

Real-Time Optimisation of Control Strategy for a Low Fuel Consumption Vehicle Engine Using STM32 Microcontroller: Preliminary Results

Jakub Sawulski¹ and Maciej Ławryńczuk²

Abstract—The goal of this work is to analyse possibility of performing real-time optimisation of control strategy for a vehicle which leads to low fuel consumption using an embedded hardware platform with limited resources (the STM32 microcontroller). An original specialised car developed to take part in the Shell Eco-marathon competition is considered. At first, the optimisation task based on a vehicle model is discussed. In order to reduce on-line computational complexity, a relatively simple, but efficient parametrised control strategy is proposed. Next, since the majority of computation time is consumed by Ordinary Differential Equations (ODE) integration of the car model, a customised fast calculation method is developed using specific problem features. The discussed method is compared in terms of speed and memory usage with two classical approaches. It is approximately 10 times faster and is characterised by low memory demand.

I. INTRODUCTION

One of the main development trend in the automotive industry is to reduce vehicles' fuel consumption. Research concerned with fuel consumption minimisation is currently carried out by many institutions: from great automotive consortia to individual scientists. A well known event for efficient car inventors is the Shell Eco-marathon (SEM) students' competition [1].

Although fuel consumption is influenced by numerous factors, the vehicle engine control strategy is one of the most important. In practice, commonly known "eco-driving" or "hypermiling" methods improve energy efficiency. The mathematical fuel minimisation problem is difficult to solve, i.e. it is non-linear and non-convex. In simulations the dynamic programming approach based on Bellman's Principle of Optimality [2] may be considered, e.g. [3], [4]. Unfortunately, because of huge computational burden this approach typically cannot be used on-line on an embedded hardware platform with limited resources (usually a microcontroller).

This work discusses implementation of on-line fuel consumption minimisation using an embedded hardware platform with limited resources (the STM32 microcontroller). A mathematical model of an original specialised car [5] developed to take part in the SEM competition [1] is considered. The fuel consumption optimisation problem is formulated. A relatively simple, but efficient parametrised

control strategy is proposed. Based on specific problem features, a customised fast calculation method leading to fast integration of model differential equations is developed. The discussed method is compared in terms of speed and memory usage with two classical approaches.

II. VEHICLE AND ITS MODEL

A. PAKS vehicle

In this work an original concept car PAKS is considered. It was built by Student Association for Vehicle Aerodynamics at Warsaw University of Technology in 2011 [5]. The main goal of its development was to maximise energy efficiency since the PAKS was designed especially to take part in Urban Concept challenge in the SEM competition [1]. As a result of this assumption, the PAKS is characterised by remarkably reduced coefficient of rolling resistance and aerodynamic drag. In-depth modifications of mass-produced Honda 139FMB engine leads to increased performance resulting in the vehicle's low fuel consumption.

B. Vehicle dynamics

The vehicle model used for optimisation purposes is derived from Newton's second law based on simplified set of acting forces. Since only engine operation strategy is examined, the influence of only the following significant physical dependencies is considered:

- aerodynamic drag force,
- rolling resistance force,



Fig. 1. The original concept PAKS vehicle built at Warsaw University of Technology

¹Jakub Sawulski is with Warsaw University of Technology, Institute of Control and Computation Engineering, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland J.Sawulski@ia.pw.edu.pl

²Maciej Ławryńczuk is with Warsaw University of Technology, Institute of Control and Computation Engineering, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland M.Lawrynczuk@ia.pw.edu.pl

- down-slope force,
- propulsion force.

Other forces effects, e.g. generated by cornering or connected with inertia of rotational parts, are neglected due to relatively insignificant impact on vehicle dynamics, especially in case of the specialised car and a defined track with specified minimal corner radius.

The second main part of the mathematical model characterises vehicle fuel consumption as a function of driving strategy. Typically, instantaneous fuel consumption is a function of various parameters which describe the current state of engine and fuelling strategy is obtained by a dedicated on-board computer called Engine Control Unit (ECU).

For comparison of two or more different engine usage strategies, precise estimation of the amount of burned fuel is not necessary, but it is necessary to classify relatively the considered strategies. Hence, in order to describe fuel consumption a simplified formula, which involves only engine rotational speed, may be used.

As the vehicle model is derived from simple physical laws, it has a form of Ordinary Differential Equations (ODE) i

$$\begin{cases} \dot{x}_1 = x_2 & (1a) \\ \dot{x}_2 = \left(\frac{p_2 u^2}{m} - \frac{\rho S c_x}{2m} \right) x_2^2 + \left(\frac{p_1 u^2}{m} - g f_1 \right) x_2 \\ \quad + \frac{p_0 u^2}{m} + g f_0 - g \sin(\alpha(x_1)) & (1b) \\ \dot{x}_3 = (r_2 x_2^2 + r_1 x_2 + r_0) u & (1c) \end{cases}$$

where the manipulated variable u is the throttle opening ($0 \leq u \leq 1$) and the state variables are: x_1 – vehicle position, x_2 – vehicle speed, x_3 – integrated fuel consumption. Parameters are: p_n – propulsion force coefficients, $m = 192 \text{ kg}$ – vehicle mass, $\rho = 1.225 \text{ kg m}^{-3}$ – atmospheric air density, $S = 1.008 \text{ m}^2$ – vehicle cross sectional area, $c_x = 0.32$ – drag coefficient, f_n – rolling resistance coefficients, $g = 9.81 \text{ kg m s}^{-2}$ – standard gravity, r_n – fuel consumption coefficients and $\alpha(x)$ is the angle of the route slope as a function of the vehicle position (the function $\alpha(x)$ is known). All those values are measurable directly.

As the parameters p_n , f_n and r_n cannot be measured directly, they are obtained by parameter identification based on collected real-time measurements. The optimised values are: $f_0 = 0.000168$, $f_1 = 0.00133$, $p_0 = 233.1 \text{ N}$, $p_1 = -18.15 \text{ N s m}^{-1}$, $p_2 = 1.039 \text{ N s}^2 \text{ m}^{-2}$, $r_0 = 0.0458 \text{ mL s}^{-1}$, $r_1 = 0.0198 \text{ mL m}^{-1}$, $r_2 = 0.00074 \text{ mL m}^{-2} \text{ s}$. In general, the mentioned parameters may be not constant which means that a robust controller or on-line model re-tuning is necessary.

III. OPTIMISATION TASK

A. Fuel consumption minimisation

During the SEM competition, maximum time of trip (minimum average speed) and continuous motion constraints are set. It leads to the necessity of solving the following

nonlinear optimisation task

$$\min_u \int_0^{t_{\text{end}}} \dot{x}_3 \, dt + |\{t : u(t^-) = 0 \wedge u(t^+) \neq 0\}| \cdot r_{\text{start}} \quad (2)$$

subject to

$$\dot{x}_1 = x_2 \quad (2a)$$

$$\dot{x}_2 = \left(\frac{p_2 u^2}{m} - \frac{\rho S c_x}{2m} \right) x_2^2 + \left(\frac{p_1 u^2}{m} - g f_1 \right) x_2 + \frac{p_0 u^2}{m} - g f_0 - g \sin(\alpha(x_1)) \quad (2b)$$

$$\dot{x}_3 = (r_2 x_2^2 + r_1 x_2 + r_0) u \quad (2c)$$

$$x_1(0) = x_2(0) = x_3(0) = u(0) = 0 \quad (2d)$$

$$t_{\text{end}} \leq t_{\text{max}} \quad (2e)$$

$$x_1(t_{\text{end}}) = s \quad (2f)$$

$$x_2(0 < t \leq t_{\text{end}}) \geq 0 \quad (2g)$$

$$0 \leq u(0 < t \leq t_{\text{end}}) \leq 1 \quad (2h)$$

where: s – the given length of route, t_{max} – the maximum allowed time for reaching the finish line and r_{start} – the amount of fuel consumed per engine start-up. The solution of the optimisation task (2) is the trajectory of the control variable ($u(t)$) for which fuel quantity used is minimised.

The consecutive constraints are derived from:

- vehicle dynamics: (2a)–(2c),
- zero initial conditions: (2d),
- maximum lap time (competition rule): (2e)–(2f),
- vehicle stop is not allowed (competition rule): (2g),
- control value limits: (2h).

B. Engine operation strategy

To solve the optimisation problem (2), continuous-time control strategy, $u(t)$, must be parametrised by a finite number of characteristic point values. A typical approach to conversion from continuous-time domain function to a finite series of values is discretisation. Using the well-known methods (e.g. zero-order hold), a continuous function may be converted into series of its approximate values on a specific time interval.

In the particular case of the SEM competition, the considered route consists of equal track laps of length approx. $s = 1700 \text{ m}$ with mean time $t_{\text{max}} = 240 \text{ s}$ for passing it. Observing relative fast vehicle dynamic (acceleration from 0 km h^{-1} to mean speed in approx. 2.5 s), it can be concluded that it is not necessary to use the sampling time, T_s shorter than 2 s . It leads to the requirement of establishing about 120 independent control values per one lap which causes significant computational complexity of the optimisation task (120 decision variables). Thus, other parametrisation methods should be considered.

Various fuel efficient engine usage strategies have been invented. One of them is Pulse and Glide (PnG) strategy, which is commonly used at the SEM competition, especially in Internal Combustion Engine vehicles category [6], [7]. Since the PAKS is equipped with the constant gear ratio, the engine usage period (the pulse phase) is characterised

by only two parameters: the time point of engine start-up and shut-down (alternatively, engine operation time period). Due to the well-reduced vehicle resistive force factors, not more than four several seconds engine operation time spans are expected. As a result, the number of decision variables is significantly reduced in comparison with the rudimentary case when the control trajectory is characterised by as many as 120 parameters. Hence, simple parametrisation of the continuous PnG strategy is chosen for further research.

IV. EXPERIMENTAL SET-UP

A. Hardware description

The main goal of the discussed methodology is to use it as a framework of real-time optimiser executed as embedded software by a simple on-board computer. For the test purposes the STMicroelectronics NUCLEO-F446RE development board is used. STM32F446RE microcontroller (MCU) equipped with hardware single precision floating-point unit and operating at 180 MHz frequency allows to solve relatively sophisticated numerical problems [8], [9]. Typically, considerable acceleration of computation (per operating frequency) in comparison to PC-class computer is observable as a result of no presence of a classical operating system handling essential for its proper operation multitasking, peripherals support etc.

The hardware platform also includes dedicated MCU hardware peripherals:

- Controller Area Network (CAN) bus interfaces: it gives possibility to acquire any available measurement from the on-board CAN-based electronic system and return the control strategy data to the engine control unit.
- UART-USB converter with Direct Memory Access (DMA) unit: for a rapid software prototyping. During software test it allows fast asynchronous data transmission next used for analysis and comparisons.
- 16-bit advanced control Timers: configurable in Master-Slave topology, 32-bit counter may be built, which results in precise measurement code execution time opportunity, independently from the main software loop.

B. Software workflow

The basic software architecture for experiments is presented in Fig. 2. Two main parts which are responsible for total computational effort are: the optimisation algorithm and objective function evaluation. During computational efficiency tests, the interface to CAN-based on-board data acquisition system is unused since it is responsible for providing accurate initial condition for simulation and collecting solutions.

Due to quite complex constraints included in the optimisation task (2), a specific solution method needs to be found. A solution candidate is a parameters' vector of continuous-time control trajectory u . At first, the constraints (2a)–(2c) are fulfilled. As those three constraints are a classical system of ordinary differential equations, numerical integration is performed in time span from 0 to t_{max} with the initial conditions (2d). After execution of this sub-routine, satisfaction

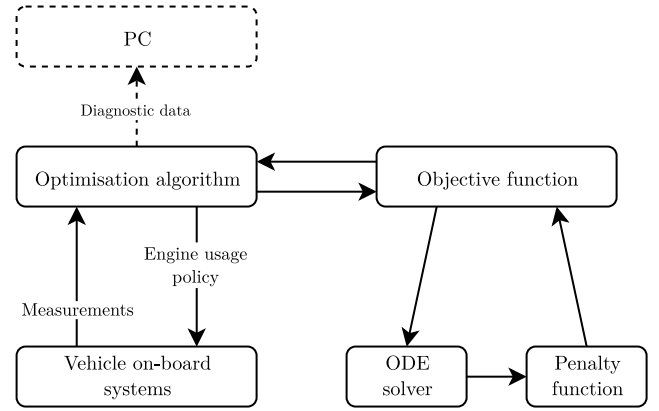


Fig. 2. Software architecture

of the remaining constraints is enforced by using exterior penalty function methods [10].

V. IMPLEMENTATION OF ON-LINE OPTIMISATION

A. Optimisation algorithm

As mentioned before, the optimisation task (2) is non-linear and non-convex which means that classical gradient-based optimisation algorithms are not expected to give good results.

In this work Jaya optimisation method is used. Jaya is a heuristic algorithm which concludes simple rule that a solution candidate in each iteration should be moved toward the best solution and evade from the worst achieved in the set of solution candidates called population [11]. Assuming that the best and the worst candidates (also called particles) are $\mathbf{b}$ and $\mathbf{w}$, respectively, the formula for update solution candidate coordinates is

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{r}_1(\mathbf{b}_i - \mathbf{x}_i) - \mathbf{r}_2(\mathbf{w}_i - \mathbf{x}_i) \quad (3)$$

where $\mathbf{r}_1$, $\mathbf{r}_2$ are random vectors with unitary interval coordinates. Unlike the majority of similar methods, Jaya does not require any tunable parameters for proper operation. 15 individuals and 100 optimisation epochs are used.

The following stop conditions are used: the total number of algorithm iterations, the number of iterations without actual solution improvement, the total time of computation. In contrast to dynamic programming, which always gives the best solution, evolutionary algorithms give some approximations of the solution in the consecutive iterations. If so far the best solution is stored and can be retrieved, there is guarantee that between algorithm epochs, solution quality does not decrease. This is a highly worthwhile feature for system with hard real-time limitations.

B. Equations of motion integration

In order to ensure that the constraints (2a)–(2c) are fulfilled during optimisation, some algorithms for numerical integration of ordinary differential equations must be used. Of course, various classical well-known numerical methods may be used for this task [12]. Furthermore, exploiting specific

properties of the considered ODE problem, some methods for improving accuracy and shortening computation time may be formulated. In the considered particular case, as engine usage strategy consists of a set of consecutive periods of engine operation and coasting, one may conclude that:

- the entire integration time span may be divided into two groups characterised by two different time constants, therefore two different integration strategies may be engaged,
- at the moments of changing the engine state the time constant of the vehicle dynamics changes discontinuously: to maintain the desired solution accuracy of the chosen ODE solver integration restart is required [13].

Three ODEs integration methods are analysed:

1) *Fixed step integration implementation:* As mentioned before, the most time consuming part of the whole optimisation procedure is integration of differential equations. One of widely used procedure for numerical ODE integration method is the 4th order Runge-Kutta (RK4) method [14], [15]. It has important advantages: simple implementation, reasonable low computational complexity and satisfactory accuracy in wide set of basic applications which can be improved by choosing a proper integration step.

2) *Adaptive step integration implementation:* One of widely used procedure for numerical ODE integration method are modified Runge-Kutta methods. For example, they allow for adaptive integration step size control based on analysis difference between 4th and 5th order solutions. It eliminates disadvantage of the “pure” RK4 method which is a requirement of choosing a small step to ensure the desired solution quality which causes significant lengthening of computation time. Implementation of Cash-Karp (RKCK) method [16] featured in [14] has been chosen.

Software implementation is slightly different from that for the classical PC platform. In particular, it concerns memory management, i.e. static memory allocation is used rather than dynamic one, which also simplifies generated executables.

3) *Fast integration implementation:* The developed integration strategy combines advantages of two aforementioned methods. As a result of characteristic features of the optimisation task (2), a simple decision table is built to select the current integration step size. As vehicle down-slope and propulsion forces have higher order of absolute values than others and may change rapidly, a step size switching strategy based on these premises is proposed. Hence, a more advanced algorithm of step size adaptation is not necessary and similar accuracy is expected, which is very important.

The integration algorithm switches between two predefined steps (t_1 and t_2) depending on the local variability of the right hand sides of the ODE system determined by aforementioned specific points, i.e. points of changing engine state and elevation data points. Shorter steps (t_1) are used in neighbourhood of them within predefined radius, in particular case: 5 m, otherwise longer time step (t_2) is set. The solver is reinitialized at points of start-ups and shut-downs as the engine state is a straightforward indicator of vehicle motion time constant.

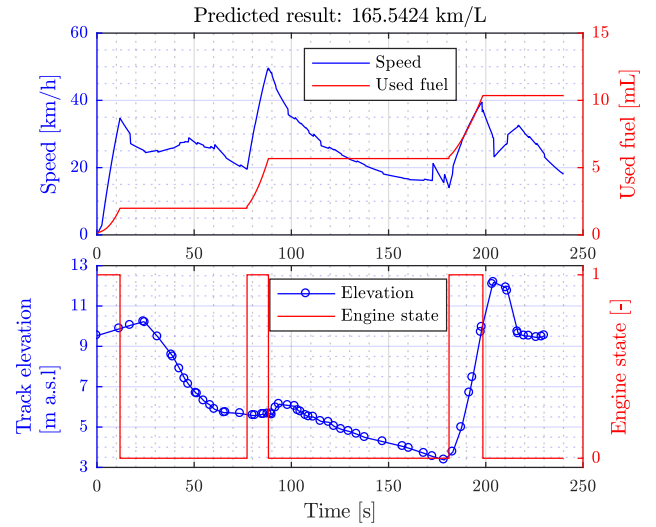


Fig. 3. Solution obtained by integration with constant step size ($t = 0.05$ s)

VI. RESULTS OF EXPERIMENT

To ensure that the integration methods can be reliable compared, for all presented solution trials the same engine control strategy is simulated. The optimisation task (2) is solved in MATLAB environment by analogous Jaya implementation with the use of `ode45` solver as internal cost-function integrator. The obtained values of the characteristic points vector of the control trajectories are

$$v = [11.864; 77.209; 88.209; 181.104; 198.517]^T \quad (4)$$

which indicate the proposed sequence of time points for engine state switching.

The most basic method of ODE integration is the RK4 constant time-step method. Choosing relatively small $t = 0.05$ s time step allows to provide high integration accuracy. The obtained state trajectories are presented in Fig. 3. For better readability only x_2 and x_3 (i.e. vehicle speed and used fuel) are presented. At long periods with no engine operation (constant amount of used fuel) no rapid trajectories changes are present. At those spans, extended integration step size may be used without considerable degradation of the solution quality. Similar trajectories calculated by variable step Cash-Karp method and also step size variation for further analysis are presented in Fig. 4.

Integration with the RKCK method returns solution comparable with the previous case, whereas utilised step sizes are much longer. One may observe that through the substantial part of the integration time relatively long steps are used, but also, for some other periods, significant step reductions are necessary. This behaviour may be a result of rapidly varying right-hand side of the ODE system.

Comparison with track elevation and engine state (Fig. 4) leads to conclusion that the majority of step reduction regions are corresponding to the engine state change or track elevation data points. As rapidly changing engine state is a key feature to fuel efficiency improvement and provided track elevation data set [17] has limited accuracy

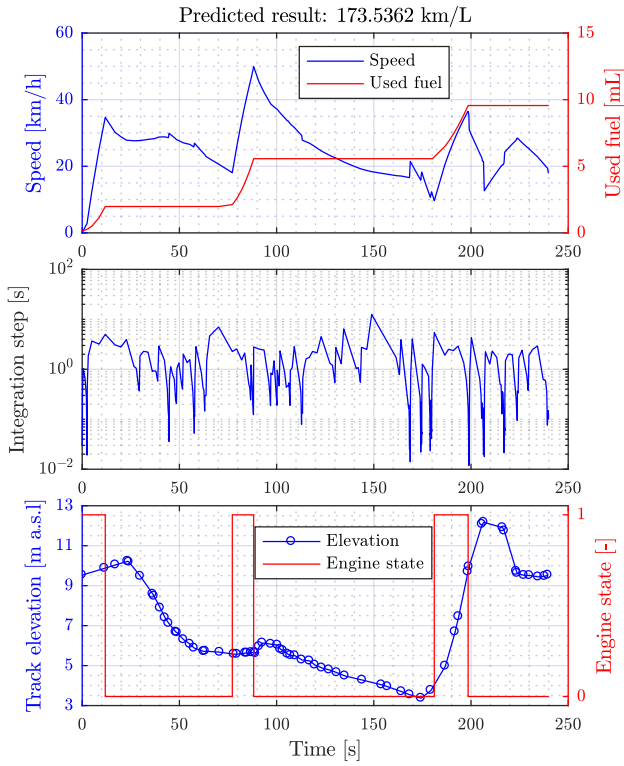


Fig. 4. Solution obtained by the RKCK method

(especially small number of data points), step size reduction is necessary to obtain sufficient solution quality. However, *a priori* knowledge about step-reduction-need regions allows for development of faster integration method.

As it can be observed in Fig. 4, the RKCK routine shortens time-step length typically in points of track elevation curve or engine state rapid changes which are known before ODE integration. Simple step size switching policy is proposed. When integration procedure reaches neighbourhood of characteristic points of aforementioned curves, shorter step is used. For time spans without fast changes in ODE right hand side functions, longer value is applied.

Fig. 5 presents comparison of solutions obtained by the RKCK method and the developed fast integration method. Despite a minor trajectory differences, final values of state variables are similar. This results are returned by a slightly improved “pure” RK4 formula which is enhanced by addition of basic rule which causes reducing integration steps in neighbourhood of characteristic points of track and near to fast engine state changes. For three analysed ODE integration methods, as depicted in Figs. 3, 4 and 5, resulting trajectories of state variables and final values of used fuel are similar.

By using the developed step size switching strategy computation time of integration may be reduced in comparison to the RKCK method. Fig. 6 presents measured total time of integration procedure evaluation as a function of the proposed step sizes. It can be observed that by using reasonable range step values, the execution time of a single integration process is of the order of single milliseconds.

Tab. I reports measured time of code execution obtained

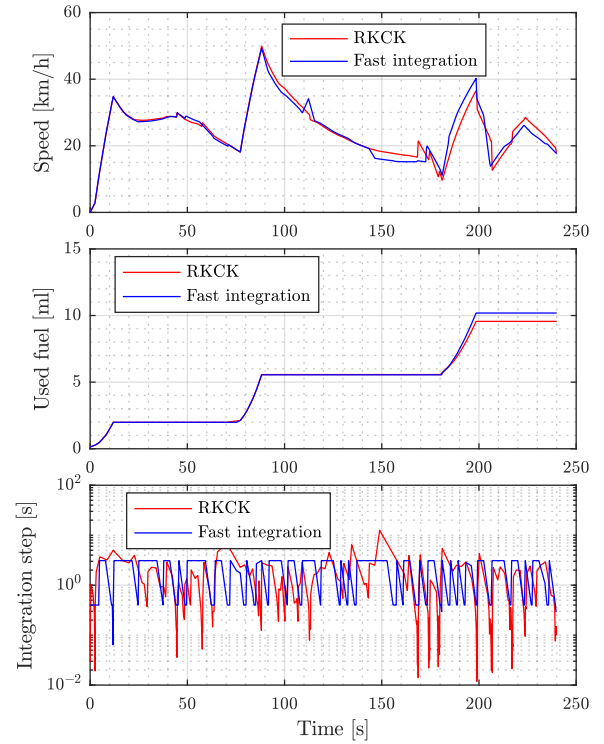


Fig. 5. RKCK and Fast Integration ($t_1 = 0.4$ s, $t_2 = 3.1$ s) method comparison

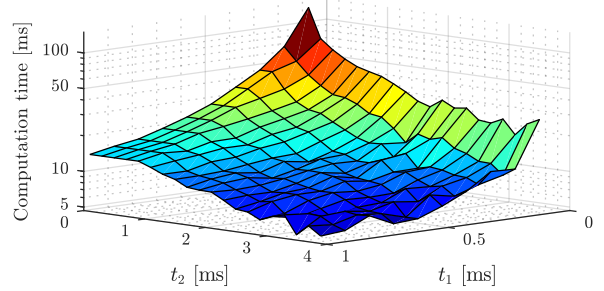


Fig. 6. Fast implementation computation time vs. time steps' sizes

by three compared methods. A measurement is performed by independent hardware 32-bit counter, it is performed with maximum available resolution. Application of fast integration method allows to reduce computation time about by ten times in comparison to the RKCK method and more than 20 times vs. the fixed step size method.

Tab. II presents solution time of the optimisation task (2) by Jaya algorithm. The fast integration method is more than 10 times faster than the RKCK approach and more than 20 times faster than the constant step size method.

Tab. III shows memory usage in the compared methods, independently for Code, Read-Only, Read-Write and Zero-Initialised memory banks. The fast integration method utilises significantly less MCU memory than the sophisticated RKCK procedure. It is crucial if the target embedded platform has limited resources. Code size for RK4 and fast integration methods are similar: the fast integration method is based on RK4 with addition of switching step size module.

TABLE I
ODE INTEGRATION TIME COMPARISON

Time of execution	Constant step size ($t = 0.05$ s)	Step size adaptation (RKCK)	Fast integration implementation ($t_1 = 0.4$ s, $t_2 = 3.1$ s)
Clock ticks	44320351	16110233	1711847
Time [ms]	246.22	89.50	9.51

TABLE II
JAYA MEAN OPTIMISATION TIME

Time of execution	Constant step size ($t = 0.05$ s)	Step size adaptation (RKCK)	Fast integration implementation ($t_1 = 0.4$ s, $t_2 = 3.1$ s)
Clock ticks	32343850060	14910893570	1381913079
Time [s]	179.68	82.84	7.68

Sophisticated formula of RKCK requires almost 50% code memory size. For RO and RW memory banks differences are not so significant as only several more integration method parameters need to be stored for RKCK method. Size of ZI-data depends directly on the number of performed integration steps. As after each step actual values of state have to be stored, increase of performed steps number linearly extends ZI memory demand. It should be stated that given memory usage applies to the entire software whose significant parts are diagnostic procedures for collecting experimental data. Independent modules responsible for integrating ODEs consume a correspondingly smaller memory resources.

VII. CONCLUSIONS

The objective of this work has been successfully achieved: an embedded real-time vehicle engine optimisation system based on the STM32 microcontroller has been developed. It returns the solution for the original concept PAKS car which leads to minimisation of fuel usage on-line. It is necessary to stress the fact that currently available hardware platforms for embedded systems have limited resources (i.e. speed and memory). Hence, taking into account the specific nature of the problem, a method for efficient integration of model equations has been proposed. As a result, the calculation time is reduced approximately 20 and 10 times vs. the integration method with a fixed step and adaptive step adjustment, respectively.

TABLE III
ODE SOLVER MCU MEMORY USAGE COMPARISON

Memory	Constant step size	Step size adaptation	Fast integration implementation
Code [B]	12344	20232	12892
RO-data [B]	692	840	692
RW-data [B]	548	640	552
ZI-data [B]	78964	6352	4752

Calculation time of the control trajectory is shorter than 8 seconds. Multiple recalculations of the strategy during the journey are possible, e.g. after unpredictable disturbances. The resulting time value indicates worst-case execution time. After driving a part of the distance, simulation time span is shortened (simulation from the current point to the finish line). Furthermore, significant shortening of execution time is caused by reduction the number of the decision variables (after every single engine usage period two parameters less have to be found). Additionally, a more computationally efficient MCU may be used. STM32F446RE microcontroller can operate properly only up to 180 MHz clock frequency. The newest generations of STM32 devices can reach up to 400 MHz which will shorten execution time at least twice.

During future research, the developed optimisation system will be implemented for the real PAKS car.

REFERENCES

- [1] S. Cosier: Race for the Best Mileage: Student-Built Caps Leave Automakers' Models in the Dust. Science World, vol. 67, no. 1, p. 18, 2010.
- [2] R. Bellman: Dynamic Programming. Princeton University Press, 1957.
- [3] H. T. Luu, L. Nouvelière, S. Mammari: Dynamic Programming for fuel consumption optimization on light vehicle. IFAC Proceedings Volumes 43.7, 6th IFAC Symposium on Advances in Automotive Control, pp. 372-377, 2010.
- [4] J. J. Santin, et al.: The World's Most Fuel Efficient Vehicle: Design and Development of Pac Car II. vdf Hochschulverlag AG an der ETH Zürich, 2007.
- [5] Students' Association for Vehicle Aerodynamics (in Polish). Wikipedia. Wikimedia Foundation, 2018. http://pl.wikipedia.org/w/index.php?title=Studenckie_Ko%C5%82o_Aerodynamiki_Pojazd%C3%B3w.
- [6] P. Sulikowski, R. Maroński: Quasi-Periodic Control Technique for Minimizing Fuel Consumption During Record Vehicle Competition. Archive of Mechanical Engineering, vol. 60, no. 2, pp. 185-197, 2013.
- [7] S. Xu, et al.: Fuel-Optimal Cruising Strategy for Road Vehicles With Step-Gear Mechanical Transmission. IEEE Transactions On Intelligent Transportation Systems, vol. 16, no. 6, pp. 3496-3507, 2015.
- [8] P. Chaber, M. Ławryńczuk: Implementation of Analytical Generalized Predictive Controller for Very Fast Applications Using Microcontrollers: Preliminary Results. Advances in Intelligent Systems and Computing, vol. 577, pp. 378387, 2017.
- [9] STMicroelectronics: STM32 Nucleo-64 boards User Manual (UM1724), Rev. 12, 2017.
- [10] Ö. Yeniyay: Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications, vol. 10, no. 1, pp. 45-56, 2005.
- [11] R. Venkata Rao: Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7, pp. 19-34, 2016.
- [12] L. Shampine, M. Reichelt: The MATLAB ODE Suite. SIAM Journal on Scientific Computing, vol. 18, no. 1, pp. 1-22, 1997.
- [13] E. Eich-Soellner, C. Führer: Integration of ODEs with Discontinuities, Numerical Methods in Multibody Dynamics, Vieweg+Teubner Verlag, pp. 193-241, 1998.
- [14] W. H. Press, et al.: Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, 2007.
- [15] C. Runge: Über die numerische Auflösung von Differentialgleichungen. Mathematische Annalen, vol. 46, no. 2, pp. 167-178, 1895.
- [16] J. Cash, A. Karp: A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides. ACM Transactions on Mathematical Software, vol. 16, no. 3, pp. 201-222, 1990.
- [17] G. Vanier: London 2017 Track information for students, Shell Eco-marathon, 2016. <http://www.shell.com/energy-and-innovation/shell-ecomarathon/europe/for-europe-participants/important-information.html>.