ELECTRIMACS 2019 – Salerno, Italy, 21st-23rd May 2019

Computation of the Lambert W function in photovoltaic modeling

Efstratios I. Batzelis · Georgios Anagnostou · Chandan Chakraborty · Bikash C. Pal

Abstract Recently, the Lambert *W* function has emerged as a valuable mathematical tool in photovoltaic (PV) modeling and other scientific fields. This increasing interest is because it can be used to reformulate the implicit equations of the single-diode PV model into explicit form. However, the computation of the Lambert *W* function itself is still not clear in the literature; some studies use the iterative built-in functions in MATLAB or other computational platforms, while others adopt their own approximation formulae. This paper takes a deeper look at the ways the Lambert *W* function is evaluated in PV models and carries out a comparative study to assess the most commonly used methods in terms of accuracy, computational cost and application range. These alternatives are implemented in a modern computer and a typical microcontroller to evaluate their performance in both simulations and embedded applications. The analysis concludes that some series expansions are good options for PV modeling applications, requiring less execution time than the built-in MATLAB *lambertw* function and exhibiting negligible approximation error.

1 Introduction

The field of photovoltaic (PV) modeling has been steadily attracting the research interest for more than three decades. Various models for different PV technologies have been proposed in the literature, the *single-diode PV model* still being the most widely adopted approach. One of the main computational challenges of this model is the implicit nature of the fundamental current-voltage (*I-V*) equation (i.e. $f(I,V)=0$), which dictates numerical or iterative solution that hinders evaluation.

An alternative was first proposed in 2005 by Jain et al. [1], further developed afterwards by Petrone et al. [2]: they used the *Lambert W function* to reformulate the *I-V* equation of the PV cell into an equivalent explicit form (i.e. $I=f(V)$ or $V=g(I)$). This allowed for easier and straightforward calculation, avoiding numerical solution difficulties such as increased computational time, initialization challenges, convergence issues etc. [3,4]. It also provided theoretical insight on how each variable and parameter relate to each other, and permitted derivation of analytical models for more complicated PV structures, such as the PV string and array under partial shading [2–4].

Historically, the mathematical expression we^w was first studied by Lambert in 1758 and afterwards by Euler in 1783, but it did not get much attention from their contemporaries at that time. It was revived in the 1990s, when developers of the *Maple* computational platform, Corless et al., managed to give an exact solution to the *double-well Dirac delta function model* in quantum mechanics and other mathematical problems of the sciences [5]. The inverse relation of we^w was then identified as a function on its own, denoted Lambert *W* function (also found as *product logarithm* or *omega function* in the literature). In PV modeling, several published studies and a book [3] have adopted this approach since 2005; a SCOPUS search using the keywords "photovoltaic" and "lambert W" yields 75 papers and more than 1000 citations in total, while the actual number of relevant papers is hundreds. Clearly, the use of Lambert *W* function in PV modeling has created a new trend in the field.

However, when it comes to the computation of this function, the literature is not very clear; as it is an elementary function, it can be calculated either through iterative algorithms or approximation formulae. In this paper, the most common calculation approaches used in PV modeling are

E. Batzelis · G. Anagnostou · B. Pal
Department of Electrical and Electronic Engineering
Imperial College London
Exhibition Road, London SW7 2AZ, UK
e-mail: e.batzelis@imperial.ac.uk, georgios.anagnostou11@imperial.ac.uk, b.pal@imperial.ac.uk

C. Chakraborty
Department of Electrical Engineering
Indian Institute of Technology Kharagpur
Kharagpur 721302, India
e-mail: cc.iitkgp@gmail.com

examined [4–9], including the MATLAB *lambertw* built-in function, and compared in terms of accuracy, computational performance and formulation complexity; this comparison takes place in a modern computer and a typical microcontroller (MCU). Common implementation challenges are discussed and the developed code in MATLAB and C for each alternative is made available. Notably, this is the first comparative study to assess the implementation methods of the Lambert W function in PV modeling.

2 Single-diode PV model

The circuit of the single-diode PV model is shown in Fig. 1, being characterized by the so-called *five parameters*: photocurrent I_{ph}, diode saturation current I_s, diode modified ideality factor a, series resistance R_s and shunt resistance R_{sh} [2–4, 7–11]. This model describes any PV generator (cell, module, array etc.) under uniform operating conditions.

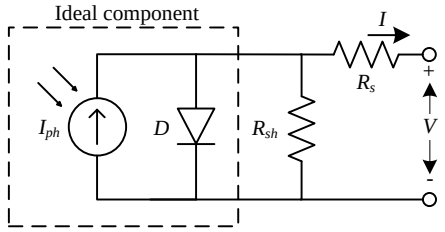


Fig. 1 Electrical equivalent circuit of the single-diode PV model [8].

The current-voltage (I-V) equation of this circuit is given conventionally in implicit form by:

$$I = I_{ph} - I_s\left(e^{\frac{V+IR_s}{a}} - 1\right) - \frac{V+IR_s}{R_{sh}} \tag{1}$$

where I and V are found in both sides of the equation; to solve (1), one has to employ a numerical or iterative algorithm with all associated difficulties. The equivalent explicit forms of (1) as $I = f(V)$ or $V = g(I)$ are found by applying the Lambert W function $W\{x\}$ [1–4, 8, 9]:

$$I = \frac{R_{sh}(I_{ph}+I_s)-V}{R_s+R_{sh}} - \frac{a}{R_s}W\left\{\frac{R_sR_{sh}I_s}{a(R_s+R_{sh})}e^{\frac{R_sR_{sh}(I_{ph}+I_s)+R_{sh}V}{a(R_s+R_{sh})}}\right\} \tag{2}$$

$$V = R_{sh}(I_{ph}+I_s)-(R_s+R_{sh})I - aW\left\{\frac{R_{sh}I_s}{a}e^{\frac{R_{sh}(I_{ph}+I_s-I)}{a}}\right\} \tag{3}$$

3 Calculation of the Lambert W function

The Lambert W function $W\{x\}$ is the inverse of the function we^w, i.e. the root of the equation $we^w = x$ (e.g. $we^w = 2 \leftrightarrow w = W\{2\} = 0.8526$). In general, this relation has several branches in the complex plane, thus not being a function in the conventional sense. In PV applications, however, the argument x takes real positive values (see (2) and (3)), which restricts $W\{x\}$ to the *principal branch* [5]. The principal branch is a function, thereafter referred to simply as the Lambert W function; an indicative plot is shown in Fig. 2 along with the logarithmic function for comparison.

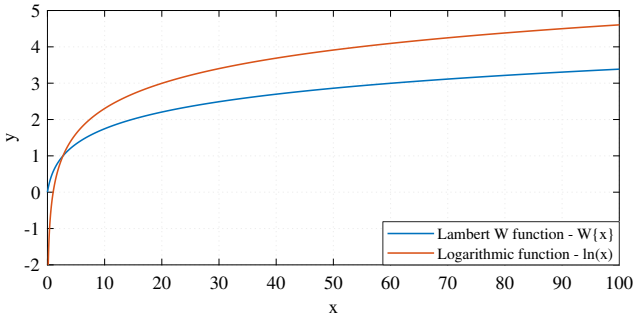


Fig. 2 The Lambert W and logarithmic functions.

The Lambert W function cannot be expressed in terms of other elementary functions and it is calculated either numerically or through approximation formulae. More information on the theoretical background and calculus may be found in [5] and other publications of these authors. In the following, the computational methods usually adopted in PV models in the literature are described and discussed.

3.1 The MATLAB lambertw function

The *lambertw* function in MATLAB provides solution to the general multi-valued relation in the complex plane; to get the principal branch, one can write $lambertw(0,x)$ or simply $lambertw(x)$. The evaluation is made numerically, using the Haley's method with an iteration step of [6]:

$$W_{j+1} = W_j - \frac{W_je^{W_j}-x}{e^{W_j}(W_j+1)-\frac{(W_j+2)(W_je^{W_j}-x)}{2W_j+2}} \tag{4}$$

where W is the Lambert W function of x and j indicates the iteration step. The majority of the relevant studies in PV modeling adopt this approach, as it is readily available and achieves machine accuracy. Nevertheless, the iterative nature renders this method computationally less efficient compared to the series expansions, especially when it comes to MCU implementation as shown later in the paper.

3.2 The asymptotic formula

The *asymptotic formula* was proposed by Corless et al. in 1996 [5]:

$$\begin{aligned} W\{x\}=&L_1-L_2+\frac{L_2}{L_1}+\frac{L_2(-2+L_2)}{2L_1^2}+\frac{L_2(6-9L_2+2L_2^2)}{6L_1^3}\\ &+\frac{L_2(-12+36L_2-22L_2^2+3L_2^3)}{12L_1^4}\\ &+\frac{L_2(60-300L_2+350L_2^2-125L_2^3+12L_2^4)}{60L_1^5}\\ &+O\left[\left(\frac{L_2}{L_1}\right)^6\right] \end{aligned} \tag{5}$$

where $L_1 = ln(x)$ and $L_2 = ln(ln(x))$. This formula is suitable for large arguments only; the *Wolfram MathWorld* suggests $x \geq 3$ [12], while the number of terms of (5) used in the literature varies: the first 7 terms in [4, 10], the first 4 terms in [7]. As shown in Section 4, the maximum approximation error for 7 terms and $x \geq 3$ is 1.32%.

A common difficulty faced when applying (5) is numerical error for very large arguments, as reported in [11] (returning Inf or NaN, calculation fails etc.). For example, if $x = 800e^{800}$ which is larger than the maximum floating-point number in IEEE double precision (64bit arithmetic), L_1 and L_2 become Inf and (5) yields NaN (not a number), although clearly $W\{800e^{800}\} = 800$ which is a perfectly finite value.

To overcome this limitation, the argument could be given in the form of $x = ae^b$, effectively making $L_1 = ln(a)+b$ and $L_2 = ln(ln(a)+b)$ in (5). In other words, instead of evaluating the large term x and then taking the logarithms, calculate directly L_1 and L_2 as functions of the finite a and b components of x. This trick is used in the implementation code of this formula and most of the following series expansions (see Appendix).

3.3 The hybrid calculation formula

To cope with the unsuitability of the asymptotic formula at small arguments, a *hybrid formula* is proposed in [4] which combines the asymptotic expansion (5) (7 terms) with a series expansion found in [13] that is accurate for small values:

$$W\{x\} = \begin{cases} W_1, & \text{when } 0 \leq x < 9 \\ W_2, & \text{when } x \geq 9 \end{cases} \tag{6}$$

where

$$\begin{aligned} W_1 = &u+\frac{u}{1+u}p+\frac{u}{2(1+u)^3}p^2-\frac{u(2u-1)}{6(1+u)^5}p^3\\ &+\frac{u(6u^2-8u+1)}{24(1+u)^7}p^4-\frac{u(24u^3-58u^2+22u-1)}{120(1+u)^9}p^5 \end{aligned} \tag{7}$$

$$\begin{aligned} W_2=&L_1-L_2+\frac{L_2}{L_1}+\frac{L_2(-2+L_2)}{2L_1^2}+\frac{L_2(6-9L_2+2L_2^2)}{6L_1^3}\\ &+\frac{L_2(-12+36L_2-22L_2^2+3L_2^3)}{12L_1^4}\\ &+\frac{L_2(60-300L_2+350L_2^2-125L_2^3+12L_2^4)}{60L_1^5} \end{aligned} \tag{8}$$

having $u = x/e$, $p = 1-x/e$ and $L_1 = ln(x)$, $L_2 = ln(ln(x))$. This formula yields a relative error lower than 0.1% for the entire real non-negative argument range [4].

3.4 The simple approximation formula

In [8], a much simpler calculation series is proposed, based on a 1961 study [14]:

$$W\{x\} = ln(x)\left[1-\frac{ln(ln(x))}{ln(x)+1}\right] \tag{9}$$

reported to yield errors less than 1.5% for $x \geq 2$ [8]. At smaller arguments, the calculation error is very large and (9) is not suitable for application.

3.5 The analytical approximation formula

A similar in nature approximation was introduced by Barry et al. in 2000 [15], adopted recently in a PV model [9]:

$$W\{x\} = (1+\varepsilon)ln\left(\frac{\frac{6}{5}x}{ln\left[\frac{\frac{12}{5}x}{ln\left(1+\frac{12}{5}x\right)}\right]}\right)-\varepsilon ln\left[\frac{2x}{ln(1+2x)}\right] \tag{10}$$

where $\varepsilon = 0.4586887$ is a constant. The approximation error of this formula is claimed to be 0.196% for $x \geq 0$. However, during our investigation we found that this level of accuracy is achieved for approximately $x \geq 3\times10^{-5}$ in 64bit double precision arithmetic ($x \geq 3\times10^{-3}$ in 32bit single precision arithmetic), since (10) does not seem to be convergent at near-zero values of the argument as explained later. The implementation code in this paper employs this consideration.

4 Simulation results

In this section, the accuracy and computational performance of the various Lambert W implementations are assessed and compared in simulations environment; these are denoted here as *MATLAB*, *Asymp7* (asymptotic formula with 7 terms), *Asymp4* (asymptotic formula with 4 terms), *Hybrid*, *Simple* and *Analyt* respectively. First, they are applied to the general case of real positive arguments, and then to an *I-V* curve of a

PV array. The implementation code is made publicly available (see Appendix), while the MATLAB approach (i.e. the *lambertw* function) is employed as a benchmark for assessing accuracy. All simulations are performed in MATLAB R2017b, in a computer with a 3.5-GHz CPU and 64-GB RAM at the default 64bit arithmetic.

4.1 Evaluation at real positive arguments

Here the six alternative methods are evaluated at real positive values of the argument; Fig. 3(a)-(b) depict the function outputs and the relative errors for the indicative case of $0 < x \leq 100$ (1 million values uniformly distributed).

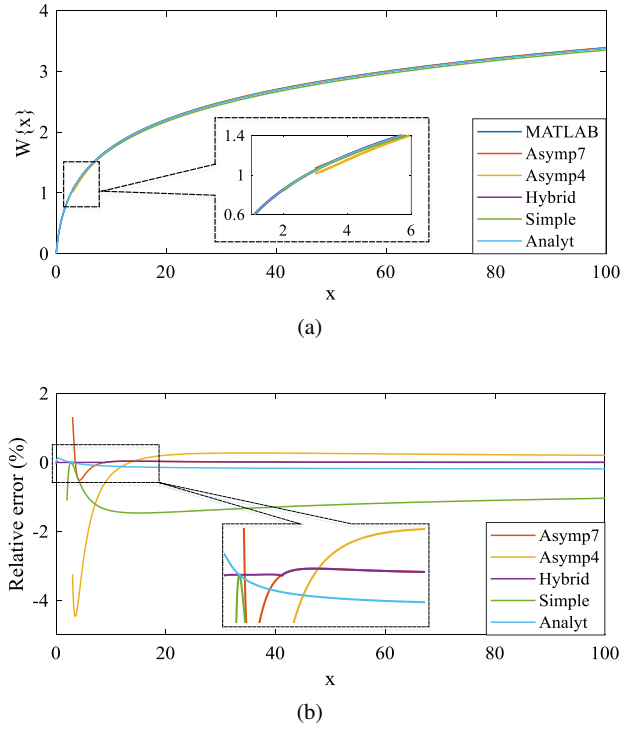


Fig. 3 (a) The Lambert *W* function outputs and (b) the respective relative errors at real positive arguments.

It seems that the results of all methods match quite well in Fig. 3(a), the main difference being the range of the argument x: the Asym7 (red line) and Asym4 (yellow line) approaches are defined for $x \geq 3$, and the Simple formula (green line) for $x \geq 2$. A clearer picture on the accuracy is given in Fig. 3(b): the errors are larger at small arguments, the Asym7, Asym4 and Simple methods exceeding 1% for some values. The Hybrid formula (purple line) seems to be the most accurate among the series expansions.

To investigate the argument range of the Analyt method, the results of the Analyt and MATLAB approaches at very small arguments up to 10^{-4} are shown in Fig. 4. It is evident that the formula (blue line) is not convergent at near-zero values; a lower bound of 3×10^{-5} has to be applied to the input range to achieve a relative error of about 0.2%, which is close to 0.196% considered by the authors [9, 15].

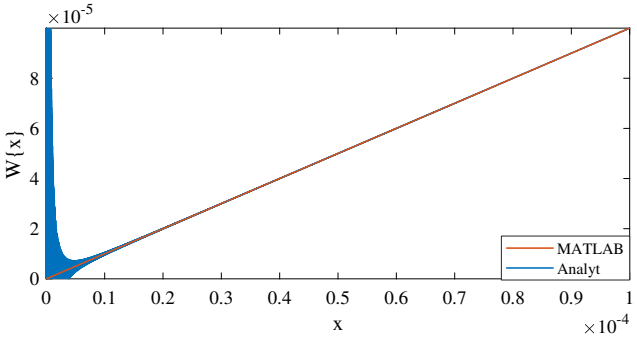


Fig. 4 The Analyt method at very small positive arguments.

The complete picture on the accuracy, argument range and execution time is given in Tab. 1, where bold font indicates most favorable values. The Hybrid formula is the most accurate, followed by Analyt method, while the rest yield higher max errors above 1%. As for the calculation cost, clearly the MATLAB function exhibits the highest execution time due to the iterative algorithm involved; the approximation formulae require approximately $1/3$ of the MATLAB time, except for the more demanding Analyt method due to the larger number of the logarithmic function calls.

Table 1 Performance of the six Lambert *W* function implementations at real positive arguments.

Method	Range	Max error	Time per evaluation
MATLAB	**Entire**	—	103 ns
Asymp7	$x \geq 3$	1.32%	35 ns
Asymp4	$x \geq 3$	4.46%	33 ns
Hybrid	**Entire**	**0.06%**	36 ns
Simple	$x \geq 2$	1.47%	**31 ns**
Analyt	$x \geq$3e-5	0.19%	63 ns

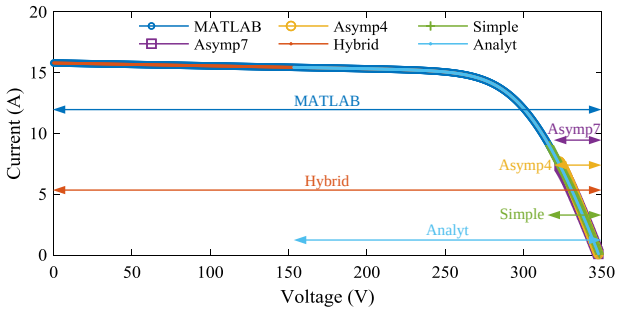
4.2 Evaluation of the I-V curve of a PV array

The results of the previous section are somewhat theoretical and do not indicate clearly the effectiveness of the Lambert *W* function implementations when it comes to a PV model. To assess this, the six alternatives are employed here to produce the *I-V* characteristic of a 4kW PV array at standard test conditions (2 strings, 12 Yingli YL-165 modules per string, parameters in Tab. 2). The *I-V* curve is found in two ways: calculating I as a function of V based on (2) (*I-approach*), and the opposite using (3) (*V-approach*). The results in the two cases are shown in Fig. 5(a)-(b) (1000 points).

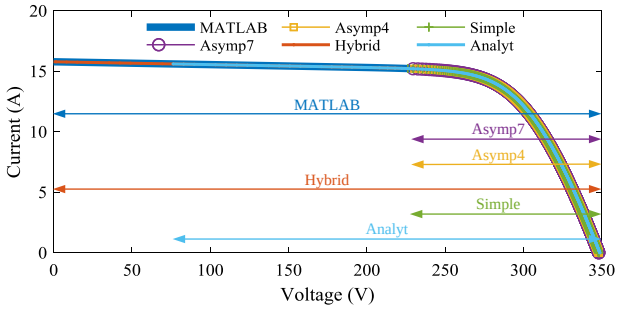
Apparently all curves coincide, but some of them are incomplete; the arrows indicate the range of application. The Asymp7, Asymp4 and Simple methods (purple, yellow and

Table 2 Five parameters of the PV array

STC Parameters					
I_{ph}	15.88 A	I_s	7.44e-10 A	a	14.67 V
R_s	2.04 Ω	R_{sh}	425.2 Ω		



(a)



(b)

Fig. 5 *I-V* curve produced: (a) I-approach and (b) V-approach.

green colors) can produce very limited part of the *I-V* characteristic in either figures, while the Analyt formula provides a larger part but still only a portion of the curve. This is due to the lower bound of the input range of these methods that leads to exclusion of some part towards the short-circuit current region. In other words, the argument of the $W\{x\}$ terms in (2) and (3) varies within [9.3e-10, 17.7] and [2.2e-8, 1.5e+192] respectively, the lower bounds being out of the input ranges of these methods (see Tab. 1). Only the MATLAB and Hybrid approaches provide the entire *I-V* characteristic in both cases.

The complete picture is given in Tab. 3. It is quite interesting how the seemingly small lower bounds in the argument range of the Asymp7, Asymp4, Simple and Analyt approaches restrict that much their application in calculating the *I-V* curve. If these bounds are relaxed, larger parts of the characteristic are produced, but they appear to be distorted and highly erroneous. These results correspond to the study-case 4kW PV array; for different PV system sizes, larger or smaller portions of the *I-V* curve can be produced.

The accuracy is measured using the normalized root mean square error (NRMSE) metric: the results seem acceptable for all methods, the Hybrid formula exhibiting the lowest errors in both I-approach and V-approach. As for the computational time per *I-V* curve point, similar results to the previous section are extracted, except that all times all slightly higher due to the additional overhead of (2)-(3). This analysis concludes that the Hybrid formula is the safest and most accurate method to produce the *I-V* curve among the series expansions.

Table 3 Performance of the six Lambert *W* function implementations in calculating the *I-V* curve of a 4kW PV array (times per points).

	I-approach			V-approach		
Method	**Range**	**NRMSE**	**Time**	**Range**	**NRMSE**	**Time**
MATLAB	**Entire**	—	203 ns	**Entire**	—	450 ns
Asymp7	>323 V	0.16%	39 ns	>229 V	0.001%	54 ns
Asymp4	>323 V	1.28%	**36 ns**	>230 V	0.009%	52 ns
Hybrid	**Entire**	**4.7e-5**	72 ns	**Entire**	**1.3e-6**	71 ns
Simple	>317 V	0.82%	40 ns	>226 V	0.021%	**50 ns**
Analyt	>153 V	0.03%	65 ns	>76 V	0.178%	96 ns

5 Experimental results

The Lambert *W* function has found recently embedded applications to PV systems, such as in model-based MPPTs and other control algorithms [16]. In these cases, the computational complexity is even more important due to the limited calculation capacity of the common MCUs. To investigate this aspect, this section discusses some experimental results from the implementation of the six Lambert *W* function alternatives in a typical MCU. The Texas Instruments model TMS320F28335 is used, which is often found in power electronics and motor drive applications; this MCU has an 150MHz clock and a 32bit hardware multiplier, but no division unit. The respective C code is properly optimized for this MCU (see Appendix).

The three case studies of the previous section are reexamined here. In terms of accuracy, the results are the same as the respective simulation ones, except from the Analyt method; the latter faced worse convergence issues at low arguments due to the 32bit arithmetic (64bit in simulations) which necessitated increase of the lower bound to 3×10^{-3} (from 3×10^{-5}). The findings on the application range and execution times are shown in Tab. 4.

Table 4 Computational performance of the six methods when implemented in an MCU (times per evaluation).

	Positive arguments		I-approach		V-approach	
Method	**Range**	**Time**	**Range**	**Time**	**Range**	**Time**
MATLAB	**Entire**	26.2 µs	**Entire**	15.0 µs	<294 V	31.4 µs
Asymp7	$x \geq 3$	7.7 µs	>323 V	8.9 µs	>229 V	7.9 µs
Asymp4	$x \geq 3$	**7.4 µs**	>323 V	8.6 µs	>230 V	7.6 µs
Hybrid	**Entire**	7.7 µs	**Entire**	**8.0 µs**	**Entire**	7.9 µs
Simple	$x \geq 2$	7.3 µs	>317 V	8.3 µs	>226 V	**7.5 µs**
Analyt	$x \geq$3e-3	17.9 µs	>221 V	21.2 µs	[90,294] V	21.5 µs

Comparing the application ranges of Tab. 4 to Tab. 1 and Tab. 3, it seems that Analyt is more restricted at low voltages, and that there is now an upper limit to both MATLAB

and Analyt methods in the V-approach. The former is due to the increased lower bound of Analyt, whereas the latter is because the maximum value of the argument 1.5e+192 (see Section 4.2) is not supported in the 32bit arithmetic of the MCU (as opposed to the 64bit MATLAB arithmetic). On the contrary, the rest of the approximation formulae do not face this difficulty, as they employ the large argument manipulation trick discussed in Section 3.2; unfortunately, this is not applicable to MATLAB's (4) or Analyt's (10).

As for the execution cost, the MCU times are of a few μs here, as opposed to the ns simulation values of Section 4. This becomes a critical computational burden to the MCU when the control algorithm requires evaluation of the Lambert *W* function at every switching cycle. For example, for a typical switching period of 50 μs (switching frequency 20 kHz), MATLAB and Analyt take up almost half of the calculation window, as opposed to the other much more cost-effective approximation formulae.

The main conclusion from this investigation is that when it comes to embedded applications, proper handling of large arguments and computational efficiency should be the main criteria in selecting the calculation method of the Lambert *W* function. The analysis concludes that the Hybrid approach exhibits the most favorable performance in this regard.

6 Conclusions

In this paper, we examine how the Lambert *W* function is calculated in PV models in the literature, including the MATLAB *lambertw* function and five approximation formulae. The accuracy, computational cost and applicability are assessed on a modern computer and a typical microcontroller.

The results indicate that the argument of the Lambert *W* function in the PV equations takes near-zero values close to the short-circuit and very large values close to open-circuit regions; this dictates that the calculation formula should be applicable to a wide range of real positive numbers (no lower or upper bounds if possible). The MATLAB *lambertw* function exhibits the best accuracy at the cost of higher execution time, but may fail at very large arguments when implemented in an MCU. Alternatively, the Hybrid formula given in [4] proves more computationally efficient and robust, but yields a calculation error of the order of 10^{-5} or 10^{-6}.

7 Appendix

The implementation code in MATLAB and C of the six Lambert *W* function alternatives is available online in the GitHub repository: https://github.com/ebatzelis/Lambert-W-function-in-PV-modeling.git.

The C code is properly optimized for MCU implementation (minimum number of divisions and logarithmic/exponential evaluations, factoring terms etc.). The implementation of the Asymptotic, Hybrid and Simple formulae overcomes the large argument contingency by inputing the Lambert W argument in the form of $x = ae^b$ (see Section 3.2). For example in (3), instead of calculating first the possibly large $x = \frac{R_{sh}I_s}{a} e^{\frac{R_{sh}(I_{ph}+I_s-I)}{a}}$ and then the Lambert *W* term $W\{x\}$, one can directly find $W\{x\} = lambertW\left(\frac{R_{sh}I_s}{a}, \frac{R_{sh}(I_{ph}+I_s-I)}{a}\right)$.

Acknowledgements
Dr. E. Batzelis' work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 746638. The rest of the authors' work has been conducted as part of the research project 'Joint UK-India Clean Energy Centre (JUICE)' which is funded by the RCUK's Energy Programme (contract no: EP/P003605/1).

References

1. A. Jain and A. Kapoor, "A new approach to study organic solar cell using Lambert W-function", *Sol. Energy Mater. Sol. Cells*, vol. 86, no. 2, pp. 197–205, Mar. 2005.
2. G. Petrone, G. Spagnuolo, and M. Vitelli, "Analytical model of mismatched photovoltaic fields by means of Lambert W-function", *Sol. Energy Mater. Sol. Cells*, vol. 91, no. 18, pp. 1652–1657, Nov. 2007.
3. G. Petrone, C. A. Ramos-Paja, and G. Spagnuolo, *Photovoltaic sources modeling*, Wiley-IEEE Press, first edition, 2017.
4. E. I. Batzelis, I. A. Routsolias, and S. A. Papathanassiou, "An explicit PV string model based on the Lambert W function and simplified MPP expressions for operation under partial shading", *IEEE Trans. Sustain. Energy*, vol. 5, no. 1, pp. 301–312, Jan. 2014.
5. R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function", *Adv. Comput. Math.*, vol. 5, no. 4, pp. 329–359, 1996.
6. C. Moler, "The Lambert W Function", [Online]. Available: https://blogs.mathworks.com/cleve/2013/09/02/the-lambert-w-function/, Sep. 2013.
7. Y. M. Roshan and M. Moallem, "Maximum power point estimation and tracking using power converter input resistance control", *Sol. Energy*, vol. 96, pp. 177–186, Oct. 2013.
8. E. I. Batzelis and S. A. Papathanassiou, "A method for the analytical extraction of the single-diode PV model parameters", *IEEE Trans. Sustain. Energy*, vol. 7, no. 2, pp. 504–512, Apr. 2016.
9. E. Moshksar and T. Ghanbari, "A model-based algorithm for maximum power point tracking of PV systems using exact analytical solution of single-diode equivalent model", *Sol. Energy*, vol. 162, pp. 117–131, Mar. 2018.
10. Y. M. Roshan and M. Moallem, "Maximum power point tracking using boost converter input resistance control by means of Lambert W-Function", in *Proc. 2012 3rd IEEE Int. Symp. Power Electron. Distrib. Gener. Syst.*, Aalborg, Jun. 2012, pp. 195–199.
11. A. Xenophontos and A. M. Bazzi, "Model-based maximum power curves of solar hotovoltaic panels under partial shading conditions", *IEEE J. Photovoltaics*, vol. 8, no. 1, pp. 233–238, Jan. 2018.
12. E. W. Weisstein, "Lambert W-Function", [Online]. Available: http://mathworld.wolfram.com/LambertW-Function.html.
13. R. M. Corless, D. J. Jeffrey, and D. E. Knuth, "Sequence of series for the Lambert W function", in *Proc. Int. Symp. Symb. Algebr. Comput. ISSAC*, Maui, HI, Jul. 1997, pp. 197–204.
14. W. Borsch-Supan, "On the Evaluation of the Function $\phi(\lambda)$ for Real Values of λ", *J. Res.*, vol. 65B, no. 4, pp. 245–250, Oct. 1961.
15. D. Barry, J.-Y. Parlange, L. Li, H. Prommer, C. Cunningham, and F. Stagnitti, "Analytical approximations for real values of the Lambert W-function", *Math. Comput. Simul.*, vol. 53, no. 1-2, pp. 95–103, Aug. 2000.
16. E. I. Batzelis, S. Papathanassiou, and B. C. Pal, "PV system control to provide active power reserves under partial shading conditions", *IEEE Trans. Power Electron.*, vol. 31, no. 11, pp. 9163–9175, Nov. 2018.