23rd International Congress on Modelling and Simulation, Canberra, ACT, Australia, 1 to 6 December 2019
mssanz.org.au/modsim2019

An integrated optimisation functionality for Workspace

D. Potter [a], **L. Hetherton** [b], **D. Thomas** [b], **R. McNaughton** [a], **and D. Watkins** [b]

[a]*CSIRO Energy, PO Box 330, Newcastle, NSW 2300, Australia*
[b]*CSIRO Data61, PO Bag 10, Clayton South, VIC 3169, Australia*
Email: daniel.potter@csiro.au

Abstract: Scientific Workflow Systems (SWSs) allow scientific workflows, networks of operations connected with directed data flows, to be efficiently developed, executed and shared with others. By leveraging the built-in functionalities of SWSs, users can focus on creating domain specific content such as physical models. Workspace (Watkins et al. (2017)) is a scientific workflow and application development platform developed by the CSIRO. Users construct workflows using an intuitive graphical interface. Input and output variables can be viewed and modified using graphical interface objects called widgets which are implemented using the Qt framework [1]. A suite of plugins that provide operations, data types and widgets for tasks such as data analysis, file IO, 3D rendering and scripting are provided out-of-the-box, and users can create their own custom plugins. Wizards for creating new plugins, operations, data types and widgets are available, allowing the C++ source code skeletons to be automatically created, minimising development overhead.

Mathematical optimisation involves utilising numerical algorithms to find the best solution to one or more objective functions, and is a functionality of great utility across many scientific and engineering disciplines. Optimisation functionalities have previously been implemented in a number of SWSs. For example, optimisation frameworks have been developed for Taverna (Crick et al. (2009)), Kepler (Abramson et al. (2010)) and Nimrod/OK (Nguyen et al. (2017)). Prior to the current work, an integrated optimisation functionality has not been available in Workspace. Whilst parameter sweeps could be set up and controlled from within Workspace, applying algorithmic optimisation methods required either executing the workflow from an external optimisation tool, or the development of optimisation logic in a custom plugin as done in the present work. There is considerable value in developing such an integrated optimisation functionality for Workspace due to its unique capabilities as a SWS — in particular, the ability for Workspace to allow research codes to be transformed into professional standalone applications. For example, standalone applications have been developed using Workspace for bushfire spread prediction (Spark, Miller et al. (2015)), modelling of elite-level diving (Dive Mechanic, Cohen et al. (2017)), and solar power plant design (Heliosim, Potter et al. (2018)). The present work would allow optimisation to be incorporated into such standalone applications with minimal reformulation of existing plugins and workflows. Furthermore, Workspace supports remote execution and parallelisation of loops that could be used to allow the rapid execution of complex objective functions.

In previous work (Potter et al. (2018)), a plugin, workflow and standalone application called Heliosim has been developed using Workspace for the optimisation and simulation of central receiver concentrating solar thermal (CST) facilities. The central receiver CST involves the use of heliostats (mirrors that track the sun) to reflect solar radiation onto a central tower-mounted receiver located at the focal point, where the thermal energy can be stored and used for power generation or industrial processes. The core functionality of the Heliosim software is the simulation of heliostat optics using Monte Carlo ray tracing and the simulation of receiver heat transfer via a finite-volume model. This core functionality is used to form objective functions for the optimisation of the heliostat field layout, tower height and receiver geometry. The C++ code underpinning the integrated optimisation functionality presented in this paper was originally part of the numerics library used by the Heliosim software. This functionality has now been exposed as the Optimisation Loop operation in a newly developed Numerical Methods plugin for Workspace. The motivation for creating a devoted numerical methods plugin is the potential for its more general use beyond the current audience for the Heliosim plugin, which is mainly within the CST research group at CSIRO. In this paper, the software design of the Optimisation Loop operation is described, and an implementation of the Nelder-Mead simplex algorithm is validated. A case study is then presented where up to seven constrained variables describing the geometry of a beam-down CST reactor for producing hydrogen are simultaneously optimised to maximise annual energy output.

[1] https://www.qt.iotest

Keywords: *Scientific workflow, optimisation, solar energy*

1 SOFTWARE DESIGN

A diagram describing the Numerical Methods and Heliosim plugins in terms of the libraries they rely on for functionality and the operations they provide to the Workspace user is shown in Figure 1. The core functionality of both plugins is defined in C++ libraries that have been in development and usage at the CSIRO for the past 12 years. These C++ libraries link with a number of specialised third-party libraries. NLopt [2] provides non-linear optimisation algorithms to the Numerics library. Armadillo (Sanderson and Curtin (2016)) provides linear algebra solvers to the Numerics library. CoolProp (Bell et al. (2014)) provides thermodynamics models to the Heat transfer library. NVIDIA OptiX (Parker et al. (2010)) provides GPU accelerated ray tracing to the Optics library. The Workspace plugins expose various functionalities from the C++ libraries so that they can be made available on the Workspace canvas as operations.

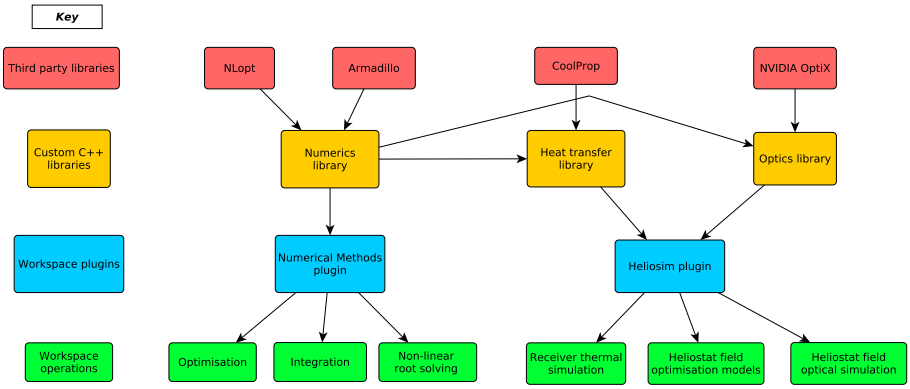
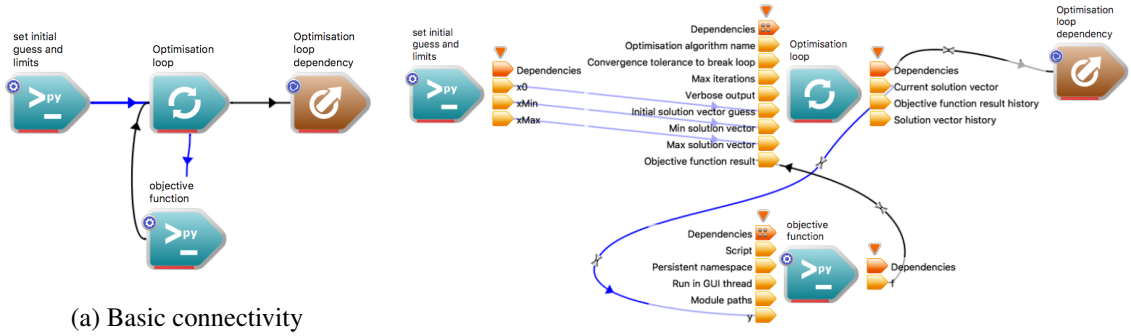


Figure 1. Structure of the Numerical Methods and Heliosim plugins for Workspace.

The layout of a very simple optimisation workflow (the Rosenbrock's parabolic valley validation case presented in § 1.1) on the Workspace canvas is shown in Figure 2. The actor labelled 'Optimisation loop' is the Optimisation Loop operation. In this simple example, the objective function is a Python script that takes a vector or list of floating point values ($\vec{y}$) as input and returns a floating point scalar quantity (f) as output. The Optimisation Loop operation has an output port labelled 'Current solution vector' that is connected to the $\vec{y}$ input port of the objective function, and the f output port of the objective function is in turn connected back to the Optimisation Loop operation via the 'Objective function result' input port. Thus a mechanism is formed whereby the Optimisation Loop operation is able to evaluate $f(\vec{y})$ for arbitrary $\vec{y}$ by setting the 'Current solution vector' output port and requesting the workflow execution manager to update the 'Objective function result' input port. From the perspective of the 'Optimisation loop' operation, the contents of the objective function workflow is arbitrary — it can be as simple as a Python script as in this example, or much more complex with networks of many operations including the full range of functionality provided by the built-in and third-party Workspace plugins (e.g. system calls, web requests, remote execution, parallelised loops, custom user created operations). From a C++ programming perspective, this is achieved by the class defining the Optimisation Loop operation using inheritance and abstraction to allow access to the optimisation functionalities defined in the core Numerics C++ library.

The Optimisation Loop operation has a number of additional input ports that allow the optimisation problem to be further defined, and additional output ports that allow the history of the optimisation iterations to be analysed. The additional input ports include the name of the optimisation algorithm, the convergence tolerance, the maximum iterations, and minimum, maximum and initial solution vectors. Currently the user can select between five different optimisation algorithms: 1) Nelder-Mead simplex, 2) differential evolution, 3) constrained optimisation by linear approximations (COBYLA), 4) evolutionary search (ESCH) and 5) controlled random search (CRS) with local mutation. The Nelder-Mead simplex and differential evolution algorithms are custom C++ implementations, whilst the COBYLA, ESCH and CRS algorithms are accessed by linking with the NLopt library. Although both the validation and case study presented in the present work utilise the

[2] http://github.com/stevengj/nloptttest



(a) Basic connectivity

(b) With input and output port connectivity detail

Figure 2. Layout of a very simple optimisation problem workflow on the Workspace canvas.

Nelder-Mead simplex method due to their suitability to gradient descent methods, many real-world optimisation problems have discontinuous objective function and thus the availability of evolutionary methods.

1.1 Nelder-Mead simplex algorithm

The simplex method for function minimisation described by Nelder and Mead (1965) has been implemented in the Numerics C++ library, which is then made available for use by the Optimisation Loop operation in Workspace. This is one of the most commonly utilised optimisation algorithms for constrained, multi-variate, single-objective optimisation problems. A simplex for an optimisation problem with n variables is a polytope with $n + 1$ vertices — the simplex for a 2-dimensional problem, for example, is a triangle. The simplex is progressively modified as the iterations proceed via replacing a vertex with a new vertex calculated by reflection, expansion, contraction and shrink operations. In the present implementation, the standard coefficients for each operation have been used (1, 2, $\frac{1}{2}$ and $\frac{1}{2}$ for reflection, expansion, contraction and shrink, respectively). Convergence is deemed to have occurred when the ratio between the current simplex diameter D and initial simplex diameter D_0 falls below a given tolerance ϵ, where the diameter of a simplex is computed as the maximum 'edge length' (i.e. L2 norm of the difference between any two vertices of the simplex).

Validation. Rosenbrock's parabolic valley has been considered as an objective function to test the performance the Nelder-Mead simplex implementation. It is a bivariate objective function with many local minima, and is therefore a strong test of an optimisation algorithms ability to find the actual global minimum. The equation describing Rosenbrock's parabolic valley is shown in Equation 1.

$$f(y_0, y_1) = 100(y_1 - y_0^2)^2 + (1 - y_0)^2 \tag{1}$$

A summary of the input parameters and results for optimisation of Rosenbrock's parabolic valley with the Nelder-Mead simplex algorithm using the Optimisation Loop operation in Workspace is shown in Table 1. The global minimum of $\vec{y} = (1, 1)$ was found to within a root mean square deviation (RMSD) of 1×10^{-6} in 119 iterations. Although this number of iterations to convergence is high, many other gradient based optimisation algorithms (such as the COBYLA algorithm from NLopt) fail to reach the actual global minimum.

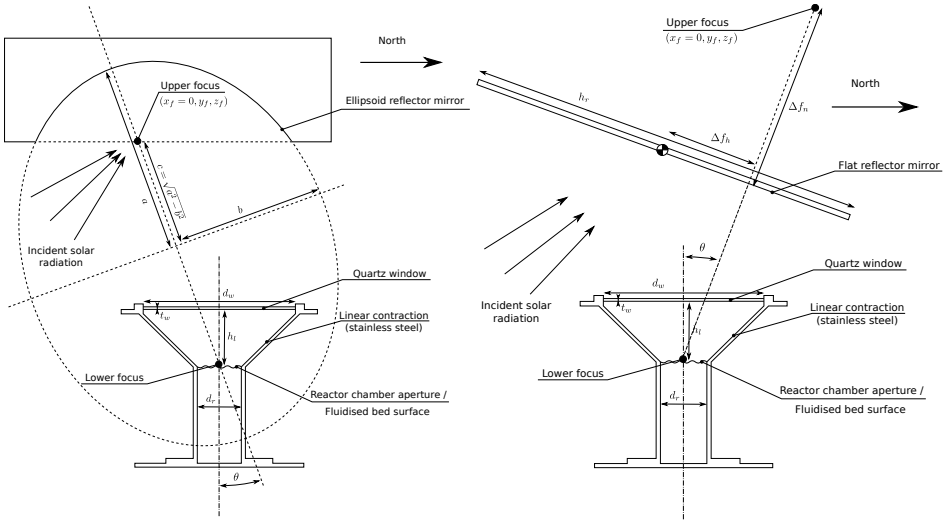
2 Case study: optimisation of a beam-down CST receiver

The CSIRO, in collaboration with Niigata University and the Institute of Applied Energy in Japan, have recently started a project investigating a solar powered thermochemical water splitting process for the production of hydrogen. As described by Kodama et al. (2019), the process includes the cyclic reduction and oxidation of a cerium oxide catalyst at temperatures of up to 1300 °C. To achieve these temperatures, solar energy is required to be concentrated to the equivalent energy of 1500 suns, and redirected (i.e. 'beam-down') by a mirrored reflector onto a horizontal fluidised bed in a reactor chamber. As part of the project, an experimental reflector and reactor will be installed and operated at CSIRO Energy in Newcastle, using the Solar Field 1 experimental facility. The facility consists of 165 heliostats, which are rectangular paraboloid mirrors that track

Table 1. Summary of input parameters and results for optimisation of Rosenbrock's parabolic valley with the Nelder-Mead simplex algorithm using the Optimisation Loop operation in Workspace.

Parameter	Value
Convergence tolerance, ϵ	1×10^{-6}
Initial solution, $\vec{y}_0$	(0.084, 1.6)
Converged solution, $\vec{y}$	(0.9999997, 0.9999993)
RMSD of converged and exact solutions	5.57×10^{-7}
Iterations to convergence	119

the sun to provide a beam of concentrated solar radiation onto the focal point, and a 20 m tall tower with a support ring that the reflector and reactor will be mounted onto. The overall system is physically constrained by the pre-existing layout of the heliostat field and tower design, resulting in the need for an optimised secondary reflector and reactor design to deliver adequate performance. Workspace and the optimisation loop operation have been used to assist the design of the reflector and reactor geometry. Two different reflector concepts are considered: 1) ellipsoid and 2) flat. Diagrams of both concepts are shown in Figure 3.



(a) Ellipsoid reflector (b) Flat reflector

Figure 3. Diagram of the beam down CST reactor concept with various reflector configurations.

2.1 Objective function

The objective function to be optimised is the annual thermal energy captured by the fluidised particle bed. The inputs into the objective function are the geometric parameters describing the reflector and reactor design (see Figure 3). Therefore, the optimisation problem is constrained, multi-variate and single-objective. For a given point in time, the thermal power captured by the fluidised particle bed ($q_{t,f}$) can be approximated as the absorbed solar radiation minus grey-body thermal radiation losses:

$$q_{t,f} = (1 - \rho_{s,f})\, q_{s_f} - \sigma \epsilon_{t,f} \pi \frac{d_r^2}{4} T_f^4 \tag{2}$$

where $\rho_{s,f}$, q_{s_f}, $\epsilon_{t,f}$ and T_f are the diffuse solar reflectance, incident solar irradiance, thermal emittance and temperature of the fluidised bed surface, respectively. Values of $\rho_{s,f} = 0.3$, $\epsilon_{t,f} = 0.85$ and $T_f = 1500$ K are assumed based on Ackermann and Steinfeld (2017) and Matsubara et al. (2014). The incident solar irradiance

can be expressed as the product of the optical efficiency to the fluidised bed surface and the solar direct normal irradiance (DNI). The optical efficiency is dependent on the reflector and reactor geometry and sun position, and can be simulated via Monte Carlo ray tracing using functionality from the Heliosim plugin. The solar DNI is obtained from historical measurements using pyrheliometers installed at CSIRO Newcastle. To reduce computational expense when integrating Equation 2 on an annual basis, the optical efficiency is interpolated from ray tracing simulations for a coarse grid of sun positions covering the annual sky traversal of the sun.

2.2 Workflow design

A screenshot of the Workspace GUI running the beam-down CST receiver optimisation workflow is shown in Figure 4. The 'objective function' actor is a nested workflow that computes the annual thermal output from the receiver. It is constructed using 30 operations, sourced from both the Heliosim plugin (e.g. GPU accelerated Monte-Carlo ray tracing), and operations from the built-in catalogue (e.g. Python scripts and system calls). A critical component of the objective function is the creation of the reflector and reactor surface meshes required for the ray tracing simulations. Due to the complex nature of the geometry, the surface meshes are created parametrically by system calls to SALOME[3], an open source computer aided design software that supports Python scripting. The workflow also incorporates post-processing of the optimisation iterations and converged solution to allow the results to be visually displayed via interactive 2D charts and 3D scenes.

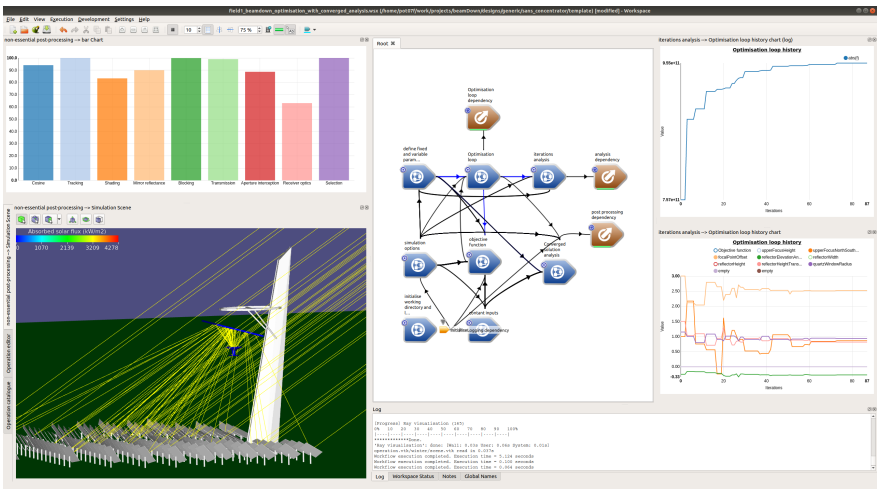


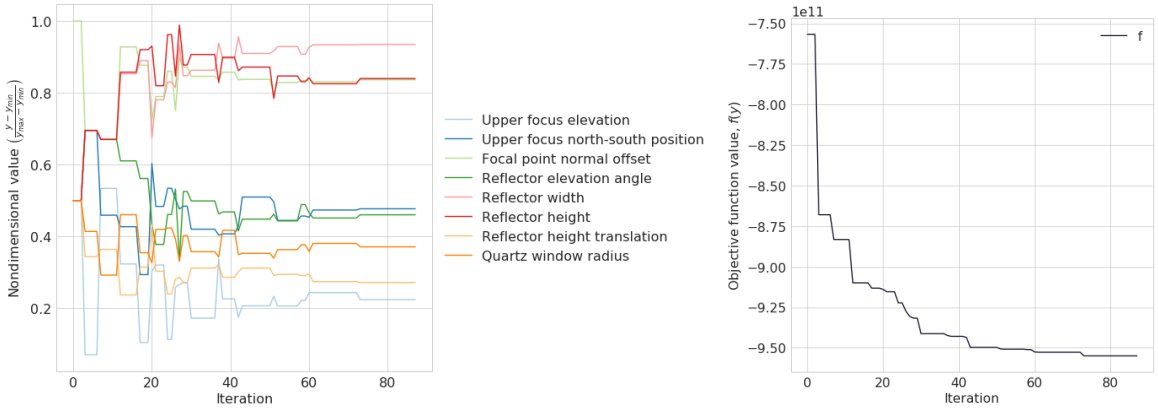
Figure 4. Workspace GUI running the beam-down CST receiver optimisation workflow.

2.3 Results

Initially, all geometric parameters (see Figure 3) except for the height of the linear contraction (h_l) and the reactor chamber diameter (d_r) were allowed to be optimised. The linear contraction height was fixed at 0.35 m as this is the minimum distance estimated to be required to keep the quartz window far enough away from the fluidised bed surface to ensure it remains clean during operation. The reactor chamber diameter was fixed at 0.4 m based on the requirement of an average incident solar irradiance of 1.6 MW/m^2 on the fluidised bed surface as described by Matsubara et al. (2014), and a desired design thermal capacity of at least 100 kW.

The optimisation convergence history for the flat reflector geometry is shown in Figure 5. All variables can be seen to have optimised values somewhere between the minimum and maximum values provided to the optimisation algorithm, indicating that these ranges did not artificially constrain the optimisation. The Nelder-Mead simplex algorithm was used due to its suitability to the constrained, multi-variate and single-objective nature of the problem, and the convergence tolerance was set to 0.01. The converged solution was reached in 83 iterations, requiring a 1.5 hour wall time on an Intel Core i7-6920HQ CPU and NVIDIA Quadro M2000M GPU. Reducing the convergence tolerance was found to substantially increase the number of iterations and wall time required to reach the converged solution, without significantly improving the solution.

[3]http://www.salome-platform.orgtest



(a) Nondimensionalised variable parameter values

(b) Objective function

Figure 5. Optimisation convergence history for the flat reflector geometry.

The optimised quartz window radii for both reflector cases was found to be very large, being almost 1 m for the flat reflector case. Due to the cost of quartz windows, such a size is not practical. A parametric study was therefore performed where the quartz window radius was held constant at various sizes and the other parameters re-optimised. The results of the parametric study are presented in Figure 6, where annual efficiency is plotted against the quartz window radius for both reflector geometries. The annual efficiency is calculated as the annual thermal energy absorbed by the fluidised bed divided by the annual solar DNI incident on the heliostat field aperture area. The ellipsoid reflector allows good performance for relatively small quartz window radii. The ellipsoid reflector, however, is significantly more complex to fabricate than the flat reflector. A flat reflector with a quartz window radius of 500 mm has been selected to proceed with to the design phase of the project as it represents a good balance of performance, simplicity and affordability.

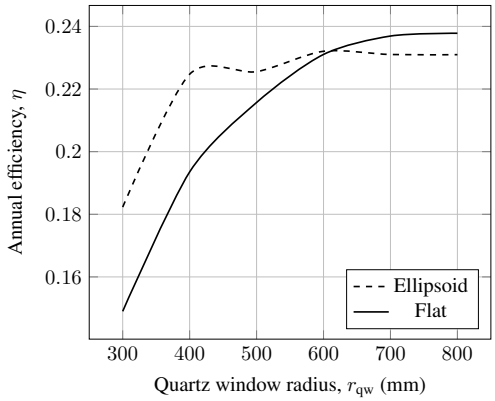


Figure 6. Annual efficiency versus quartz window radius for ellipsoid and flat beam-down reflectors.

3 Conclusions

An integrated optimisation functionality for Workspace has been validated and successfully applied in a case study to optimise up to seven constrained variables of a scalar objective function describing the performance of a beam-down CST reactor. The case study demonstrates some of the benefits of constructing an optimisation problem with Workspace and the Optimisation Loop operation. The objective function combines system calls to third party software, functionality from custom C++ libraries exposed via a Workspace plugin, and Python scripting. The convergence history and converged solution are then visualised via interactive 2D charts and 3D scenes, allowing immediate insight into the optimisation process. These capabilities, namely the ability to combine third party software, custom plugins and scripting languages, and data visualisation via interactive

2D charts and 3D scenes, are features available out-of-the-box within Workspace. Although an optimisation system with these features could be developed using languages such as Python or C++, the development process is much simpler with Workspace due to the intuitive graphical interface and large number of existing operations and widgets that can be leveraged by users. In this way Workspace and the Optimisation Loop operation allows the creation and modification of complex workflows incorporating optimisation by users without software development experience, as would be required with languages such as Python or C++. Work is currently underway to improve the scope and performance of the Optimisation Loop operation by allowing for multi-objective optimisation and parallelisation of the optimisation algorithms.

ACKNOWLEDGEMENT

This Activity received funding from ARENA as part of ARENA's Research and Development Program — Renewable Hydrogen for Export. The views expressed herein are not necessarily the views of the Australian Government, and the Australian Government does not accept responsibility for any information or advice contained herein.

REFERENCES

Abramson, D., B. Bethwaite, C. Enticott, S. Garic, T. Peachey, A. Michailova, and S. Amirriazi (2010). Embedding optimization in computational science workflows. *Journal of Computational Science 1*(1), 41–47.

Ackermann, S. and A. Steinfeld (2017). Spectral hemispherical reflectivity of nonstoichiometric cerium dioxide. *Solar Energy Materials and Solar Cells 159*, 167–171.

Bell, I. H., J. Wronski, S. Quoilin, and V. Lemort (2014). Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. *Industrial & Engineering Chemistry Research 53*(6), 2498–2508.

Cohen, R. C. Z., S. M. Harrison, P. W. Cleary, and M. Bolger (2017). Dive Mechanic : Bringing 3D virtual experimentation to elite level diving using the Workspace workflow engine. In J. Syme, G., Hatton MacDonald, D., Fulton, B. and Piantadosi (Ed.), *22nd International Congress on Modelling and Simulation*, pp. 431–437. Modelling and Simulation Society of Australia and New Zealand.

Crick, T., P. Dunning, H. Kim, and J. Padget (2009). Engineering design optimization using services and workflows. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367*(1898), 2741–2751.

Kodama, T., H. S. Cho, K. Inoue, T. Saito, S. Watanabe, N. Gokon, and S. Bellan (2019). Particles fluidized bed receiver/reactor with a beam-down solar concentrating optics: First performance test on two-step water splitting with ceria using a Miyazaki solar concentrating system. In *AIP Conference Proceedings*, Volume 2126, pp. 180011. AIP Publishing LLC.

Matsubara, K., Y. Kazuma, A. Sakurai, S. Suzuki, L. Soon-Jae, T. Kodama, N. Gokon, C. H. Seok, and K. Yoshida (2014). High-temperature fluidized receiver for concentrated solar radiation by a beam-down reflector system. *Energy Procedia 49*, 447–456.

Miller, C., J. Hilton, A. Sullivan, and M. Prakash (2015). Spark – a bushfire spread prediction tool. In R. Denzer, R. M. Argent, G. Schimak, and J. Hřebíček (Eds.), *Environmental Software Systems. Infrastructures, Services and Applications*, Cham, Switzerland, pp. 262–271. Springer International Publishing.

Nelder, J. A. and R. Mead (1965). A simplex method for function minimization. *The Computer Journal 7*(4), 308–313.

Nguyen, H. A., Z. van Iperen, S. Raghunath, D. Abramson, T. Kipouros, and S. Somasekharan (2017). Multi-objective optimisation in scientific workflow. *Procedia Computer Science 108*, 1443–1452.

Parker, S. G., J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich (2010). OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics 29*(4), 10.

Potter, D. F., J.-S. Kim, A. Khassapov, R. Pascual, L. Hetherton, and Z. Zhang (2018). Heliosim: an integrated model for the optimisation and simulation of central receiver CSP facilities. In *AIP Conference Proceedings*, Volume 2033, pp. 210011. AIP Publishing LLC.

Sanderson, C. and R. Curtin (2016). Armadillo: a template-based C++ library for linear algebra. *The Journal of Open Source Software 1*(2), 26.

Watkins, D., D. Thomas, L. Hetherton, M. Bolger, and P. W. Cleary (2017). Workspace – a Scientific Workflow System for enabling research impact. In G. Syme, D. Hatton MacDonald, B. Fulton, and J. Piantadosi (Eds.), *MODSIM2017, 22nd International Congress on Modelling and Simulation*, pp. 459–465. Modelling and Simulation Society of Australia and New Zealand.