Benchmark of Optimizers for Optimal Dispatch of Nuclear Hybrid Energy Systems

Daniel Hill, Nicholas Cooper, Qinyu Zhu
ME 575 - Dr. Andrew Ning
Brigham Young University
Project report

***Abstract*— Over the last decade, more intermittent renewable power has been introduced to the power grid, causing instabilities in the power grid and fluctuations in the price of electricity. Nuclear-renewable hybrid energy systems (NHES) have been proposed as a potential solution. These systems can take advantage of flexible loads and energy storage to more flexibly meet the demands of highly-dynamic power grids. Economical operation of these systems requires careful optimization. In this project, a simple NHES model is developed to capture the dynamic operational behavior of these systems. The operation of this model is then optimized to minimize economic costs. A series of optimizers and algorithms, including but not limited to SNOPT, SLSQP, Nelder-Mead, and Fmincon, are then benchmarked for the optimization process. The resulting strengths and weaknesses of each solver for this problem domain are then compared and reported.**

I. Background

Modern society depends heavily on access to vast quantities of electrical energy, and there is a strong interest in providing that energy in clean and environmentally-responsible ways. As a result, the amount of power generation coming from intermittent renewable sources, such as wind and solar, has increased dramatically in recent years [25]. These intermittent renewable sources have both predictable and unpredictable variations, which often pose significant challenges to the stability of the modern power grid [7].

Nuclear-renewable hybrid energy systems (NHES) are seen as a promising method of reducing the environmental impact of the power grid while integrating valuable renewable energy sources and maintaining grid stability [10]. NHES are systems that combine nuclear and renewable power sources with one or more loads or energy storage methods to enable highly flexible power generation to the grid. Many varying models of NHES have been developed in the literature with a broad array of configurations and involved components [23], [10], [31].

One set of related NHES systems are those developed by Garcia et al. [10]. These two systems both utilize a small modular reactor (SMR) with a renewable energy source and a flexible load to provide economical, clean power generation capable of scaling rapidly to meet power demand. One of the two systems utilized wind energy and a natural gas reforming plant to convert natural gas to gasoline during periods of low power demand. The other system used solar energy together with a water desalination plant, operating as a flexible electrical load, to provide clean water during off-peak periods. Both of these systems demonstrated economical power production capable of reliably facilitating higher penetration of renewable energy sources.

Other examples of NHES include that of Zhao et al. [31], where a thermal energy storage unit was combined with an SMR and concentrating solar plant, and Papaionnou et al. [23], where both flexible thermal and electrical loads were combined with an SMR and a wind farm.

Overall, the optimization of NHES provides a unique class of problems. Current literature examples provide some insight into how these systems may be optimized. Du et al. [8] found that hybrid energy systems, not necessarily NHES, are often very noisy and highly multi-modal. This motivated their use of the Nelder-Mead [21] gradient-free algorithm in their generalized optimization framework for hybrid energy systems. Some authors,

such as Augutis, Martišauskas and Krikštolaitis[3], and Ioannou et al. [15], used custom algorithms using stochastic sampling for unit sizing, but did not attempt to optimize dispatch. Other authors, such as Baker et al. [4], used a nested-loop approach. This approach, developed by multiple authors at Idaho National Laboratories [1], [2], [9], [26], [28], features inner and outer optimization loops, where unit capacity sizing is handled in the outer loop with a stochastic optimizer and dispatch optimization is managed on the inner loop with generic or custom methods.

The focus of this work is to determine which optimizers are most suited to handling the dispatch optimization problem for NHES. Once the best method is determined, it can be used together with the platform previously developed by Idaho National Laboratories to perform effective optimization of NHES and thereby help to enable environmentally-responsible, robust power generation.

II. NHES Model

To facilitate benchmarking of the various solvers for dispatch optimization of NHES, a sample NHES model was developed. This model features a nuclear reactor, variable renewable generation sources, and a thermal energy storage unit, as shown in Figure 1. At every point in time, the NHES must match the dynamic load required by the power grid and must stay within the upper and lower temperature bounds set on the thermal energy storage unit.

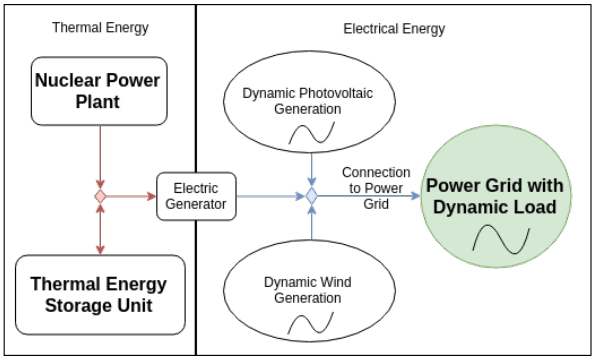

Fig. 1. Sample NHES configuration used for benchmarking

The change in thermal energy storage unit temperature is accounted for by a single differential equation given in Equation 1.

$$\frac{dT}{dt} = (gen - load_{net})\frac{mass_{salt}}{C_{p,salt}} \tag{1}$$

Detailed parameters for the model are given in Table I.

TABLE I
PARAMETERS USED THE SAMPLE NHES MODEL

Parameter	Value	Units
$cost_{nuclear}$	0.021	\$/MWh
$cost_{ramp}$	1	\$/MW/h
$cost_{blackout}$	$1 * 10^{10}$	\$/MW hr undersupply
$cost_{oversupply}$	$1 * 10^{10}$	\$/MW hr overage
$cost_{overramp}$	$1 * 10^{10}$	\$/MW hr overramp
$ramp_{max}$	2000	MW/hr
$C_{p,salt}$	1530	J/kg K
T_{min}	300	K
T_{max}	700	K
$mass_{salt}$	6e8	kg
capacity	54000	MW
T_0	350	K

Real power grid data for Oct 1 2019, taken from the Texas ERCOT power grid [22], was used for running this model. This allows capturing realistic net load variations over the course of a typical day on the scale required by a modern power grid.

In meeting this load, it was assumed that the entire net load of the power grid would be met by the sample NHES. While the results and discussion refer to this NHES as if it were a single unit, it would more accurately be a large number of distributed smaller NHES operating on the grid. Because of this, many system parameters, such as ramp rate, are not realistic for a single NHES, but rather reflect the combined effects of a number of cooperating systems. Economic costs associated with this model are also not intended to be accurate to the real system, but rather simply act as motivators for the optimization.

The objective of the optimization is to minimize the daily operational cost of the NHES, by adjusting the nuclear generation at each time point (gen). There are two different constraints in this problem. One is the temperature of the thermal storage unit at each time point (T), which should be between 300 K to 700 K at each time point. The other is the ramping rate of the nuclear generation ($\frac{dgen}{dt}$), which should not exceed $ramp_{max}$. Since we are benchmarking different optimizers and not

all the optimizers can directly handle constraints, both constrained and penalized formulations of the optimization problem are presented. In the penalized formulation, the constraints are directly added to the objective function through an external penalty method.

Optimization algorithms capable of handling constraints will be benchmarked against both the constrained and penalized formulations, while unconstrained optimizers will be benchmarked against only the penalized formulation.

A. Constrained Formulation

In the constrained formulation, the objective function consists of the cost of energy generation and the cost of ramping up and down the reactor core. The constraints are then implemented as normal non-linear constraints as shown in Equation 2. The evaluations of the ramping rate and the temperature profile are independent of the constrained model, and are evaluated using separate functions to reduce computational complexity.

$$minimize \quad cost_{ramp}\frac{dgen}{dt} + cost_{nuclear}\ gen \tag{2}$$

$$subject\ to \quad T_{min} < T < T_{max} \tag{3}$$

$$\frac{dgen}{dt} < ramp_{max} \tag{4}$$

$$gen \leq capacity \tag{5}$$

B. Penalized Formulation

In the penalized formulation, the constraints are added to the objective function using an external penalty method, as shown in Equation 6. Due to this interconnected nature of the constraints and the objective function, this formulation requires evaluating the constraints with every call to the objective function.

$$\begin{aligned} minimize \quad & cost_{ramp}\frac{dgen}{dt} + cost_{nuclear}\ gen \\ & + cost_{blackout}\ max(0, T - T_{max}) \\ & + cost_{oversupply}\ max(0, T_{min} - T) \\ & + cost_{overramp} \\ & \ max(0, \frac{dgen}{dt} - ramp_{max}) \end{aligned} \tag{6}$$

III. Results and Discussion

A number of gradient-based and gradient-free optimizers were used to optimize the dispatch of the sample NHES model described in Section II. The algorithms that support constrained optimization were benchmarked against both the constrained and the penalized formulations of the model, while unconstrained optimizers were only benchmarked against the penalized formulation.

A number of key metrics were used for the benchmarking process. These consisted of the final value of the objective function itself, the number of function calls required for convergence, and the feasibility of the solution. The results for each algorithm are listed under the respective headings below and are summarized in Table II.

A. SLSQP

Both the constrained and penalty-based formulations of the NHES model were solved using the Sequential Least Squares Programming (SLSQP) [16] method from the SciPy [29] Python package. SLSQP is a well-known gradient-based method for sequential quadratic programming (SQP) problems. Overall, this algorithm performed quite poorly in both cases.

SLSQP required 8105 function calls to converge to a final objective function value of \$26499.0. The resulting solution, shown in Figure 2, is feasible and quite close to the true optimal solution. It is important to note that the temperature constraints were very slightly $(7.4 * 10^{-7})$ violated in this solution. This is likely due to how the algorithm deals with the constraints.

For the penalized formulation, SLSQP required 4794 function calls to converge to a final objective function value of $\$9.6 * 10^{12}$. This solution, shown in Figure 3, is not feasible, clearly violating the lower temperature constraint. It did, however, stay within the ramp constraints. For this solution, it is important to notice that the solution is quite smooth, indicating that the solver can utilize the correlated nature of the inputs.

B. Nelder-Mead

The gradient-free Nelder-Mead method [21] was implemented using the SciPy [29] Python package. It is incapable of handling bounds and constraints,

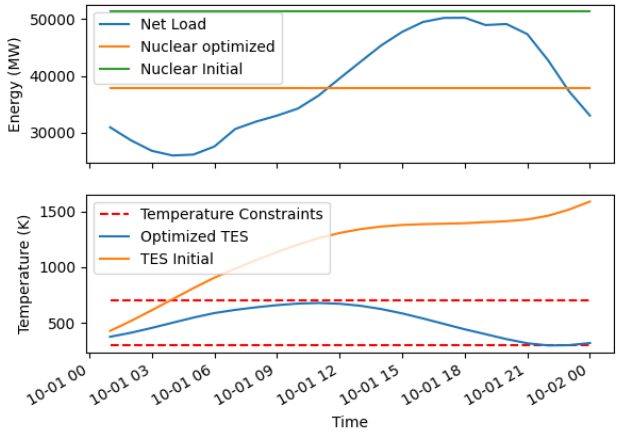

Fig. 2. SLSQP Optimal results for constrained formulation

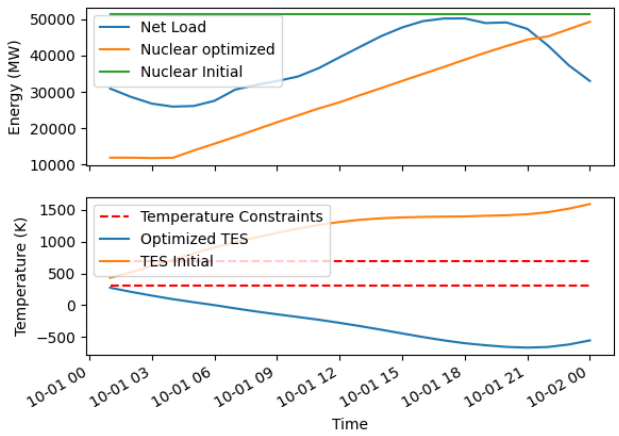

Fig. 3. SLSQP Optimal results for penalized formulation

so only the penalized formulation of the model could be used. The Nelder-Mead algorithm required 13135 function calls to converge to a final objective function value of $\$6.7 * 10^{13}$. The final solution, shown in Figure 4, shows that some minimal progress was being made, but the optimizer exited before converging closer to the true optimum. Ultimately, the large number of input variables combined with the very tight feasible solution space make the gradient-free Nelder-Mead algorithm a poor choice for this system.

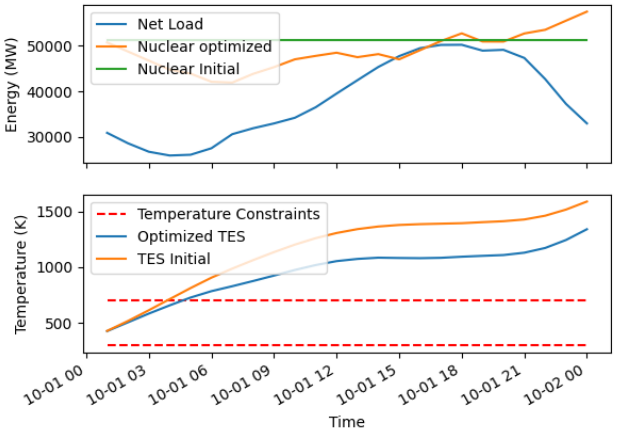

Fig. 4. Nelder-Mead optimization results

C. SNOPT

Both the constrained and penalized formulations were solved using SNOPT [11], [12] and the pyOptSparse [24] Python package. While SNOPT supports gradients, it also has internal finite-differencing methods for approximating gradients if no gradients are given. In both formulations, the SNOPT internal methods were used for approximating the gradients.

The SNOPT algorithm performed quite well on both the constrained and penalized formulations. The constrained optimization required 4248 function calls to converge to an objective function value of \$19077.5. This is nearly the same as the solution produced by *fmincon* while using approximately two-thirds of the function calls. The final solution, shown in Figure 5, demonstrates a very flat solution with no ramping and effective use of the thermal energy storage unit.

For the penalized formulation, SNOPT required 6084 function calls to converge to an objective function value of \$19146.9. This result is somewhat more computationally demanding that constrained counterpart, but achieves nearly the same level of accuracy. The optimal solution is visually identical to that of the constrained optimization shown in Figure 5, so the plot is omitted for brevity.

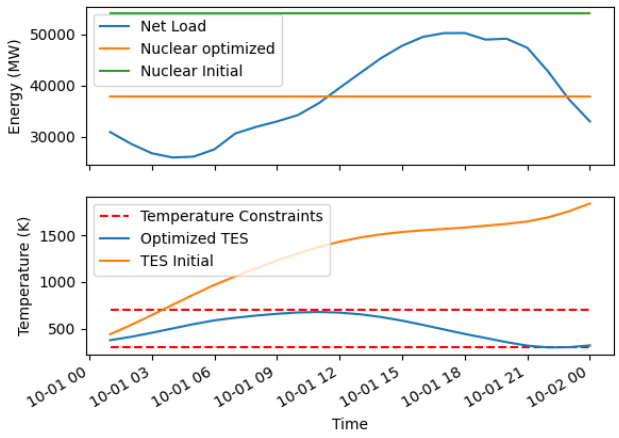

Fig. 5. Optimal results from SNOPT constrained method

D. GEKKO

GEKKO [5], a Python package for dynamic optimization, contains options for interfacing with IPOPT [30] - an interior point constrained optimizer, and other optimizers either widely used or unique to GEKKO. There were difficulties formulating the NHES problem to work well with these optimizers through GEKKO, thus no solutions can be presented at this time. However, some initial tests varying the initial guess and generation parameters allowed GEKKO to solve the problem with three of its default optimizers, including IPOPT. Time did not allow further analysis, but will be beneficial to explore this behavior in future research.

E. Pyomo

Pyomo [13], [14] is a Python package designed for modeling and interfacing with a substantial list of optimizers. A major benefit of using modeling packages such as Pyomo and GEKKO [5] is their ability to use automatic differentiation in optimization. Unfortunately, similar to attempts made to use the GEKKO package, there were difficulties formulating the NHES problem so that Pyomo could correctly pass it to an optimizer. Future use of modeling software such as Pyomo and GEKKO will likely yield promising results.

F. Genetic Algorithms

Part of the motivation for the NHES problem is to explore the load-following capability of the system. Gradient-based optimizers yield solutions that are near steady state (for the specific parameters used in this experiment). The interest in gradient-free optimizers and genetic algorithms (GAs) in particular was to see if the increased variation of design variables could find a global solution with more obvious load-following behavior. Two genetic algorithms were used:

- SciPy's *differential_evolution* [29], [27]
- *geneticOpt* - a simple implementation for constrained optimization using the feasibility rule [17]

The results from SciPy's algorithm can be seen in Figure 7 for the constrained objective, and Figure 6 for the penalized objective. It is obvious that the algorithm performs much better for the penalized model than for the constrained model. As the model is time-based, we are looking for smooth curves as an indication of whether the optimizer can tell that each point influences the next point. SciPy's algorithm with the constrained model clearly does not gain this sense through the optimization, while the same algorithm with the penalized model does. The optimized cost produced with the penalized model is \$25793.0, while the constrained optimization produces an infeasible solution with a cost of $\$4.4 * 10^{15}$. The algorithm fails to report the number of function evaluations, presumably because it was unable to find a feasible solution. It is important to note that the capability of the *differential_evolution* algorithm to handle constraints is very new and was just released in SciPy version 1.4.0. This could possibly explain the poor performance for the constrained model. In any case, it would be interesting to test this algorithm again in the future.

The results from the custom GA (*geneticOpt*) can be seen in Figure 8. the custom GA was only run with the constrained model. The custom GA performs similarly to SciPy's GA with the penalized model, but is slightly better with a cost of \$25608.4. The custom GA is also less computationally expensive, requiring 72821 function calls where SciPy's GA requires 114840 function calls. Both algorithms are very computationally expen-

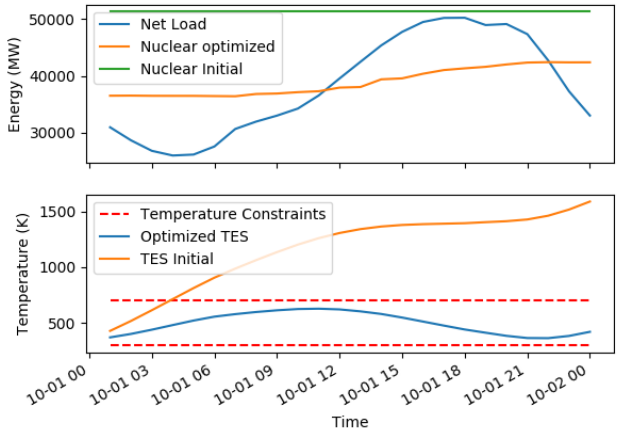

Fig. 6. Optimal results from *differential_evolution* penalized method

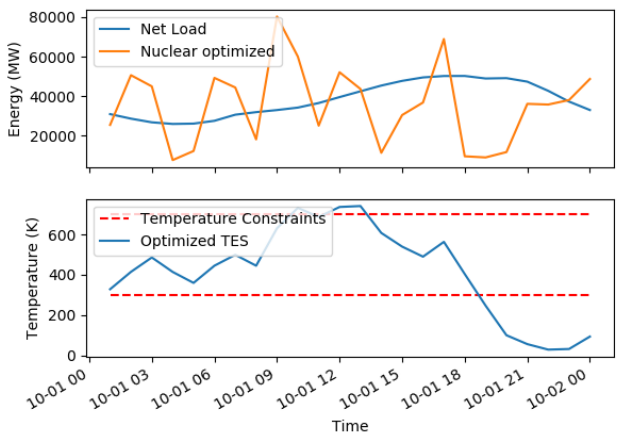

Fig. 7. Optimal results from *differential_evolution* constrained method

sive and require around 10 minutes to complete the optimization, while there is no guarantee that either algorithm will converge every time.

G. Trust Region

SciPy [29] has several implementations of trust region methods in the *scipy.optimize.minimize* package. The "trust-constr" method [6] was selected for this project as it is designed to handle constraints. Trust region methods can be considered gradient-free or gradient-based, and this specific algorithm is gradient-based. The distinguishing attribute of trust region methods as opposed to line searches is that they choose a distance to search, then a direction. This algorithm seems to find solutions similar to those found by other gradient-based optimizers. Results for the penalized model can be seen in Figure 9, and for the constrained model in Figure 10. The algorithm performed similarly for each model in terms of cost (penalized model: $21737.0, constrained model: $21510.9) with the constrained model yielding the lower cost. However, for the constrained model, the algorithm required ten times as many function calls as was required for the penalized model (penalized: 25025, constrained: 347825). Although this trust region implementation can produce very reasonable solutions, the computational expense is too large compared to other methods. In future work, it may be beneficial to test other trust region methods or develop a customized implementation.

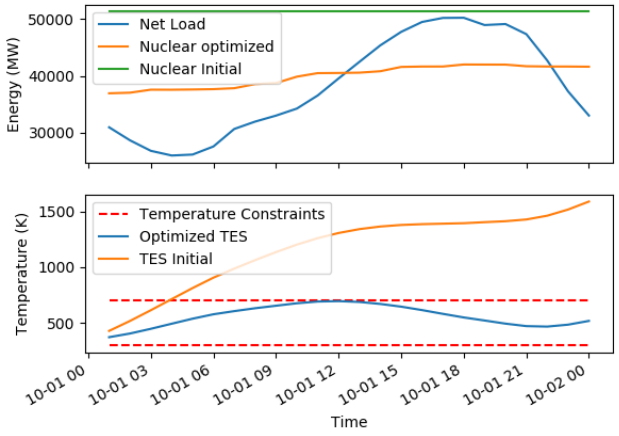

Fig. 8. Optimal results from the constrained *geneticOpt* custom GA

H. Fmincon

Both the constrained and penalty-based models were solved by the Sequential Quadratic Programming (SQP) algorithm using *fmincon* in Matlab optimization toolbox[18], [19]. This method was chosen because SQP is an iterative approach for solving non-linear constrained problems. And in our case, approximating the problem with a quadratic model might be a cheaper way due to the complexity of the model. The built-in finite-

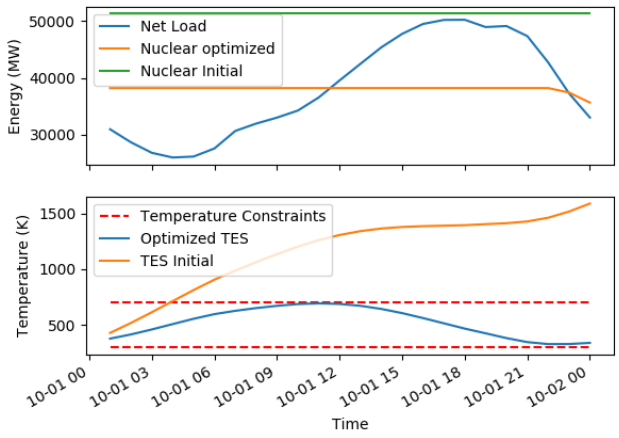

Fig. 9. Optimal results from the penalized model with the trust region method

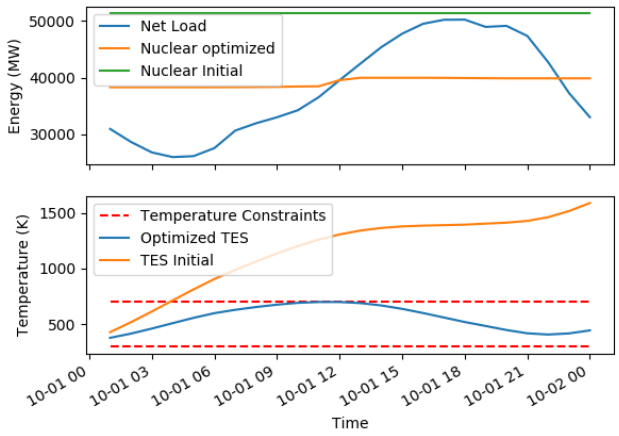

Fig. 10. Optimal results from the constrained model with the trust region method

differencing method, more specifically the forward differencing method, was used for approximating the gradients. And for this model, though the objective function and variables are pretty much on the same order of magnitude, the '*ScaleProblem*' option was set 'true' for numerical reasons.

The penalty-based model took 10 iterations to converge to an utterly feasible result, and the total number of function calls is only 279. The optimized cost is \$19672.5. The optimized nuclear generation and temperature profile are plotted in Figure 11, which is entirely feasible in terms of the temperature constraints and ramping rate.

Figure 12 shows the result of *fmincon* optimized constrained model. As we can observe from the plot, the optimized solution is pretty close to that of the penalty-based model, except for that the optimized nuclear generation line is even flatter. And the optimized cost of NHES is \$19077.1, which further decreases the cost by 3% compared to that of the penalty-based model. This improvement took 257 iterations and 6683 function calls.

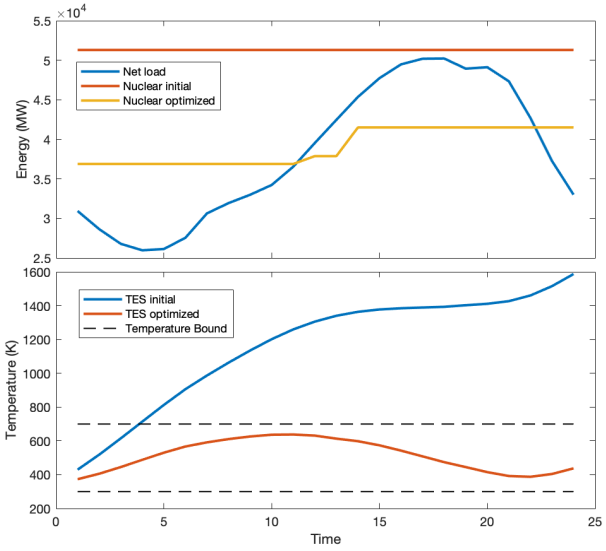

Fig. 11. Fmincon optimal results of penalty-based model

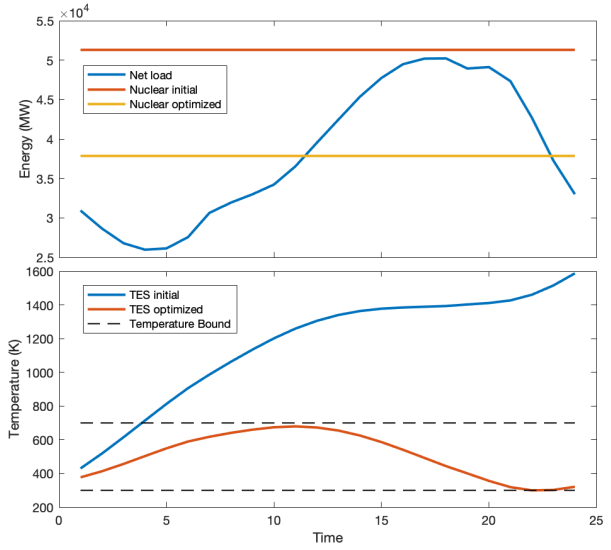

Fig. 12. Fmincon optimal results of constrained model

I. *Fminunc*

Only the penalty-based model was optimized by quasi-newton method[20] using *fminunc* in Matlab optimization toolbox[18], [19], since *fminunc* does not take any constraints. The quasi-newton method is a second-order optimization algorithm that solves unconstrained problems. The reason for using this method is that it approximates the inverse of hessian, which should give us a better search direction without being overly expensive. Again, the gradients were numerically approximated using forward differencing, and the '*ScaleProblem*' option was set 'true' for numerical reasons. The results are plotted in Figure 13. The solution does not violate either the temperature constraints or the ramping rate bounds. The optimization took only 3 iterations, and 275 function calls in total. However, the optimized cost is $24313.6, and this value is higher than both value optimized using *fmincon*. The quasi-newton method is better at searching for local optimum. If we start with a random initial condition that is not even close to the global optimum, this method might not give a decent solution.

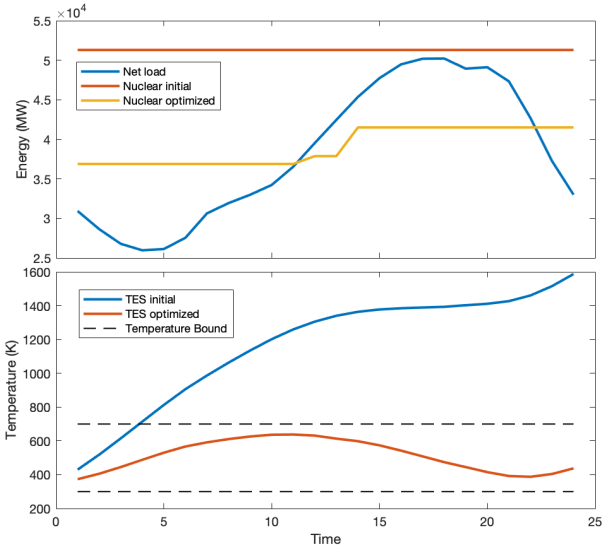

Fig. 13. Fminunc optimal results of penalty-based model

IV. Summary

The results are summarized in Table II. As shown in the table, *fmincon* using the constrained model gives optimal results. However, it is also costly, and it takes 6683 function calls, which is almost twice the number of function calls of SNOPT using the constrained model, while only improving the result by 0.03%. The constrained model generally works better and is more expensive when compared to the penalized model, which is reasonable since penalized methods approximate the actual problem using a series of unconstrained sub-problems, and it is hard to perform further optimization when the current solution is near the actual optimum. And when comparing the gradient-based and gradient-free method, it is obvious that the gradient-based method is more efficient and is capable of correlating the variables at different time points, while the gradient-free method is very expensive and the results are not reproducible due to its stochastic nature. One more thing to mention is that the results from the penalized model and gradient-free optimizer are pretty sensitive to scaling, which is not considered as a priority in this project. The original model also includes the one-time capital cost of molten salt, which significantly increase the total cost and scaling makes a huge difference in that case, as indicated by the results from *fmincon* with 'ScaleProblem' option on or off(not shown in the report). The current model brings the objective and design variable to the same order of magnitude; thus, scaling is not a significant impact factor.

TABLE II

Summary of the results of various algorithms applied to the sample NHES models

Algorithm	Type	Feasible	Calls	Optimal cost
Fmincon	Constrained	Yes	6683	19077.1
SNOPT	Constrained	Yes	4248	19077.5
SNOPT	Penalized	Yes	6084	19146.9
Fmincon	Penalized	Yes	279	19672.5
Trust Region	Constrained	Yes	347825	21510.9
Trust Region	Penalized	Yes	25025	21737.0
Fminunc	Penalized	Yes	275	24313.6
Custom GA	Constrained	Yes	72821	25608.4
Scipy GA	Penalized	Yes	114840	25793.0
SLSQP	Constrained	Yes	8105	26499.0
SLSQP	Penalized	No	4794	$9.6 * 10^{12}$
Nelder-Mead	Penalized	No	13135	$6.8 * 10^{13}$
Scipy GA	Constrained	No	-	$4.4 * 10^{15}$

Overall, most of the optimizers generated desirable results. However, this model is still limited. This simple prototype model of NHES gives opti-

mum for the selected load data. But in reality, the grid demand is not merely a repeated set of data. Instead, it is more dynamic and stochastic. And this might be improved in the future with further modifications.

V. Conclusions and Future Work

The model presented in this project has been simplified, and is a generalization of how nuclear hybrid energy systems might run. There is still much work to be done in order to form a physical model that is more realistic and thus more helpful in developing a real-world design. Specifically, some elements that should be studied include implementing model predictive control methods for real-time applications, further examining the dynamics of power grid economics, and combining an optimal system design with dispatch optimizations such as the ones that have been run here.

Additional work should also be done with the formation of the optimization model. As this problem was studied, it became clear that not all optimizers are effective with this kind of time-based problem. The results of this project show that the most efficient and effective methods are constrained sequential quadratic programming (SQP) optimizers. Interior point optimizers were not successfully benchmarked, but they would potentially yield similar results. There are many reasons why some optimizers perform more poorly than others, and the formulation of the problem usually determines whether an optimizer is successful or not. In this project some scaling of design variables and constraints was used, but inconsistently. In future work, it will be essential to determine the effect of scaling on all inputs, outputs, and constraints.

Other techniques that might be advantageous to study in this problem include methods of optimization under uncertainty and surrogate-based optimization. The load on an NHES always involves some degree of uncertainty. Using load data that has been collected previously, it would be beneficial to quantify that uncertainty and include it in the optimization model. If an optimal surrogate model can be developed, surrogate-based optimization could be useful to simplify calculations and obtain better gradients. This will be most effective as the complexity of the problem increases with more realistic elements and parameters.

References

[1] A. Alfonsi, C. Rabiti, D. Mandelli, J. J. Cogliati, and R. A. Kinoshita. Raven as a tool for dynamic probabilistic risk assessment: Software overview. *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, M and C 2013*, 2:1247–1261, 2013.

[2] A. Alfonsi, C. Wang, C. Rabiti, and D. Mandelli. Adaptive Surrogates within the RAVEN Framework for Dynamic Probabilistic Risk Assessment Analysis. *Proceedings of ANS Best Estimate Plus Uncertainty International Conference (BEPU 2018)*, pages BEPU2018–314, 2018.

[3] J. Augutis, L. Martišauskas, and R. Krikštolaitis. Energy mix optimization from an energy security perspective. *Energy Conversion and Management*, 90:300–314, Jan 2015.

[4] T.E. Baker, A.S. Epiney, C. Rabiti, and E. Shittu. Optimal sizing of flexible nuclear hybrid energy system components considering wind volatility. *Applied Energy*, 212:498–508, Feb 2018.

[5] L. D. R. Beal, D. C. Hill, R. A. Martin, and J. D. Hedengren. GEKKO optimization suite. *Processes*, 6(8), 2018.

[6] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods*, page 19. Siam, 2000.

[7] P. Denholm, M. O'Connell, G. Brinkman, and J. Jorgenson. Overgeneration from Solar Energy in California: A Field Guide to the Duck Chart. Technical report, National Renewable Energy Laboratory, 2013.

[8] W. Du, H. Garcia, and C. Paredis. An Optimization Framework for Dynamic Hybrid Energy Systems. *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, 96(March):767–776, 2014.

[9] A. Epiney, J. Chen, and C. Rabiti. Status on the Development of a Modeling and Simulation Framework for the Economic Assessment of Nuclear Hybrid Energy Systems (FY 16). Technical report, Idaho National Lab, 2016.

[10] H. E. Garcia, J. Chen, J. S. Kim, R. B. Vilim, W. R. Binder, S. M. Bragg Sitton, R. D. Boardman, M. G. McKellar, and C. J.J. Paredis. Dynamic performance analysis of two regional Nuclear Hybrid Energy Systems. *Energy*, 2016.

[11] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005.

[12] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong. User's guide for SNOPT 7.7: Software for large-scale nonlinear programming. Center for Computational Mathematics Report CCoM 18-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2018.

[13] W. E. Hart, C. D. Laird, J. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition, 2017.

[14] W. E. Hart, J. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[15] A. Ioannou, G. Fuzuli, F. Brennan, S. W. Yudha, and A. Angus. Multi-stage stochastic optimization framework for power generation system planning integrating hybrid uncertainty modelling. *Energy Economics*, 80:760–776, May 2019.

[16] D. Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[17] J. R. R. A. Martins and A. Ning. Engineering design optimization, 2020.

[18] MATLAB. *9.7.0.1190202 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2018.

[19] Matlab optimization toolbox, 9.7.0.1190202 (R2019b). The MathWorks, Natick, MA, USA.

[20] W.A. Murray, Institute of Mathematics, Its Applications, and National Physical Laboratory (Great Britain). *Numerical methods for unconstrained optimization*. Institute of Mathematics and its Applications Series. Academic Press, 1972.

[21] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 01 1965.

[22] Electric Reliability Council of Texas. Grid information. http://www.ercot.com/gridinfo, 2020.

[23] I. T. Papaioannou, A. Purvins, D. Shropshire, and J. Carlsson. Role of a Hybrid Energy System Comprising a Small/Medium-Sized Nuclear Reactor and a Biomass Processing Plant in a Scenario with a High Deployment of Onshore Wind Farms. *Journal of Energy Engineering*, 140(1):04013005, Mar 2014.

[24] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, Jan 2012.

[25] A. Sayigh. Renewable energy - the way forward. In *Applied Energy*, 1999.

[26] J. C. Spall. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[27] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[28] P. W. Talbot, A. Gairola, P. Prateek, A. Alfonsi, C. Rabiti, and Richard D. Boardman. Light Water Reactor Sustainability Program HERON as a Tool for LWR Market Interaction in a Deregulated Market. Technical Report Dec, Idaho National Lab, 2019.

[29] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, CJ Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[30] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[31] B. Zhao, M. Cheng, C. Liu, and Z. Dai. Conceptual design and preliminary performance analysis of a hybrid nuclear-solar power system with molten-salt packed-bed thermal energy storage for on-demand power supply. *Energy Conversion and Management*, 166:174–186, Jun 2018.