty in one or two of them can be scaled. However, the individual errors accounted for 92% of the total error when all errors were taken into account. The model is non – linear in the rest of the inputs, parameters and sub – models.
Finally, the effect of uncertainty on the determined optimal design variables was estimated by interval analysis. This was done to encourage future extension of this work to the accuracy of the determined optimal design variables.

1. Parallel computing basics

The amount of processing power that is employed for execution of a code is determined by the amount of available central processing units (CPUs) that are involved. This depends on the amount of independent instructions as well as the number of CPUs and their independence. Figure 24 A) and B) shows how two independent multiplications can be split between two CPUs, cutting the execution time in half. In Figure 24 C) and D), the multiplied numbers have to be added. The improvement by parallelization in this case is not proportional to the available CPUs because parts of the code cannot be executed by several CPUs. In addition, the CPUs have to communicate. The main CPU has to make sure the other one has finished its operation before it can initiate addition. Hardware memory limitations can also require CPUs to communicate and lower the performance. Figure 24 E) shows a case where both numbers are on a single physical unit of memory which can be accessed only by one CPU at a time. The latter case is pronounced for CPUs which are on the same core, i.e. chip. Such CPUs share a fixed internal memory unit on the core itself to support their control, arithmetic, and logical functions.

Parallelization of operations is carried out by the software operational system on global scale. The higher the computational demands of the program, the higher the processing power that is allocated to it. The type of demanded computational power, i.e. lineal vs. parallel is also of importance in the allocation process. When all computational power is taken up, the priority of the programs determine which programs suffer from the limitation and to what extent. The default ability of MATLAB® to identify parallelizable structures within a code or allow pre-programming of specific parallel architectures is limited. Threads, i.e. operations, in MATLAB are run in a lineal fashion and are allocated to one logical unit at a time. However, certain operations which are repeated multiple times for different bits of data, e.g. element by element multiplication in matrix multiplication, are split and carried out simultaneously over a few CPUs. This is part of MATLABs® default multithread ability.
The multithread ability of MATLAB is inefficient when a set of lineal instructions have to be carried out for multiple cases. It does not offer optimal parallelization when many operations between vectors and matrices are involved. Figure 25 A) shows matrices with two independent parts. Each element of each part has to undergo like and independent transformations that results into an element at the same position in the end matrix. Figure 25 B) shows there is a lot of communication between the CPUs if the matrices are split and combined for every transformation step. Figure 25 C) shows the performance would be near the theoretical maximum, i.e. the speed would be proportional to the number of CPUs, if the CPUs only communicate at the beginning and end to split the cases and combine the results.

In this work, a simplistic approach is used to parallelize numerous (set of) independent simulations in optimizations, uncertainty, and sensitivity analyses. A code was developed which splits the cases and selects the case depending on the CPU number entered into the code. Then, for each separate CPU a MATLAB instance with its own independent interface was initiated and the code was run on it. Lastly, a code to combine the results was run on a MATLAB instance when all of them executed the simulations. In principle, the Parallel Computing Toolbox® provides programming constructs to split the cases, run them on separate instances, and combine the results. However, it was not used in this work because it is not available in the libraries of student MATLAB® editions. The approach in this work is slower than the ones available in the Parallel Computing Toolbox®. However, the difference is of a few minutes, which is deemed acceptable for simulations in the order of an hour.
Figure 24 Scheme for demonstrating parallel computing architectures. One operation takes time $t_o$. The time for each case is signified with a $t$ and a subscript indicating the case. The arrows indicate operations and the numbers above them their sequence. The brown ellipsoids signify data in the computer memory. The purple boxes signify the central processing units (CPUs). The green boxes signify the relative execution time of the architecture. The grey box signifies the physical units of memory which can only transmit data to one CPU at a time.
Figure 25 Scheme for demonstrating the difference between the default multithreading ability of Matlab and a more complex parallel structure. The black arrows indicate operations and the numbers above them their sequence. The black boxes signify the central processing units (CPUs). The green boxes signify the relative execution time of the architecture. Blue and red rectangles signify parts of a matrix. Green and yellow rectangles signify the result of N number of transformations to the red and blue parts of the initial matrix. Rectangles of blue and red with lower colour intensity signify intermediate results. The time for each case is signified with a t and a subscript indicating the case. One transformation to a blue or red matrix parts takes time $t_{\sigma}$. Splitting and combining a matrix parts takes a time signified with $t_s$ which is much lower than $t_{\sigma}$. The time for each case is signified with a t and a subscript indicating the case.
2. Sobol variance decomposition analysis

The Sobol sensitivity coefficients arise from the properties of models which can be represented in high dimensional form (HDMR) [13], e.g. eq. 47. Such models can be expressed as combination of functions which depend on the individual function arguments or any combination of them. The models of Slegers, et al, [9] are too involved to check if they can be restructured in such a form. However, they are continuous and very accurate truncated Taylor series exist given enough expansion terms. The total Sobol coefficient of an error is the sum of all Sobol coefficients which involve that error, e.g. see eq. 48. Moreover, all Sobol coefficients add up to one, e.g. see eq. 49. Thus, the total Sobol coefficient of an error is equal to the first order one if the model is linear in that error. Furthermore, if the model is linear in all errors, the sum of first order coefficients and the sum of total coefficients are both equal to one.

\[ f(E_{1...n}) = f_0 + \sum_{i}^{n} f_i(E_i) + \sum_{i}^{n-1} \sum_{j>i}^{n} f_{ij}(E_i, E_j) \cdots f_{ij...n}(E_i, E_j, \ldots E_n) \]  

\[ S_{Ti} = S_i + \sum_{j}^{n} S_{ij} + \cdots S_{ij...n} \]  

\[ 1 = \sum_{i}^{n} S_i + \sum_{i}^{n-1} \sum_{j>i}^{n} S_{ij} \cdots + S_{ij...n} \]

The Sobol coefficients can be determined analytically or estimated numerically [13]. Analytical computations are not an option, considering the complexity of the models of Slegers, et al, [9]. Developing meta-models, i.e. regression model approximations to the original models, and then evaluating the Sobol coefficients analytically is an option but was not considered in this work. Table 6 shows the available methods for numerical estimations and their computational cost [22]. In this work, the method of Saltelli was used because it can estimate both the first order and the total Sobol coefficients. It was also selected because it is procedurally simple to implement.

| Table 6. Numerical methods, their capabilities, and computational cost for variance decomposition analysis [22]. The parameter \( N \) is the number of simulations and should be at least at least 1024 for sufficient convergence of the methods. The symbol \( n \) is the number of error terms. |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| FAST            | Extended FAST   | Sobol           | Saltelli        | Brute force     | Analytically    |
| Indexes         | \( S_i \)       | \( S_i, S_{Ti} \) | \( S_i, S_{Ti} \) | \( S_i, S_{ij}, S_{Ti} \) | \( S_i, S_{ij...n}, S_{Ti} \) |
| Simulations     | \( O(n^2) \)    | \( n \times N \) | \( N \times (2 \times n + 2) \) | \( N \times (n + 2) \) | \( N^2 \times n \) |
It should be noted that variance decomposition analyses takes into account the shape and scale of the uncertainty distribution. If it is different from uniform, a pseudo-random number sampling approach should be taken. In such approach, the Sobol quasi-random variables should be transformed to the actual distribution. It employs numbers from Sobol quasi-random sets. The low discrepancy property is advantageous to the accuracy of the method. A uniform distribution is suitable from the perspective of interval analysis. The variance decomposition analysis indicates how the variance of the output is affected by the variance within the bounds of the input factors. Even though Bayesian and Probability analysis are only touched upon in this work, it should be noted they relate differently to variance decomposition analysis.

An overview of the model input and model form errors are given in Tables 5 & 6. It should be noted the in the above treatment, the uncertainty was modelled as epistemic, and it was quantified with bounds which limit all possible values. All uncertainty was treated as epistemic because the data originated from lab scale experiments and/or at conditions beyond the ones used in the work of Slegers, et, al, [9]. The only uncertainties that were modelled as aleatory were the ones associated with the isotropic light models. These models have been validated experimentally and backward uncertainty analysis has been performed in the study of Loutzenhiser, et al, [23]. Uniform probability distributions have been identified for the light intensity data error, the ground reflectivity, and the model form error. This structural information on the errors was incorporated in the probability bounds analysis in this work.
References


