Computing, Telecommunication and Control, 2023, Vol. 16, No. 4, Pp. 18–27.
Информатика, телекоммуникации и управление. 2023. Том 16, № 4. С. 18–27.

Software of Computer, Telecommunications and Control Systems

Программное обеспечение вычислительных, телекоммуникационных и управляющих систем

Research article
DOI: https://doi.org/10.18721/JCSTCS.16402
UDC 004.67

NEW METHODS FOR EFFICIENT ENERGY MANAGEMENT OF A SOLAR VEHICLE ON A FIXED ROUTE

I.A. Selin ✉

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

✉ selin_ia@spbstu.ru

Abstract. This paper is devoted to the problem of efficient energy use on a fixed route for solar car racing. Solar cars are electric vehicles that can charge the battery while traveling using a built-in solar array. Solar car races are characterized by very long distances and prohibition of wall charging: vehicles can only be charged using solar energy. This alone creates a serious energy shortage, but coupled with the main goal of finishing in the shortest possible time, the task becomes even more complex. Thus, proper power management is the key to success. Since optimization of power management strategy is a computationally intensive process, modified route representation and use of automatic differentiation are advised to raise the performance of the optimization process. The proposed methods are implemented in the SPbPUStrat software solution, which is capable of solving the problem in a short time: this allows the strategy to be recalculated in full upon request.

Keywords: solar car, strategy, power management strategy, optimization, route representation

Citation: Selin I.A. New methods for efficient energy management of a solar vehicle on a fixed route. Computing, Telecommunications and Control, 2023, Vol. 16, No. 4, Pp. 18–27. DOI: 10.18721/JCSTCS.16402

Научная статья
DOI: https://doi.org/10.18721/JCSTCS.16402
УДК 004.67

НОВЫЕ МЕТОДЫ ОБЕСПЕЧЕНИЯ ЭФФЕКТИВНОГО ИСПОЛЬЗОВАНИЯ ЭНЕРГИИ СОЛНЦЕМОБИЛЕМ НА ЗАДАННОМ МАРШРУТЕ

И.А. Селин ✉

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

✉ selin_ia@spbstu.ru

Аннотация. Статья посвящена проблеме эффективного использования энергии на заданном маршруте в гонках солнцемобилей. Солнцемобили это электромобили с возможностью заряда по ходу движения с помощью встроенных солнечных панелей. Гонки солнцемобилей характеризуются очень длинными дистанциями и запретом подзарядки от электросети во время соревнований, болиды могут заряжаться только от солнечной энергии. Это создаёт большой недостаток энергии, а в сочетании с целью соревнований в виде минимизации времени прохождения дистанции, задача ещё больше усложняется. Поэтому критически важно правильно использовать энергию на дистанции. Но оптимизация режимов использования энергии имеет высокую вычислительную сложность, поэтому для ускорения процесса предложено использовать модифицированное представление маршрута и автоматическое дифференцирование. Методы реализованы в программном комплексе SPbPUStrat, способном решать проблему за короткое время, что позволяет перейти к пересчёту стратегии энергопотребления в полном разрешении по запросу.

Ключевые слова: солнцемобиль, стратегия, стратегия использования энергии, оптимизация, представление маршрута

Для цитирования: Selin I.A. New methods for efficient energy management of a solar vehicle on a fixed route // Computing, Telecommunications and Control. 2023. Т. 16, № 4. С. 18–27. DOI: 10.18721/JCSTCS.16402

Introduction

There are quite a few engineering sports competitions. Often used as a competitive testing ground, they help to develop new approaches for solving real-world problems. One of the sustainability-focused competitions mentioned is the solar car racing. Each participating team must build a compliant solar car and drive it over a specified route in the shortest possible time on a single battery charge. But the task seems to be problematic because race courses usually last thousands of kilometers and charging from the grid is prohibited. Participating vehicles are only allowed to charge using built-in solar panels. The lack of energy contradicts the main goal, to complete the route as fast as possible, and creates the need to use available resources in the most optimal way.

Vehicle models used for optimizing the movement of regular vehicles on the route are not suitable for vehicles using alternative power sources. There is a substantial difference in how those vehicles replenish energy. Unlike conventional cars, which only spend stored energy, solar cars can be recharged from solar panels as they move. The rate at which a solar car obtains energy depends on environmental conditions such as solar radiation and weather [1], making it dependent on both time and location. But energy consumption, as in the case of conventional cars, depends on the profile of the road surface and the movement speed. This discrepancy between the energy expenditure and the replenishment significantly complicates the problem. Energy replenishment can be reduced to an indirect dependence on the movement speed, but only at the cost of generating recurrence relations. Considering the goal of

finishing the race in the shortest possible time, this may lead to some counterintuitive conclusions: at some point in the race, it may be more efficient to go faster with more energy consumption, because the energy gain later will be even greater. That's why usually teams in solar car competitions rely heavily on energy management software for computing efficient power management strategies.

The common approach comes down to building a solar car model and formulating an optimization problem in terms of selecting the best controlling inputs to minimize the travel time over the fixed route while satisfying plausibility conditions. The route is usually represented in discrete form as a series of sections connecting route points, with ideally one section corresponding to one control input in the form of speed. Considering that, the problem takes the following form (1):

$$\begin{gathered} J\left(V_1,\, V_2,\, \ldots V_{N-1}\right) = J\left(V_1,\, V_2,\, \ldots V_{N-1}\right) = \sum_{i=1}^{N-1} \frac{\Delta s_i}{V_i} = t_{finish} \to \min, \\ s.t. 0 \le V_i \le V_{\max}, \\ 0 \le E_i\left(V_1,\, V_2,\, \ldots V_{N-1}\right) \le E_{\max}, \\ i \in \mathbb{N},\ 1 \le i \le N, \end{gathered} \tag{1}$$

where Δs_i is a route section length, V_i is a speed on i-th route segment, E_i is an energy at i-th point of the route, N is the number of route points (with $N-1$ route segments). E_i represents the energy balance and can be identified as follows (2):

$$\begin{gathered} E_i = E_{i-1} + E_{i-1}^{+}\left(V_i\right) - E_{i-1}^{-}\left(V_1,\, V_2,\, \ldots V_{i-1}\right),\ \ 2 \le i \le N, \\ E_1 = E_{\max}, \\ 0 \le E_i \le E_{\max}. \end{gathered} \tag{2}$$

The $0 \le E_i\left(V_1,\, V_2,\, \ldots V_{N-1}\right) \le E_{\max}$ is a nonlinear inequality, so the (1) problem can be classified as a single-criteria problem with nonlinear constraints.

Since the routes in solar car races are of great length, the number of track sections is also very large, which leads to the curse of dimensionality. As a result, the problem is considered to be too big to be solved in a direct form. Existing works resort to either simplifying the problem by reducing its resolution [2–4], or by optimizing only for the short distance in front of the solar car with full resolution [5, 6]. Some of these approaches [2, 5, 6] support the so-called online mode, where the strategy is recalculated on demand. But none of them allows obtaining a solution for the full problem size, let alone it being calculated on request in a short amount of time.

This paper is an extension of the already published research by the authors [7, 8] and is devoted to development of new techniques for effective power management of a solar car with an emphasis on the possibility of recalculating the strategy on demand. Suggested approach introduces notable differences in this process: the use of a modified track representation and the use of automatic differentiation in the solar car model for optimization purposes.

Route representation

Route data is introduced as an irregularly discrete set of points with the following information: latitude, longitude and altitude. Lateral data is not needed, since "the lateral vehicle dynamics is omitted in simulations" [2], thus the latitude and longitude are converted into the distance from the start of the race to the point of interest. This is the most common representation and is usually shown on graphs as altitude versus distance (route profile). Next, the data is processed to obtain a representation convenient for modeling purposes: sections between the route points. These segments contain such information as:

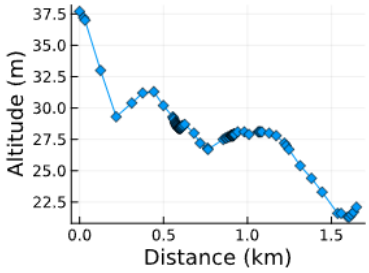

Fig. 1. Route profile

distance between points, elevation difference and inclination angle, calculated based on the original data [3]. Route representation is shown on Fig. 1.

The problem is that routes are very long and contain a very large number of segments, which does not contribute to the complexity of the optimization process. Knowing this, existing approaches work with fairly sparse route data, where one segment corresponds to a route length of more than 5 kilometers [5], which may be sufficient for tracks with a low elevation profile, but not for tracks with high altitude variance. In the literature on solar-powered auto racing, the question of choosing the discreteness of the track has not yet been raised. Therefore, a modified track representation is needed that is as small as possible in terms of the number of segments while still providing adequate simulation results.

Two track data processing techniques are proposed, based on merging neighboring segments with recalculation of their characteristics: keeping only elevation extremum points and parametric segments merge based on the difference in inclination. Both methods consist of 2 stages: selecting track points to keep (points of interest, POI) and then recalculating segment characteristics based on the selected points. Methods are inspired by Zhang S., et. al [9], but for the stated altitude over distance route representation.

When using the extremum point method, only points that are either above neighboring or below neighboring ones, or on the edges of plateau sections are saved in the POI list.

The parametric comparison is based on the slope angle since it is a normalized measure independent of the distance of each segment, which is vital for irregular grid representation. If the difference in a slope angle between adjacent track sections is less than a threshold value (k), then the shared point between the segments is added to the POI list. Threshold value of $k = 0$ will yield original track representation.

After the POI list is formed, the characteristics of new the segments are recalculated to obtain new values for the inclination angle and altitude. Instead of using only the information from the POI to construct the new segments, we are suggesting recalculating the new segments data using all the information so that the characteristics of newly formed combined segments are the same, as if we had simply taken unprocessed segments between the same points.

In the usual building of track segments, characteristics such as latitude, longitude and altitude can be obtained by simple averaging between the edge points of the segment (3):

$$avg = \frac{\mathrm{alt}\left(x_{i+1}\right) + \mathrm{alt}\left(x_i\right)}{2}. \tag{3}$$

But since the merger occurs on an uneven distance grid between the new POIs, there may be different altitude profiles with different average values, while having same altitude at POI. As a result, simply averaging data across POIs will produce wrong results. An example of this can be seen in Fig. 2, where the POIs are located at distances 1 and 4, and depending on the altitude profile, the true average value will be different compared to simple averaging by POI's altitude, which will always yield 4 as a result.

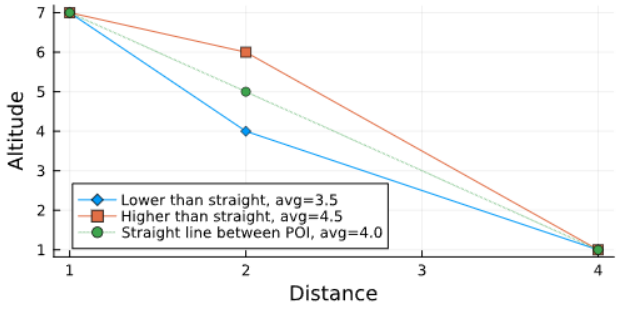

Fig. 2. Averages between POIs

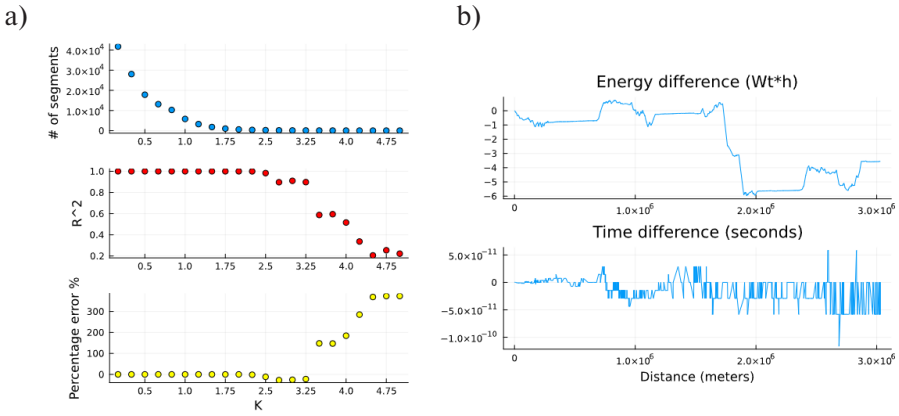

Fig. 3. Modeling quality with different parametric track representation

To avoid this effect, we suggest to perform numerical integration using the trapezoid rule and then average over the distance between the POIs (4):

$$avg = \sum_{i=1}^{k} \left[\frac{\mathrm{alt}(x_{i+1}) + \mathrm{alt}(x_i)}{2} \times (x_{i+1} - x_i) \right] \Big/ (x_k - x_1). \quad (4)$$

It is easy to see that in the case of the usual construction of segments without extracting POI, result will be the same as in (3), since k is equal to 2 and only one sum will be taken, so $(x_k - x_1)$ term cancels. But if multiple segments need to be merged, this will provide true average value, which will give almost the same result when using the processed segments in the model as the original track representation.

Keeping only extremum points as POI allows to reduce the number of segments by approximately 3 times. Parametric variation produces a wide range of results depending on the threshold value (Fig. 3, a). It is clearly seen that there is a certain interval of threshold values (k from 1 to 2.25), where the number of segments is greatly reduced, but the quality of the modeling does not suffer. An example of accumulated simulation error for $k = 1.75$ can be seen on Fig. 3, b. An energy difference of 5 W*h at the finish line is considered negligible, since it is less than 0.1% of the total capacity of the battery and is insignificant compared to the total energy consumption over the whole distance.

A numerical experiment was carried out, where the optimization process was carried out for the given problem (1) on routes of different lengths and different representations. Route representations used: unprocessed, the extrema method, the parametric method with $k = 1.75$. Optimization time and target

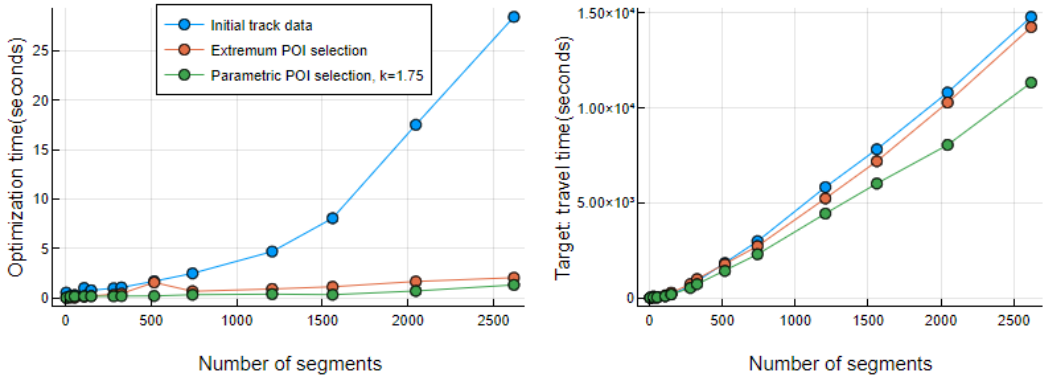

Fig. 4. Optimization and target time

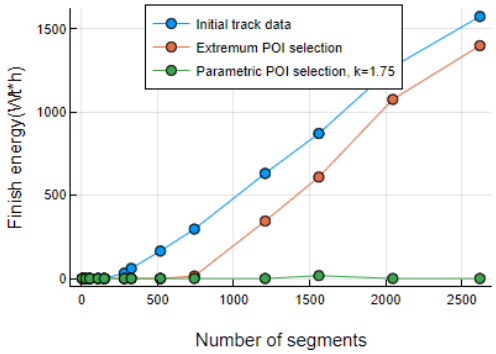

Fig. 5. Energy at the finish line

value (travel time) were measured. Fig. 4 shows that the developed methods show both better performance and achieve better target values.

It is worth noting, that the optimization process does not converge on the original track representation after reaching a problem size of 250 route segments, and after reaching 750 segments for the extrema method. The optimization process converges on routes of all sizes when using the parametric method. Convergence can be determined by the energy value at the finish line, it should be equal to zero. Otherwise, the solution is not optimal. A graph of the amount of energy at the finish is shown in Fig. 5.

Both the parametric method and the extrema method of simplifying the route representation make it possible to reduce the number of sections on the route without significant loss of quality, which leads to an improvement in both the performance and the convergence of the optimization process.

Automatic differentiation

Although the solar car model itself has already been described in the literature [10, 11], its implementation may differ in many ways. Ultimately, the model is built for use in the optimization process. And since optimization requires a lot of resources, its performance should be increased as much as possible.

Gradient-free optimization methods such as Nelder-Mead [12] or evolutionary strategies [13] are commonly used. The literature also describes the use of such 1st order methods as BFGS [14], but only using numerical differentiation, which is fraught with errors and requires additional function calls. However, literature review revealed no use of automatic differentiation in solar car models. Automatic differentiation [15–19] (AD, or differentiable programming) is a technique for automated building of a derivative of a program function, that operates with source code and overloads mathematical functions

and operators with their derivative counterparts. AD allows to program only the original function and to get derivative version of that function automatically, which will yield the same result as in analytical (symbolical) differentiation. But to execute AD, it is necessary to comply with some restrictions on the code of the target function: it must be unary and must not call code written in another language, and must use a certain hierarchy of data types.

Using automatic differentiation in the model allows to generate functions that calculate gradients and Hessians. And having them, it will be possible to apply 1st and 2nd order optimization methods without using numerical differentiation, which will lead to faster convergence.

Our model was adapted to use the automatic differentiation. A computational experiment was carried out to optimize the stated problem using 0th and 1st order methods on problems of different sizes. Both optimization time and target values were recorded. Methods requiring gradients were tested using numerical differentiation (ND) and automatic differentiation (AD). The optimization problem was solved by using methods: Nelder-Mead [12], BFGS [14] and Conjugate gradient [20]. Results can be seen in Table 1. It is clearly seen that 1st order methods using automatic differentiation give both the best result and the best optimization time.

Table 1

Optimization of the stated problem with and without the use of automatic differentiation

Problem size (segments)	Nelder-Mead	BFGS with ND	BFGS with AD	Conjugate gradient with ND	Conjugate gradient with AD
Optimization time (seconds)					
25	0.378	0.100	0.006	0.204	0.363
75	32.726	0.546	0.084	0.023	0.099
150	62.609	0.457	0.152	1.608	0.169
300	159.439	32.641	0.832	1.418	0.625
500	531.791	19.581	3.160	208.245	2.000
Target: travel time (seconds)					
25	356.857	356.810	356.810	356.810	356.810
75	1992.816	1756.572	1676.040	1643.247	1578.299
150	4084.410	3756.140	3642.381	3504.665	3508.342
300	9526.859	8869.733	9020.479	7827.724	7870.441
500	16977.256	14209.110	14636.160	14037.576	13905.262

Software implementation

A typical technology stack for implementing software that includes modeling and optimization will require a variety of tools and technologies. Implementation of the model requires tools with appropriate capabilities (e.g., Modelica, GPSS, SimPy, Simulink), formulation of the optimization problem requires an algebraic modeling language (e.g., AMPL, GAMS, Gekko) connected to the solver, and the application software is developed using a general-purpose language. This results in redundant integration code with additional performance overhead and possible errors. Using a multi-paradigm programming language can help smooth out the corners and get all the necessary steps done.

Our software solution, called SPbPUStrat, is developed using the Julia programming language [21], which is a multi-paradigm language for scientific computing. The solar car and environment models are made in pure Julia and adapted to use automatic differentiation with ForwardDiff.jl package [22]. Optimization problem statement and optimization itself were performed using Optim.jl [23] with

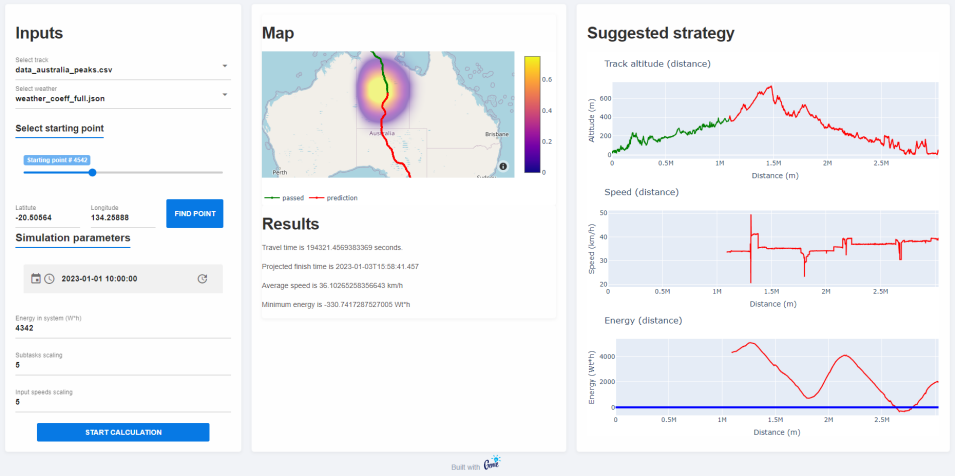

Fig. 6. Strategy dashboard

interior point Newton method [24]. The applied software was created using the Genie framework [25]. Additional packages include CSV.jl [26], Dataframes.jl [27] and Distributions.jl [28]. Thanks to a unified ecosystem, development time was significantly reduced, and almost no integration code was written. It also made debugging much easier and streamlined the delivery process.

The developed software is a client-server solution built according to the MVVM pattern. The software consists of two modules: calculation module and dashboard project. Calculation module provides all the necessary methods for setting the optimization problem and getting the solution. Dashboard module provides the user interaction. User interface can be seen on Fig. 6.

Results and discussion

The use of automatic differentiation and simplified track representation in combination with earlier results were tested by calculating the power management strategy on the full problem size. For comparison purposes, naïve approach (one speed for the entire route) and common approach (reduced problem resolution, 50 inputs) approaches were also tested. The power management strategy was calculated for the case of weather disturbances in the middle of the journey, which significantly affects the energy supplies. The results can be seen in Table 2. Tests were performed on a following configuration: AMD Ryzen 9 5900X, 64GB RAM, Windows 10, Julia 1.9.4 x64.

Table 2

Overall performance and quality results

Approach	Finish time	Improvement (hours)	Calculation time (seconds)	Energy at the finish line (Wt*h)	Scale
Naive	6d 05:17:01	–17.56	0.53	14683.0	Minimal (1)
Common	5d 11:42:68	0.00	14.4	1225.3	Reduced (50)
Proposed	5d 04:54:47	6.81	35.7	15.2	Full (10k+)

The naïve approach does not work well with weather disturbance. The speed has to be reduced to very low values to comply with energy constraints, resulting in much later completion times. The common approach does solve the weather problem and gives a better solution in terms of time. But it is clear that the solution is not optimal and there is a room for improvement, since the energy at the finish line is

above zero. The proposed methods provide a solution where the target value is 6.81 hours faster than the common approach, energy at the end is almost depleted.

Although the running time of the proposed methods is longer than that of other solutions, this is not a critical drawback. Typically, a recalculation of the power management strategy is necessary when the execution of the strategy begins to deviate from the original plan or when environmental conditions have changed, constant recalculation is unnecessary. The performance of the proposed methods is sufficient to perform calculations on request.

Conclusion

The problem of developing software for efficiently managing the energy of a solar car on a fixed route for racing is considered. The advantages and disadvantages of existing approaches were discussed. Opportunities for improvement were found in two areas: route presentation and automatic differentiation. The modified route representation made it possible to reduce the number of segments by up to 10 times without a huge loss in modeling quality. Automatic differentiation in the solar car model enabled the use of higher order optimization, resulting in better performance and convergence.

The proposed methods were implemented in the SPbPUStrat software solution, which allows the end user to recalculate the strategy on the fly during the race: this significantly improves strategic capabilities and increases the chances of success.

REFERENCES

1. **Teshima Y., Hirakoso N., Shigematsu Y., Hirama Y., Kawabata H.** Optimal Driving Strategy for Solar Electric Vehicle, IEEJ Journal of Industry Applications. 2021, Vol. 10, No. 3, Pp. 303–309.

2. **Yesil E., Onol A.O., Icke A., Atabay O.** Strategy optimization of a solar car for a long-distance race using Big Bang – Big Crunch optimization, 2013 IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI). 2013, Pp. 521–526.

3. **Schoeman S., Carroll J.** A combined energy system model for solar racers, 2013 Africon. Pointe-Aux-Piments, Mauritius: IEEE, 2013, Pp. 1–5.

4. **Wright G.S.** Optimal energy management for solar car race, Proceedings of the 39th Midwest Symposium on Circuits and Systems. 1996, Vol. 3, Pp. 1011–1014.

5. **Mocking C.** Optimal design and strategy for the SolUTra, MSc Report. 2006, Pp. 1–134.

6. **Guerrero Merino E., Duarte-Mermoud M.A.** Online energy management for a solar car using pseudospectral methods for optimal control, Optimal Control Applications and Methods. 2016, Vol. 37, No. 3, Pp. 537–555.

7. **Selin I.A., Kasatkin I.I., Zakhlebaev E.A., Hemminger O.P.** Building a time-optimal power consumption strategy for a solar car, IOP Conference Series: Material Science and Engineering. 2019, Vol. 643, No. 1. P. 012004.

8. **Selin I.A.** Otsenivaniye i optimizatsiya strategii energosberezheniya dlya solntsemobilya [Evaluation and optimization of the energy saving strategy for the solar vehicle], Materials of the XII Multi-conference on Management Problems (ICPU-2019), Southern Federal University Press. 2019. Vol. 2. Pp. 213–216.

9. **Zhang S., Liu K., Yao J.** A Simplified Approach to DEM Based on Important Points and the Triangular Irregular Network Angles, 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI). 2023, Pp. 934–93.

10. **Pudney P.** Optimal energy management for solar-powered CARS. University of South Australia, 2000.

11. **Thacher E.F.** A Solar Car Primer: A Guide to the Design and Construction of Solar-Powered Racing Vehicles. Cham: Springer International Publishing, 2015.

12. **Nelder J.A., Mead R.** A Simplex Method for Function Minimization, Comput. J. 1965. Vol. 7, No. 4. Pp. 308–313.

13. **Erol O.K., Eksin I.** A new optimization method: Big Bang–Big Crunch, Adv. Eng. Softw. 2006. Vol. 37, No. 2. Pp. 106–111.

14. **Broyden C.G.** The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations, IMA J. Appl. Math. 1970. Vol. 6, No. 1. Pp. 76–90.

15. **Ayda-Zade K.R., Yevtushenko Yu.G.** Bystroye avtomaticheskoye differentsirovaniye na EVM [Fast automatic differentiation on a computer]. Matematicheskoye modelirovaniye [Mathematical modeling], 1989, Vol. 1, No. 1. Pp. 120–131.

16. **Zasukhina E.S.** Fast automatic differentiation as applied to the computation of second derivatives of composite functions, Comput. Math. Math. Phys. 2006. Vol. 46, No. 11. Pp. 1835–1859.

17. **Evtushenko Y.** Computation of exact gradients in distributed dynamic systems, Optim. Methods Softw. Taylor & Francis, 1998. Vol. 9, No. 1–3. Pp. 45–75.

18. **Wengert R.E.** A simple automatic derivative evaluation program, Commun. ACM. 1964. Vol. 7, No. 8. Pp. 463–464.

19. **Linnainmaa S.** Taylor expansion of the accumulated rounding error, BIT Numer. Math. 1976. Vol. 16, No. 2. Pp. 146–160.

20. **Hestenes M.R., Stiefel E.** Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 1952. Vol. 49, No. 6. Pp. 409.

21. **Bezanson J., Edelman A., Karpinski S., Shah Viral B.** Julia: A Fresh Approach to Numerical Computing, SIAM Rev. 2017. Vol. 59, No. 1. Pp. 65–98.

22. **Revels J., Lubin M., Papamarkou T.** Forward-Mode Automatic Differentiation in Julia: arXiv:1607.07892. arXiv, 2016.

23. **Mogensen P.K., Riseth A.N.** Optim: A mathematical optimization package for Julia, J. Open Source Softw. 2018. Vol. 3, No. 24. Pp. 615.

24. **Wächter A., Biegler L.T.** On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 2006. Vol. 106, No. 1. Pp. 25–57.

25. Genie Framework – Productive Web Development With Julia. Available at: https://genieframework.com/ (accessed: 26.11.2023).

26. **Quinn J. et al.** JuliaData/CSV.jl: v0.10.11. Zenodo, 2023.

27. **Bouchet-Valat M., Kamiński B.** DataFrames.jl: Flexible and Fast Tabular Data in Julia, J. Stat. Softw. 2023. Vol. 107, No. 4.

28. **Dahua Lin et al.** JuliaStats/Distributions.jl: v0.25.103. Zenodo, 2023.

INFORMATION ABOUT AUTHOR / СВЕДЕНИЯ ОБ АВТОРЕ

Selin Ivan A.
Селин Иван Андреевич
E-mail: selin_ia@spbstu.ru
ORCID: https://orcid.org/0000-0002-8805-5887

Submitted: 30.10.2023; Approved: 18.12.2023; Accepted: 21.12.2023.
Поступила: 30.10.2023; Одобрена: 18.12.2023; Принята: 21.12.2023.