A SUSTAINABLE COMPUTATIONAL DESIGN CONCEPT USING WEB SERVICE METHODS

Gopsill, James (1,2);
Hicks, Ben (1);
Schiffmann, Oliver (1);
McClenaghan, Adam (1)

1: University of Bristol, UK;
2: Centre for Modelling and Simulation, UK

ABSTRACT

Simulation is fundamental to many engineering design processes and powers the field of computational design. Simulation inherently consumes energy resulting in CO2 emissions that impact our environment. While one can source energy from renewable sources and use energy efficient hardware, efforts need to also be made in how we can use simulation in a sustainable manner.

This paper presents a sustainable simulation framework that borrows concepts from web services. The framework makes it easy for engineering firms to adopt and embed sustainable simulation practices thereby removing the burden from the designer tin thinking about how to design sustainably. An illustrative example reveals a 25% reduction in computational effort can be achieved by adopting the framework.

Keywords: Simulation, Computational design methods, Sustainability

Contact:
Gopsill, James
University of Bristol
United Kingdom
james.gopsill@bristol.ac.uk

Cite this article: Gopsill, J., Hicks, B., Schiffmann, O., McClenaghan, A. (2023) 'A Sustainable Computational Design Concept Using Web Service Methods', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.43

1 INTRODUCTION

Simulation is a fundamental, and increasingly necessary, element of Engineering Design. Simulations translate designers from the problem space, in part or full, to the solution space enabling them to reason about design parameters and make design decisions (Figure 1) (Kautz et al., 2018). Computational design studies, such as trade-offs, sustainability assessments, narrative-based simulation, design space reasoning, optimisation, exploration and optioneering provide methodological frameworks for how we can use simulation in the design process.

Simulation has also founded new design processes, such as Model-Based Systems Engineering (MBSE), Set-Based Design, Generative Design, Data-Driven Design and Digital Twins (al Handawi et al., 2020; LaSorda et al., 2018; McCormack et al., 2004; Muller et al., 2007; Parraguez and Maier, 2017; Tyflopoulos et al., 2018). MBSE, for example, has gained significant traction (25%) in the aerospace, defence, and whole systems design sectors. Those who have adopted the practice have observed a Return-on-Investment of 3.5:1 on average with a peak of 7:1 (Honour, 2010). The result has been designs that would have not been imaginable by designers alone. Argyres, (1999) report on the development of the B2-bomber highlighted that it could not have been achieved without simulation. In Formula 1, a sport that pushes Engineering Design to the very limited, there have been teams (Virgin F1) who have designed entire cars through simulation alone. Simulation has also been critical in the design of the 2022 rule changes that have enabled closer racing (Reynolds, 2019).

Today's computational design is an extensive digital ecosystem featuring:

1. A broad range of low to high-fidelity computational techniques capable of modelling all manner of design feature(s)[1];
2. Increasing computational capability to model designs within the permissible timespans[2];
3. Parametric simulation enabling design spaces to be represented;
4. Numerical methods to explore design spaces; and
5. Simulation Lifecycle Management (SLM) to manage the historic record of models and model runtimes.

However, the insights generated from simulation come with a cost both financial and environmental. The price for a single design point simulation in Autodesk Fusion360 is between $9-$18 (Autodesk, 2023). A design exploration study featuring multiple simulations and hundreds, if not thousands, of design points can quickly ramp into the thousands of dollars and a Digital Twin running simulations on a continual basis can result in a significant increase in operational expenditure. In terms of the environmental impact and continuing with the F1 2022 rules example, the computational design element of the project used 500,000GB and 16.5 million core hours resulting in an estimated CO2e of 13700kg[3] (MacAlpine, 2022). The Net Zero agenda and Global Energy Crises have put sustainability into sharp

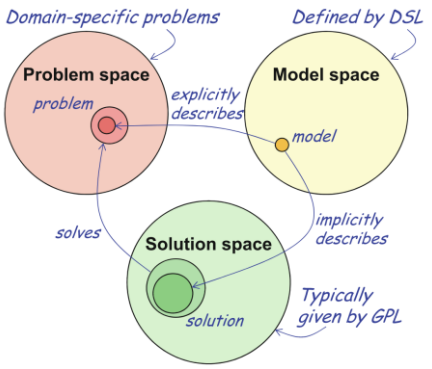


Figure 1. Moving between the problem, model, and solution spaces (Kautz et al., 2018).

[1] E.g., Finite Element Analysis (FEA), Computational Fluid Dynamics (CFD), Multi-Physics and Kinematics.
[2] E.g., GPU-acceleration, High-Performance Computing, Supercomputing, and the Cloud.
[3] 0.0008303kg CO2e/CPU AWS (us-west-1) CPU (https://www.climatiq.io/explorer/emission-factor/8e1fdcb8-0f37-429a-a275-4933d1e9f161)

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

focus, and we must be considerate of the carbon footprint of our simulations. Especially when our simulations are not only used in design but likely throughout the lifecycle as we digitally twin our products.

Physics-based simulation forms a major component of simulation in design and are notoriously computationally intensive (Selivanova, 2022). And while there are methods one could employ to reduce the computational effort, such as computationally performant simulation code, databases to store results and surrogate models, there is a lack of frameworks and guidance in providing a sustainable simulation environment. At best, methods are used ad-hoc requiring considerable expert knowledge, experience, and skills, and there are many cases where firm's re-compute design options, be it through repeating and/or overlapping design studies.

To address the issue of operating computational design sustainably, the paper contributes a concept borne from web service practices to introduce sustainability into computational design that is both automated and requires little to no changes to the operation of a design process. The paper continues with an illustrative example of the computational sustainability challenge and a reflection on existing approaches to computational sustainability (Section 2). The paper then introduces readers to the practices of OS-level virtualisation and API gateways that have been used extensively in serving computationally sustainable web services (Section 3). A concept for sustainable simulation is then presented (Section 4). Section 5 then applies the concept to Section 2's illustrative example. The results are presented in Section 6 and a discussion then ensues featuring the next steps and future research in Section 7. The paper then concludes with key findings (Section 8).

2 COMPUTATIONAL DESIGN SUSTAINABILITY CHALLENGES

To illustrate the computational sustainability challenge, let's examine a numerical solution to one-dimensional heat transfer. One-dimensional heat transfer is used in composites design (Fisher et al., 2023), where the conduction of heat is dominated by a ply's top and bottom surfaces, and the design of rods that are being heated at both ends and insulated along their length. The general problem is described in Figure 2 where the one-dimensional rod starts at a temperature, T_s, and is heated from the two ends by a temperature, T_b. A common feature of interest is how long it takes for a position along the length (i.e., the middle) to reach a desired temperature. The problem can be described by the one-

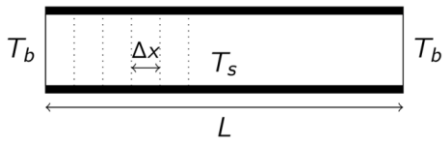


Figure 2. One-dimensional heat transfer.

dimensional transient heat conduction equation without heat generating sources:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) \tag{1}$$

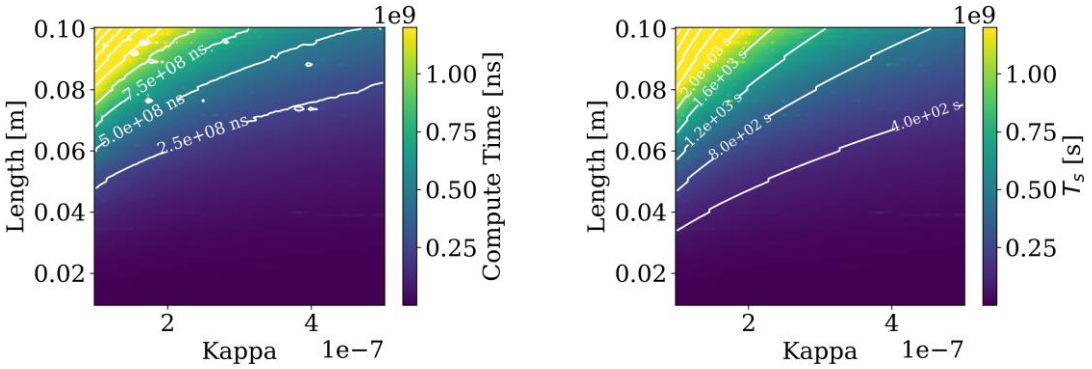
where ρ is the density, c_p is the heat capacity, k is the thermal conductivity, T is the temperature, x is the distance along the rod, and t is time.

Setting ρ, c_p, and k constant reduces the equation to:

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{2}$$

Differential equations are common across many simulations and can be solved numerically by discretising the continuous derivatives, $(\partial t, \partial x)$, using finite difference methods. For our example, we can approximate Equation 2 as follows:

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press



(a) Computational time *(b) Solution space.*

Figure 3. Computational effort in creating the design input/output response surface.

$$T_i^{n+1} = T_i^n + \kappa \Delta t \left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \right) \quad (3)$$

Where n is the time step and i is a position along the discretised domain. The challenges in computational sustainability lie in what designers wish to typically do in this space. That is to reason about the space the model represents as well as perform differing types of exploration that may lead to traversals over already 'trodden' ground.

2.1 Reasoning about the design space

Having defined the model space that takes us from the problem space to the solution space, lets perform a typical grid search across an arbitrary area of interest – 1e–2 m $\leq l \leq$ 1e–3 m and 1e–7 $\leq \kappa \leq$ 5e–7 with 100 increments along each dimension. As a designer may do to map the topology of the design space. Incrementing along each dimension in this manner results in the computation of 10,000 design options.

Total compute time (a proxy for effort) for the grid search was 44 mins. Figure 3a offers an insight into the computational effort across the domain and reveals it is non-uniform with a low kappa and increased length increasing the computational effort required to solve the problem. This was purposefully selected to demonstrate to aspects of a design space that contribute to computational effort. The first is increasing the length of the bar, which results in an increase in the mesh size resulting in more operations per timestep. The second is in decreasing kappa resulting in less heat being conducted across the bar per timestep resulting in more timesteps being computed until the bar to reach the desired temperature.

Figure 3b presents the solution space generated by the computation which suggests a polynomial relationship between the design inputs (length and kappa) and outputs (time to reach T_s). This type of result is often the case even for highly technical design cases, such as satellite design (Timperley et al., 2023). However, one must start with the computational expensive physics-based model to reveal the topology of the solution space. **Challenge 1** therefore lies in minimising the number of physics-based model evaluations to approximate the solution space with confidence.

2.2 Repeated excursions

While a grid search is a commonly applied design study to examine the topology of the solution space, other studies including design optimisation and exploration are also applied to identify optimal solutions. Figure 4 presents the options explored via different computational minimisation optimisation design studies – a refined grid search, Nelder-Mead and Powell methods (Nelder and Mead, 1965; Powell, 1964). It demonstrates how the different options explore the space with design options being repeatedly considered (computed). These studies are often performed independently by different design engineers deploying the simulation on different machines and at different points in the design process. The result is a high degree of design option re-evaluation. **Challenge 2** is in providing an environment that prevents siloed operation of simulations to preventing design options being re-computed.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

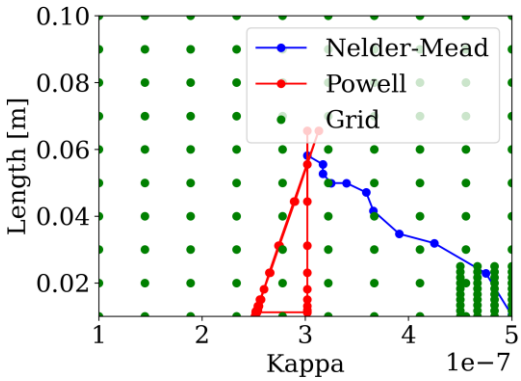


Figure 4. Computational design studies exploring the design space.

2.3 Existing methods to reduce computational effort

Researchers have sought to reduce the computational effort of exploring designs, with surrogate models being a main line of enquiry. Surrogate models, once trained, offer a lower computational overhead in evaluating a design space. See, for example, (Hanna et al., 2020; Jeon et al., 2019). While they offer long-term computational sustainability, they all require extensive evaluation of the design space to train the model. One could liken it to trading capital expenditure for operational expenditure. Thus, there is a trade-off to consider as well as expert knowledge to understand that extent of the design space that needs to be evaluated. There is also a concern to the surrogate models validity if the underlying physics-based model receives an update during the design process.

From this brief illustration of computational design and the methods currently being used to increase its sustainability, there remains opportunities to explore different concept to how we can make computational design more sustainable.

3 WEB SERVICES – AN AREA OF SUSTAINABLE COMPUTING TECHNIQUES

Web services power our ability to access information across the internet. Critical to their operation is the ability scale on-demand, provide information to the end-user in a timely manner, and operate as efficiently as possible due to their 24hr operation. Of particular interest in the generation of this paper's concept are OS-Level Virtualisation and API Gateways.

OS-Level Virtualisation is a $6.3bn market with a compound annual growth rate of 16% (Grand View Research, 2022). It is fundamental to the operation of Information & Technology (IT) services powering the internet, web services, Internet-of-Things (IoT), high-performance computing and cloud computing. Applications are built through a layering process that installs the required dependencies (OS, software) resulting in an image (Figure 5a-b). The image is then stored in a repository, which provides lifecycle management – versioning, access, and issue tracking4 (Figure 5c). Images are then

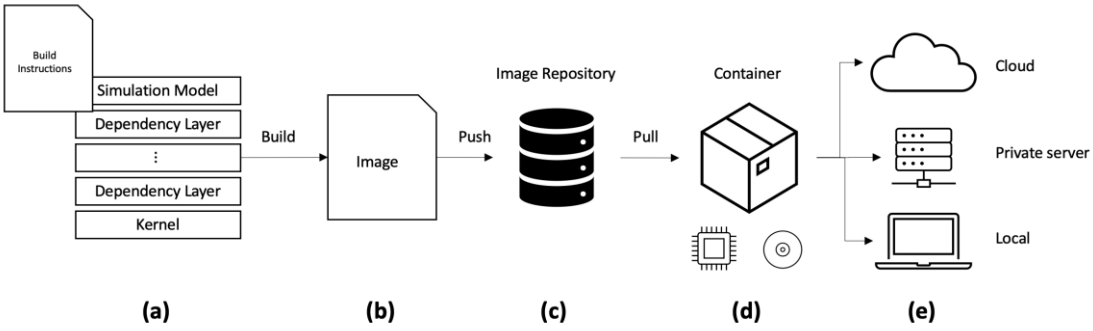


Figure 5. OS-level virtualisation process.

[4] One of the largest is DockerHub, which holds over one million images.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

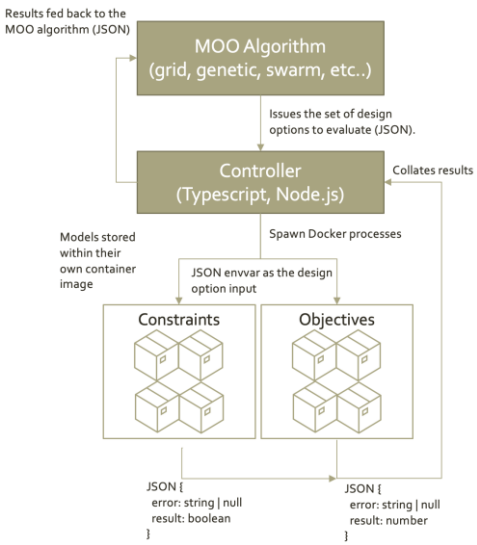


Figure 6. Docker Multi-Objective Optimisation (DoMOO) using OS-level virtualisation for MOO across simulations requiring different compute requirements.

pulled from their respective repository and instantiated in containers that assign the required compute resources to run the application (Figure 5d). The container can exist on a local machine, private server and/or cloud service (Figure 5e).

Orchestration tools, such as Docker Compose and Kubernetes, enable administrators to manage fleets of containers, establish virtual networks for communication, and connect to persistent storage. Containers are designed to be ephemeral enabling them to be destroyed and re-built without loss of information. This facilitates horizontal scaling through container duplication. Loads are then balanced across the duplicated services.

An example is the orchestration of a container featuring a database image, a container featuring a web application and a container featuring an API gateway. The API gateway routes traffic to users to access the webapp as well as access to backend services, such as the database. The capabilities of API gateways to route traffic to different backend services, handle authentication and authorization, cache responses, and ability to scale could be useful in enabling sustainable design computation.

While mainly used in web services, the application of OS-level virtualisation to support computational design has seen some proof-of-concept experimentation. Docker Multi-Objective Optimisation (DoMOO) is one such application (Figure 6) (Gopsill and Hicks, 2022). The application features several MOO algorithms and a Controller process. The Controller process orchestrates the spawning of simulations via Docker in accordance with the MOO algorithm and the design options it wishes to evaluate. It then waits for the simulations to complete and collates the results so they can be fed back into the MOO algorithm. DoMOO can readily scale with the resources made available to the docker instance that is deployed upon, offers the opportunity to parallelise operations, and can exit simulations early if another simulation returns and informs the process that the option is invalid (saving computational resource). This enables effective and efficient exploration of design spaces represented by multiple simulation models.

4 A CONCEPT FOR SUSTAINABLE SIMULATION USING OS-LEVEL VIRTUALISATION AND API GATEWAYS

The sustainable simulation concept for computational design utilising OS-level virtualisation and API gateway methods is shown in Figure 7. To start, we need a means to accept design options and the ability to return the solution metrics (e.g., the time taken to reach a temperature, C_l, C_d, or stress value). The framework achieves this by introducing an API gateway that provides information on the available simulations, authenticates, and authorises transactions, and routes design options to one or more simulations and one or more ‘caching’ mechanisms (e.g., database store, interpolator and/or Machine Learning algorithm).

Both the simulations and caching mechanisms exist in their own containers. The containers offer the resources they require to run and expose API endpoint’s that the gateway can route information to.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

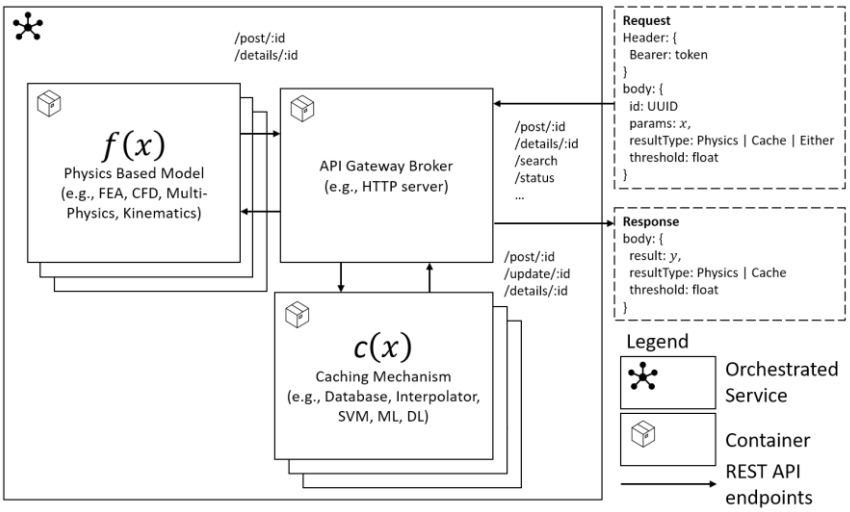


Figure 7. The sustainable simulation framework.

Associating an API endpoint to a simulation is achieved via a layer during the image build process. Instantiating these images results in a set of networked containers holding our API gateway, simulation model(s) and caching mechanism(s).

A client service (engineer, script, or process) sends requests to the API gateway. The requests require an auth token, simulation id, design option and any constraints on whether they wish the results to come from the physics-based simulation model or caching mechanism, and whether there is an unacceptable threshold by which they will accept the caching mechanism result. The API gateway will first authenticate and authorize the request before checking whether there is a caching mechanism assigned to the simulation model.

If a caching mechanism exists and its use is permitted, the gateway will route the traffic to the caching mechanism. The caching mechanism will then return a response along with a confidence value. If the confidence is above the desired threshold, the API gateway will send the response to the client mitigating the need to run a computationally expensive physics-based model.

If the confidence value is not above the threshold, the cache is the incorrect type or the request did not permit the use of a cache mechanism, the API gateway will pass the request to the simulation model to compute our 'ground-truth' result. The response is then passed back to the API gateway, which returns it to the client as well as passing it to the caching mechanism to update its internal model.

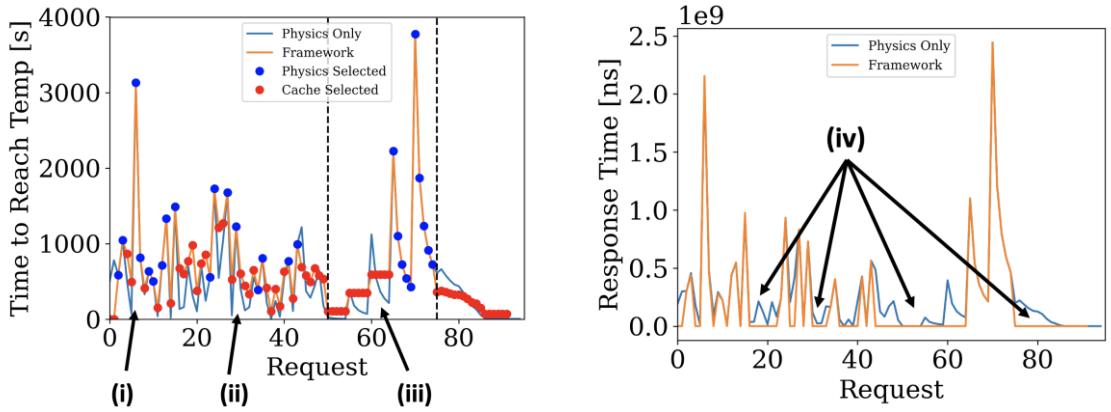
The concept meets the two challenges by utilising a caching mechanism to learn and override the use of the physics-based model in offering solutions to designers. The designers do not need to change their behaviour and/or design activity they wish to perform. They simply query the service as if it was a physics-based simulation (Challenge 1). Providing a single API for all engineers within a firm/project to access and utilise simulations mitigates re-computation by taking advantage of the caching mechanism to report previously computed results (Challenge 2).

5 ILLUSTRATIVE EXAMPLE

To illustrate the potential, the concept was used to investigate the design space of the design problem described in Section 2. Our solution to the one-dimensional heat transfer problem was placed in an image with an API access point for sending requests based on bar length, l, and thermal diffusivity, κ. Another image was created for the caching mechanism that featured an API endpoint that connected to an online learning Bayesian linear regression model with a polynomial extender feature extraction pipeline. Online learning models are a subset of Machine Learning (ML) models that can be updated as new values are generated making them a suitable candidate for a caching mechanism. The regression model returns a prediction as well as a confidence score.

The two models were connected through an API gateway. Requests from computational design studies would be sent to the API gateway and then passed to the caching mechanism followed by the physics-based model if the desired confidence was not met. The combined service was orchestrated ("spun-up") using Docker Compose.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press



(a) Comparison of values returned

(b) Speed of response

Figure 8. Comparison of responses from the sustainable simulation framework and physics-based model only operation.

Table 1. Comparing simulation operations.

Item	Physics Only	Framework
# Physics	92	36
# Caching Mechanism	n/a	56
Total Time (ns)	2.434e+10	1.815e+10 (75%)

The service then received three computational design studies: (1) 50 random evaluation of design options; (2) A 5x5 grid search; and, (3) Nelder-Mead minimisation optimisation.

These were performed in sequence and mimic some of the operations that a design team might perform over the course of a design project. The threshold for accepting the caching mechanisms result was set to 99.9%. The computational time, a proxy of computational effort, and number of physic-based vs. caching mechanism responses were recorded. A physics-based only simulation of the design studies was performed to form the baseline computational effort for comparison. The questions posed were:

1. How and whether the caching mechanism could 'take-over' the physics-based simulation in providing results for the computational design studies?
2. How, if at all, did it effect the results of the computational design studies? and,
3. What, if any, reduction in computation effort was achieved?

6 RESULTS

Figure 8 and Table 1 present the results from the study. Figure 8a shows how the caching mechanism starts to supersede the physics-based model during the request sequence. In this case, we see that only 5 options were required before the caching mechanism felt confident enough in providing its own predictions (8ai). Thus, we can start to gain a computational benefit surprisingly early in a simulation's lifecycle. We then see the service flipping between the two showing the cache is 'learning' and is confident in some areas of the design space but not others (8aii). Figure 8a also shows that the results do not match the physics model entirely (8aiii). Figure 8b shows the gains in response time between the Table 1 shows the computational effort of the existing and new approach and reveals a 25% reduction in computational effort/time was achieved. The impact is two-fold in reducing the carbon-footprint of computational design studies as well as speeding up the design process. Table 1 also shows that the caching mechanism responded to 56 out of the 92 requests for evaluation.

7 DISCUSSION

The paper has demonstrated that the concept built on OS-level virtualisation and API gateway technologies can have a significant positive impact on the environmental sustainability of computational design. The concept also removes the burden for a design engineer to consider computational sustainability, allowing them to focus on the design process. By way of indicative example, the study showed in a 25% reduction in computational effort.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

Expert knowledge was required in deciding the caching mechanism model and development of best practice and guidelines for cache mechanism selection that would support adoption of the concept. The concept also permits multiple caching mechanisms to be implemented alongside a simulation model thereby providing a means to trial cache mechanisms in parallel. Periodic evaluation could then nominate the mechanisms that should continue to operate and others that should be retired.

Erroneous results did pass through to the end-user where the cache 'felt' confident in its results, which may be of concern for mission-critical applications. However, the degree of error exhibited in the study may be acceptable in early conceptual design where a broader and less accurate understanding of the design space is often required.

The reported computational saving is illustrative of the concept's potential. It has yet to be optimised and it is likely that further gains could be made. For the example problem, further efforts could be made to optimise the physic-based model using a lower-level language and/or highly tuned scientific software packages. Different caching mechanisms and experimenting with different computational design sequences may reveal sequences that yield 'quicker' learning.

The illustrative example also provides a pragmatic view of where one might start in modelling a particular element of physics for a design problem. An affordance of the concept is that simulations can be updated and 'pushed' to the service through image repositories. Methods can be implemented that monitor for new images and 'hot reload' containers with updated images. This is performed automatically with little to no impact on end-users using the service.

There is also the environmental impact of having a service that is always 'on' to consider. OS-level virtualised services can scale up and down based on demand and there will always be an environmental footprint of operating the service. In a simulation intensive design process, it is likely that this "standby" energy use will be less than a few physics-based simulations but requires confirmation.

Research into adoption also needs to be performed. The concept affords for the end-user to specify whether they wish the physics or cache-based response and the threshold they are willing to accept the cache-based response. It would be interesting for the community to run a series of design studies offering simulations through the concept and investigating the choices designers make in deciding the form of response. One hypothesis is that designers in early-stage design may be more accepting of cache mechanisms, but as one continues to detailed design, designers may wish the certainty that comes from the physics-based simulation.

The last discussion point considers the fact that the caching mechanism is, in effect, providing simulation with a memory. This is a fundamental capability if one wishes to pursue and apply innovations and research from cognitive computing. It would be interesting to see how the concept can further support the application of cognitive computing in simulation, making simulations self-aware, learn, reason, communicate, and co-operate. Could we get to a position where simulations could explore the design space together and potentially eliminate the need for engineering designers to design computational design studies?

8 CONCLUSION

Simulation is fundamental to modern Engineering Design and is increasingly being used across the design process. The extended use of simulation has a knock-on environmental impact due to the energy simulations consume, with physics-based models being notoriously computationally intensive. They can also take considerable time to run reducing the number of iterations a design team can perform within a given period. To overcome these challenges, this paper has proposed a world first concept that exploits web service approaches to introduce caching mechanisms to simulation models. An illustrative example shows that a 25% reduction in computational effort can be achieved for a typical combination of computational design study featuring 100 design options evaluation. It is likely this value would increase as more computational design studies are run.

Expert knowledge is still required in determining a suitable caching mechanism for a simulation and best practice guidelines and decision-trees need to be generated. The framework also opens a promising opportunity to add further intelligence to simulation. The caching mechanism is, in effect, the beginnings of simulations with memory. This is one of the fundamental steppingstones towards cognitive computing. The framework could be extended to feature other cognitive features, such as self-awareness, self-learning, reasoning, communication, and co-operation across simulations, providing an exciting transformative means by which we manage, orchestrate, and perform computational design.

https://doi.org/10.1017/pds.2023.43 Published online by Cambridge University Press

ACKNOWLEDGEMENTS

The work has been undertaken as part of the Engineering and Physical Sciences Research Council (EPSRC) grants – EP/R032696/1 and EP/V05113X/1.

REFERENCES

al Handawi, K., Andersson, P., Panarotto, M., Isaksson, O., Kokkolaras, M., 2020. Scalable Set-Based Design Optimization and Remanufacturing for Meeting Changing Requirements. Journal of Mechanical Design 143. https://doi.org/10.1115/1.4047908

Argyres, N.S., 1999. The Impact of Information Technology on Coordination: Evidence from the B-2 "Stealth" Bomber. Organization Science 10, 162–180. https://doi.org/10.1287/orsc.10.2.162

Autodesk. 2023. Tokens for Fusion 360 Simulation Studies. [WWW Document] URL https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-97268379-A0C4-4426-8929-2FB3F3FB439C (accessed 16.2.23).

Fisher, A., Levy, A., Kratz, J., 2023. Effects of heat transfer coefficient variations on composite curing. J Compos Mater 57, 363–376. https://doi.org/10.1177/00219983221145506

Gopsill, J., Hicks, B., 2022. Through-Lifecycle Whole-Design Optimisation Using Software Deployment Toolchains. Proceedings of the Design Society 2, 1855–1864. https://doi.org/10.1017/pds.2022.188

Grand View Research, 2022. Data Center Virtualization Market Size, Share & Trends Analysis Report By Component, By Organization Size, By End Use Industry, By Region, And Segment Forecasts, 2022 - 2030 [WWW Document]. Grand View Research. URL https://www.grandviewresearch.com/industry-analysis/data-center-virtualization-market-report (accessed 11.2.22).

Hanna, B.N., Dinh, N.T., Youngblood, R.W., Bolotnov, I.A., 2020. Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD). Progress in Nuclear Energy 118, 103140. https://doi.org/https://doi.org/10.1016/j.pnucene.2019.103140

Honour, E., 2010. 11.4.2 Systems Engineering Return on Investment. INCOSE International Symposium 20, 1422–1439. https://doi.org/https://doi.org/10.1002/j.2334-5837.2010.tb01150.x

Jeon, K., Yang, S., Kang, D., Na, J., Lee, W.B., 2019. Development of surrogate model using CFD and deep neural networks to optimize gas detector layout. Korean Journal of Chemical Engineering 36, 325–332. https://doi.org/10.1007/s11814-018-0204-8

Kautz, O., Roth, A., Rumpe, B., 2018. Achievements, Failures, and the Future of Model-Based Software Engineering, in: Gruhn Volker and Striemer, R. (Ed.), The Essence of Software Engineering. Springer International Publishing, Cham, pp. 221–236. https://doi.org/10.1007/978-3-319-73897-0_13

LaSorda, M., Borky, J.M., Sega, R.M., 2018. Model-based architecture and programmatic optimization for satellite system-of-systems architectures. Systems Engineering 21, 372–387. https://doi.org/https://doi.org/10.1002/sys.21444

MacAlpine, J., 2022. Driven by Simulation, a Regulations Overhaul Promises a New Era of Racing in Formula 1 [WWW Document]. engineering.com. URL https://www.engineering.com/story/driven-by-simulation-a-regulations-overhaul-promises-a-new-era-of-racing-in-formula-1 (accessed 11.28.22).

McCormack, J., Dorin, A., Innocent, T., 2004. Generative Design: A Paradigm for Design Research.

Muller, D., Reichert, M., Herbst, J., Poppa, F., 2007. Data-Driven Design of Engineering Processes with COREPROModeler, in: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007). pp. 376–378. https://doi.org/10.1109/WETICE.2007.4407191

Nelder, J.A., Mead, R., 1965. A Simplex Method for Function Minimization. Comput J 7, 308–313. https://doi.org/10.1093/comjnl/7.4.308

Parraguez, P., Maier, A., 2017. Data-driven engineering design research: opportunities using open data, in: DS 87-7 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 7: Design Theory and Research Methodology, Vancouver, Canada, 21-25.08. 2017. pp. 41–50.

Powell, M.J.D., 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comput J 7, 155–162. https://doi.org/10.1093/comjnl/7.2.155

Reynolds, J., 2019. 2021 F1 RULES: The Key Changes Explained [WWW Document]. Formula One. URL https://www.formula1.com/en/latest/article.2021-f1-rules-the-key-changes-explained.2dCtCkxNofk20K1B4rJwTk.html (accessed 11.28.22).

Selivanova, S., 2022. Computational Complexity of Classical Solutions of Partial Differential Equations. pp. 299–312. https://doi.org/10.1007/978-3-031-08740-0_25

Timperley, L., Berthoud, L., Snider, C., Tryfonas, T., Prezzavento, A., Plamer, K., 2023. Towards Improving the Design Space Exploration Process Using Generative Design With MBSE, in: 44th IEEE Aerospace Conference.

Tyflopoulos, E., Tollnes, F.D., Steinert, M., Olsen, A., 2018. State of the art of generative design and topology optimization and potential research needs. DS 91: Proceedings of NordDesign 2018, Linköping, Sweden, 14th-17th August 2018.