Progress in PHOTOVOLTAICS

PROGRESS IN PHOTOVOLTAICS: RESEARCH AND APPLICATIONS
Prog. Photovolt: Res. Appl. (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/pip.2301

APPLICATION

Fast and reliable calculation of the two-diode model without simplifications

Stephan Suckow*, Tobias M. Pletzer and Heinrich Kurz

Institute of Semiconductor Electronics, RWTH Aachen University, Sommerfeldstr. 24, 52074 Aachen, Germany

ABSTRACT

An algorithm to calculate the current in the two-diode equivalent circuit of a solar cell is described and characterized in detail. It enables fitting measured current–voltage characteristics with hundreds of voltage points and six fit parameters at practically instantaneous speeds and can handle thousands of voltage points within a few seconds, without simplifications of the two-diode model. This performance enables routine two-diode model parameter extraction at in-line speeds, which may help to enhance cell characterization for module integration. The source code is publicly available. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

two-diode model; *J–V* curve analysis; real-time analysis; open source

***Correspondence**

Stephan Suckow, Institute of Semiconductor Electronics, RWTH Aachen University, Sommerfeldstr. 24, 52074 Aachen, Germany.
E-mail: suckow@iht.rwth-aachen.de

Received 22 December 2011; Revised 18 June 2012; Accepted 8 September 2012

1. INTRODUCTION

Since the two-diode model for the solar cell equivalent circuit has been introduced by Wolf and Rauschenbach in 1963 [1], it has become a valuable standard tool in the area of solar cell characterization. It contributed significantly to a better understanding of the processes inside the cells and related efficiency loss mechanisms [2–5]. Compared with the simpler one-diode model, the two-diode model provides better accuracy, especially in the vicinity of the maximum power point of conventional Si solar cells (see e.g., [6] and references therein).

In solar cell analysis, a fit of the two-diode model parameters to a measured current density–voltage (*J–V*) characteristic is commonly performed. This fit routine can be a classical gradient based numerical procedure such as a Newton–Raphson (NR) method [5] or more exotic methods such as particle swarm optimization [7,8] or genetic algorithms [7,9].

At some point, all methods require the calculation of J for a given set of parameters, including V. Unfortunately, the underlying equation is transcendental and cannot be rearranged to solve for J directly (Section 2). The problem is caused by the series resistance R_S; hence, an approximate solution could easily be obtained by neglecting R_S and correcting it later. However, this is neither accurate nor suitable for a general purpose curve fitting algorithm, which must be able to deal with any R_S. On the other hand, calculating $J(V)$ numerically can be time consuming because of a large number of iterations. It also requires a careful search algorithm, as it could easily encounter numerically infinite function values. For a *J–V* curve with many voltage points N_V and a less than optimal algorithm, this procedure can take several minutes, making it unsuitable for interactive use and automatic analysis at in-line processing speeds. These difficulties have lead to numerous approaches to ease the problem, including single-diode models [10,11], approximations aimed at reducing the number of fit parameters and thus iterations [6], and estimation methods [12,13].

In this paper, we describe an algorithm to calculate $J(V)$ for a given set of solar cell parameters on the basis of the full two-diode model without simplifications in a very fast and reliable way. This routine provides a solid basis to be used in conjunction with any curve fitting algorithm to fit the *J–V* characteristic by varying the set of cell parameters. Our algorithm and the internal parameter choices are specifically tailored for the problem at hand, with both aspects being discussed in detail in Section 3. Algorithm run times, including scaling with the number of voltage points, are presented and discussed in Section 4. Curve fitting is demonstrated for an ordinary multicrystalline

silicon (mc-Si) solar cell using a curve fitting algorithm built into MATLAB (MathWorks, Natick, MA, USA). Neither the capabilities of this curve fitting algorithm nor a discussion of the two-diode model itself is the scope of this paper.

The entire program and its source code are available as a download on nanoHUB, as stand-alone executable and as MATLAB code [14]. The *J(V)* calculation discussed in this paper is contained in the file "diodeI.m" contained in "MATLAB files.zip".

2. FUNDAMENTALS

The equivalent circuit of a solar cell using two diodes is shown in Figure 1. The illumination is represented by the current source and leads to the photo current J_{Ph}. By setting $J_{Ph} = 0$, the circuit can be used to model dark J–V measurements. The diodes are D_1 and D_2 with the corresponding currents J_1 and J_2. D_2 is used to model Shockley–Read–Hall recombination currents in the space charge region, whereas D_1 represents recombination currents elsewhere. The latter can be Shockley–Read–Hall or Auger recombination in the base and emitter, or front and rear surface recombination [15]. Using this model, an injection independent bulk lifetime is assumed. The parallel resistance working as a shunt is R_P, and the series resistance of the entire circuit lumped into one resistor is R_S. The external voltage and current are V and J, respectively. J is the sum of four contributions, J_{Ph}, the current through the shunt, J_1 and J_2:

$$J = J_{Ph} - \frac{V + JR_S}{R_P} - J_{01}\left(e\frac{q(V + JR_S)}{n_1 k_B T} - 1\right) - J_{02}\left(e\frac{q(V + JR_S)}{n_2 k_B T} - 1\right). \quad (1)$$

J_{01} and J_{02} are the saturation currents and n_1 and n_2 the ideality factors of the diodes D_1 and D_2, respectively, k_B is Boltzmann's constant and T is the temperature. From here on, n_1 is kept constant at the value 1 and is thus omitted. According to classic theory, n_2 should be equal to 2 [16], but in practice often exceeds this value [2,17] and is thus treated as a fit parameter from here on.

Equation (1) cannot be rearranged to solve for J directly because of the terms $V + JR_S$ as summand and in the exponents. Instead, the problem is transformed into finding the root of function f:

$$f(V,J) = J_{Ph} - J - \frac{V + JR_S}{R_P} - J_{01}\left(e\frac{q(V + JR_S)}{k_B T} - 1\right) - J_{02}\left(e\frac{q(V + JR_S)}{n_2 k_B T} - 1\right) = 0. \quad (2)$$

where $f(V,J) = 0$ is only true for the correct value of J, which is yet to be determined.

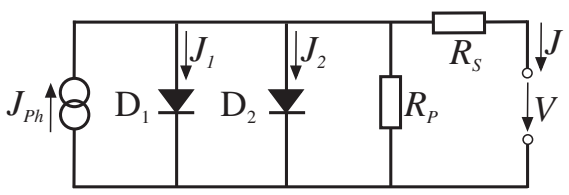

Figure 1. Equivalent circuit of a solar cell according to the two-diode model.

3. CALCULATING THE CURRENT

To calculate a single J–V characteristic, the algorithm described in this section is used to solve Equation (2) numerically for every V, that is, $f(J)|_{V=\text{const}} = 0$. All other parameters such as R_S, R_P and so on, are fixed during each J–V calculation. They are varied later on in separate J–V calculations by the curve fitting algorithm.

3.1. Preface

Our algorithm requires the data to be arranged according to the standard convention used for solar cells, that is, positive V corresponds to forward bias, the data starts with the most negative V, J is positive at reverse bias and eventually becomes negative at high forward bias. If this was changed, the algorithm had to be adapted at several positions. Using this convention, $f(J)|_{V=\text{const}}$ starts at some positive finite value for $J \to -\infty$ and decreases monotonically as $J \to \infty$. Exploiting this monotonic behavior is essential for our algorithm. Another important property of this function is an exponential increase of the slope as $J \to \infty$, caused by the exponential term in the current through diode D_1. Depending on the parameters, this slope can become extremely steep, so that a root finding algorithm can encounter $f > 0$ for a certain J and upon a small increase of J, the function value returned becomes $-\infty$, as the number is so large that it cannot be represented by a double precision floating point value. General purpose root finding and especially gradient based algorithms break in such situations and are thus not suitable to solve this task. Our algorithm avoids this problem by searching for the root carefully, which will become apparent in the next subsection.

3.2. Algorithm

The flow chart of our algorithm is depicted in Figure 2. It starts with a loop over all voltages V_i and is then further divided into three parts, for each voltage

(i) Initialize search.
(ii) Find interval of sign change.
(iii) Find root.

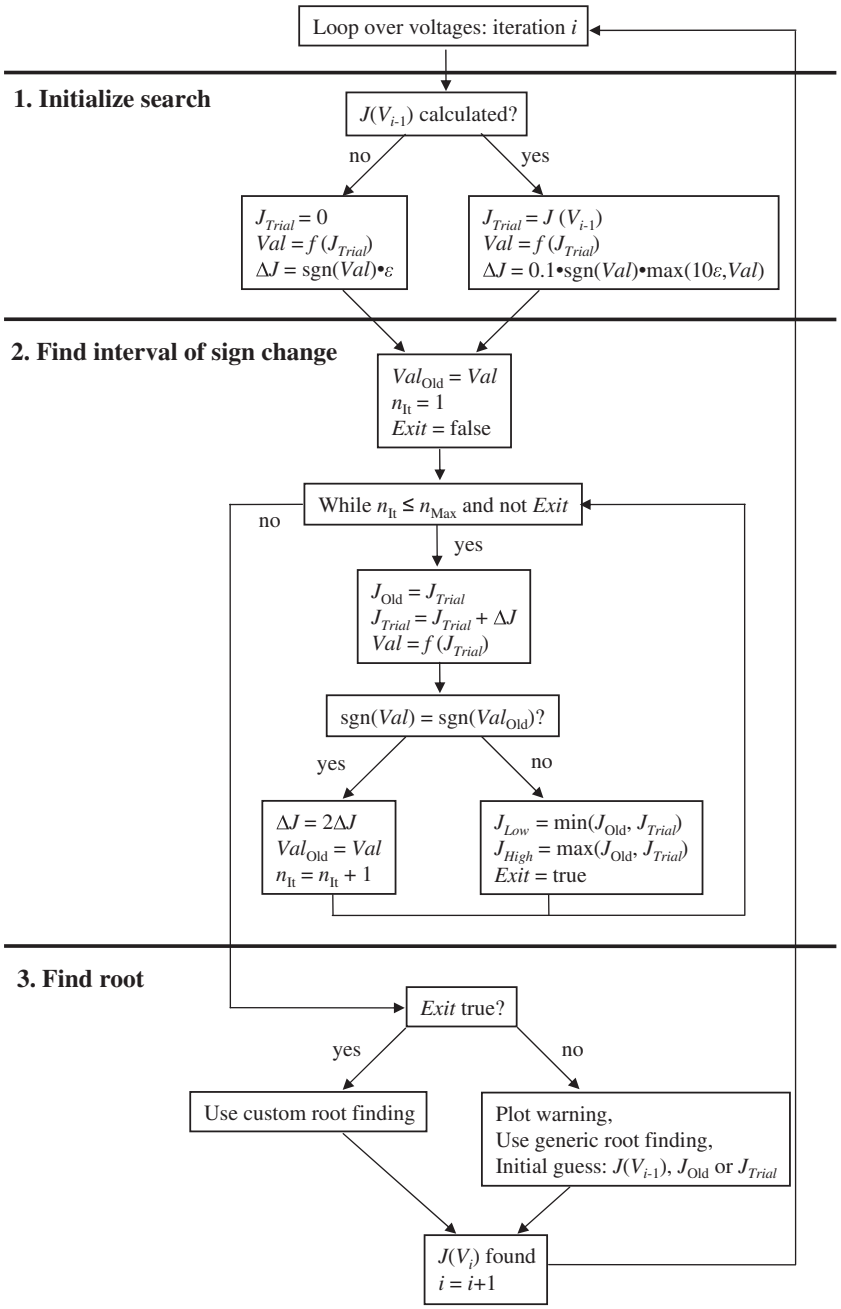

Figure 2. Flow chart of the algorithm to find the root of function *f* and thereby calculating *J(V)*.

3.2.1. Initialize search. In the first part, an initial guess J_{Trial} for J and a step size ΔJ are selected. If J was not calculated for the previous voltage V_{i-1}, which is true for the first iteration or in case anything went wrong at the previous voltage, the initial guess $J_{Trial}=0$ is chosen. This is reasonably safe as it can only fail if there is a large negative current, which can only occur at large forward bias. The next step is a function evaluation at J_{Trial}, saved into *Val*, followed by setting the magnitude of ΔJ to the smallest possible value: ε, the floating-point relative accuracy (2.2×10^{-16} for double precision). Its sign is set to the sign of *Val* using the signum function sgn, that is, if *Val* is positive, the search has to increase J_{Trial} and vice versa.

In the case that $J(V_{i-1})$ exists, this value is chosen as the current J_{Trial} (instead of 0), which is an excellent initial guess as J usually changes slowly between voltage points. Again, f is evaluated at J_{Trial}, the result is saved in *Val* and sgn(*Val*) determines the sign of ΔJ. The latter's magnitude is set to

10% of *Val* or at minimum ε. This larger initial step size is beneficial to performance, as the distance to the root of f is larger for larger *Val*.

Alternatively, the initial guess could be generated by neglecting R_S and solving Equation (1) directly. This should provide excellent results for cells with negligible R_S, but might fail for cells with problematic R_S.

3.2.2. Find interval of sign change. In the second part, an interval is searched where f changes its sign, on the basis of the initial guess J_{Trial} from the previous part. It mainly consists of a *while* loop. The initialization is comprised of copying *Val* into Val_{Old}, setting a Boolean variable *Exit* to *false* and setting the iteration counter n_{It}, which has to be managed manually in a *while* loop to avoid an infinite loop, in case anything goes wrong, to 1. The exit condition for the *while* loop is simply a limit to the iteration counter, 100 in our case, and the *Exit* flag. Inside the loop, J_{Trial} is copied into J_{Old}, J_{Trial} is incremented by ΔJ and a new *Val* is calculated as $f(J_{Trial})$. If the signs of *Val* and Val_{Old} agree, the search continues and ΔJ is set to $2 \times \Delta J$, *Val* is copied into Val_{Old} and n_{It} is incremented by 1. If the signs disagree, the minimum and maximum of J_{Old} and J_{Trial} are copied into J_{Low} and J_{High}, respectively, *Exit* is set to *true*, and the loop terminates.

3.2.3. Find root. In the third part, the actual solution is obtained by finding the root of f in the interval between J_{Low} and J_{High}, which was determined in the previous part. First, the status of *Exit* is checked. If it is *false*, the previous *while* loop ran into its iteration limit. Normally, this should not happen. If it does, for whatever reason, a warning is plotted and a generic root finding, using the best value known so far as initial guess, is launched. This guess can be either $J(V_{i-1})$, J_{Old} or J_{Trial}. With the parameter set given in this work, this code path should not be needed, but is included for additional safety.

In the regular case, *Exit* = *true*, a custom root finding routine based on bisection is used. This routine is going to be called often, so, using a self written routine with as little overhead as possible is much faster than using generic routines. Using a simple bisection algorithm would be sufficient to find the root of $f(J)$, as it is guaranteed to lie between J_{Low} and J_{High}. However, it was found that the root is often situated close to the initial J_{Low} or J_{High} due to the increasing ΔJ. This is the worst case for bisection and can result in over 30 iterations per voltage point and per function call. To accelerate this search, we use a combination of bisection and NR in our custom root finding routine. If f is reasonably linear in the interval considered, NR finds the root very quickly. If, on the other hand, the slope of f is very steep and rapidly changing due to the exponential terms in Equation (2), NR leads to points outside the current interval $[J_{Low}, J_{High}]$ and thus divergence. By combining both methods, convergence as good as regular bisection is guaranteed and often accelerated by NR. Testing root finding with only NR, and thus less calculation per iteration, caused severe instability because of divergence always occurring at some point during an entire curve fit.

Practically in each root finding iteration step, a new J and $f(J)$ is calculated for both methods, and the two values closest to the root are chosen as J_{Low} and J_{High} for the next iteration, so that the interval $[J_{Low}, J_{High}]$ still contains a sign change. The precision of the final result is entirely determined by the tolerance criterion of this root search. We are using a combination of a tolerance for J and $f(J)$. For the search to stop $|f(J_{Low})| < 1000\varepsilon$ or $|f(J_{High})| < 1000\varepsilon$ must be fulfilled and $|J_{High} - J_{Low}|$, that is, the search interval for J, must be smaller than $1000\varepsilon \max(|J_{\mathrm{Old}}|, |J_{Trial}|)$. The term $\max(|J_{\mathrm{Old}}|, |J_{Trial}|)$ ensures that in the case of very small J, the algorithm works at a tighter tolerance, which is important for fitting dark J–V curves.

3.3. Performance considerations

For the test case described in detail in the next chapter, the combined custom root finding is significantly faster than pure bisection, about a factor of 3.8 for 100 voltage points ($N_V = 100$), a factor of 6.4 for $N_V = 8100$ and a smooth transition between these extremes for intermediate N_V. Higher-order approximations such as Halley's method performed worse than simple NR in our testing, probably due to the increased computational demand for each iteration.

An obvious way to extract more performance is to loosen the tolerances of the root search. We have chosen very strict conditions to get very precise results. In our test case, the factor $10^3\varepsilon$ can be increased to $10^8\varepsilon$ for both tolerances, J and $f(J)$, to improve the performance at $N_V = 8100$ by approximately 25% while still keeping the average residuum, defined as the sum of squared residuals divided by N_V, similar up to six digits. In this case, the tolerance factor $10^3\varepsilon$ seems overly strict, but the algorithm is fast enough to allow extreme precision in the J–V calculation.

At this point, the algorithm is optimized well and further performance enhancements are difficult to achieve. When trying to optimize the scaling parameter 2 in $\Delta J := 2\Delta J$ (i.e., ΔJ is assigned twice its previous value) a trade-off exists between increasing it to accelerate the search for the interval of sign change and decreasing it to make the result obtained more precise to reduce the number of iterations required in the actual root finding. A larger value also makes the search more likely to encounter infinite function values and thus less stable. We have seen performance drop by approximately 1% for parameter values of 1.5 and 3, so the choice of 2 is a good compromise.

One could also try to optimize the choice of the initial ΔJ for the first voltage. It is chosen very small and thus up to 50 iterations may be needed to find the interval of sign change. However, choosing a larger initial step size does not lead to a measurable performance benefit, as this is only relevant once per entire J–V curve. For large N_V, when performance starts to matter, the influence of this parameter diminishes. A larger value would also make the algorithm more likely to encounter infinite function

values and thus less stable, so choosing the smallest possible initial step size is a good choice.

The last parameter requiring attention is the initial ΔJ for V_i with $i > 1$. All results presented are obtained at an initial $\Delta J = 0.1 f(J(V_{i-1}))$. In our test case, the scaling factor can be reduced to yield better performance. At $N_V = 8100$, a speed-up of 11% was obtained at 10^{-3}, with the performance benefit leveling off afterwards and eventually becoming negative. However, in this case, the performance impact of this parameter is exaggerated as N_V is unusually large, which in turn means the difference between J at neighboring voltage points is unusually small. For a general purpose curve fitting tool, 10^{-1} is a reasonable choice, but 10^{-2} may yield slightly better performance.

The entire current calculation is implemented in MATLAB, so that a performance increase is expected if the same algorithm is coded in a fast compiled language (e.g., C or C++).

4. RESULTS AND DISCUSSION

To demonstrate the performance of our code, we fitted the *J–V* curve of an ordinary mc-Si solar cell with a fill factor (*FF*) of 78.3% and no remarkable features illuminated by a cw lamp under AM 1.5 conditions at 1000 W/m^2. It is well suited to fitting by the two-diode model and thus shifts the focus from the actual fit algorithm to the current calculation. To challenge our current calculation algorithm and to demonstrate its scalability, we increased N_V by linear interpolation from 100 up to 8100, so that the same data is fitted regardless of N_V.

The measurement data and fit curves are shown in Figure 3 for the case of $N_V = 100$. The simple one-diode model with $n_1 = 1$ is included to show the benefit of using the two-diode model. The fit using two diodes and n_2 fixed at 2 is already quite good, but visibly improved upon setting n_2 as a free fit parameter ("floating"), resulting in $n_2 \approx 3.0$. The average residuum improves by approximately one order of magnitude upon each of these refinement steps. Further examples of fitting results including parameters can be found in References [18,19].

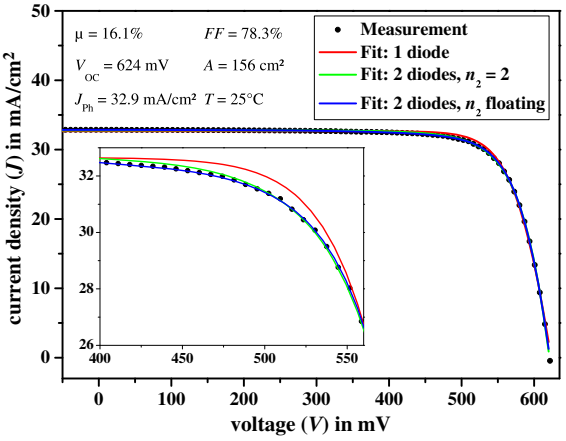

Figure 3. Measured *J–V* characteristic with $N_V = 100$ and fit curves obtained using different models. "Fit: two diodes, n_2 floating" is the one for which the run times are reported. The region where the second diode is needed to fit the curve is magnified in the inset. Cell efficiency is η, open circuit voltage V_{OC}, cell area A and the temperature is T.

The actual curve fitting was performed using the Levenberg–Marquardt (LM) [20,21] optimization built into MATLAB 2011a with a reasonable starting guess listed in Table I. Nine fit iterations are necessary to satisfy the fit tolerances, defined according to the *optimset* instruction in MATLAB 2011a, of 10^{-5} for $f(J)$ and 10^{-11} for J. The fit parameters were J_{01}, J_{02}, J_{Ph}, R_P, R_S, and n_2, leading to 71 current calculations per curve fit. Using this number, the approximate time per *J–V* calculation can easily be deduced, so that run times under different conditions (i.e., different initial guess, different sample, different fit algorithm) can be estimated if the necessary number of current calculations is known. For example, if the fit tolerance is set to ε for J and $f(J)$, 20 fit iterations are required and the overall time for the fit increases linearly to 20/9 of the values listed in Table II, whereas the average residuum remains constant up to at least six digits. The high performance of the fit routine thus enables extreme precision. However, the practical applicability of the results is always limited by the ability of the two-diode model to accurately describe the *J–V* characteristic and measurement uncertainties in J and V. This noise is not taken into account in the employed simple least-squares based fit, but could be included in more sophisticated fitting algorithms [22].

We obtained our performance data on two different systems, characterized by the central processing units (CPU) CPU1 and CPU2, respectively. CPU1 is an Intel Core i7 2600 at 3.9 GHz, that is, a current high end processor featuring four cores and supporting up to eight concurrent threads via Hyper-Threading. CPU2 is an Intel Core 2 Duo T8300 clocked at 1.2 GHz. It features two cores, can run two concurrent threads and is deliberately clocked down to simulate an older machine.

Results are included for serial execution of all current calculations as well as for a parallelized LM curve fit.

Table I. Solar cell parameters for the initial guess and after the fit with "n_2 floating" as depicted in Figure 3. The starting value of J_{Ph} was automatically extracted at $V \approx 0$. The fit improves the average residuum by 3.2×10^4.

	Initial guess	Fit
J_{Ph} in mA/cm^2	32.856	32.863
J_{01} in A/cm^2	1×10^{-12}	7.565×10^{-13}
J_{02} in A/cm^2	5×10^{-6}	8.580×10^{-7}
n_2	3	2.937
R_S in Ω/cm^2	0.5	0.451
R_P in Ω/cm^2	1500	2864

Table II. Run times in seconds for exemplary curve fit (Figure 3) with different numbers of voltage points, averaged over five runs. The errors given are the standard deviations. Refer to Section 4 for details of the different configurations.

N_V	CPU1	CPU1	CPU2	CPU2	CPU1
	serial	parallel	serial	parallel	old code
100	0.23 ± 0.01	0.33 ± 0.03	0.67 ± 0.04	0.86 ± 0.01	1.48 ± 0.02
300	0.66 ± 0.02	0.56 ± 0.04	1.62 ± 0.01	1.51 ± 0.04	5.32 ± 0.01
900	1.75 ± 0.01	1.02 ± 0.01	4.15 ± 0.03	3.20 ± 0.02	15.8 ± 0.01
2700	5.02 ± 0.04	2.78 ± 0.05	11.4 ± 0.1	7.38 ± 0.03	47.7 ± 0.08
8100	14.3 ± 0.1	6.51 ± 0.1	32.5 ± 0.2	20.5 ± 0.3	143 ± 0.5

During each fit iteration, the LM algorithm performs a trial step into each possible direction in the search parameter space, that is, six function evaluations in our case, decides which direction to take using this data and performs the step, requiring another function evaluation leading to a total number of seven. For the parallel results, we modified the LM algorithm built into MATLAB to execute all trial steps in parallel.

All performance results are contained in Table II, and a selection is depicted in Figure 4. The voltage scaling for serial execution on CPU1 and CPU2 is slightly sub-linear, as calculating J at the first voltage takes considerably more time than at the following voltages. There is also the runtime of the LM fit algorithm itself, which is largely independent of N_V and can be estimated at $N_V \approx 0$. It is comparably small, but the routine is entirely written in MATLAB and could thus be optimized further by implementing it in, for example, C++. Due to the almost linear scaling, the performance at $N_V = 500$, which is typically used in J–V measurement setups with flash lamps, can be interpolated easily. The high performance CPU1 is able to calculate the fit in less than 1 s at this complexity level.

Performance of the serial code scales approximately linear with CPU frequency. In the case of the parallel execution, there is an additional overhead associated with distributing the individual tasks and gathering the results. Therefore, this code is slower at $N_V = 100$. However, once the actual function evaluations are started, their total execution time is lower and thus the slope of the voltage scaling curve is flatter. At $N_V = 8100$, the performance increase due to parallelization is a factor of 1.6 for CPU2 and 2.6 for CPU1. The result for CPU2 is remarkably close to the theoretical maximum, which is "# serial current calculations per LM iteration" divided by "(# search parameters / # CPU cores) + 1", which works out to be $\frac{7}{6/2+1} = 1.75$. In the case of CPU1, the scaling is better than the maximum for a simple quad core ($\frac{7}{2+1} = 2.3$) due to Hyper-Threading, but further off the limit for a real 6+ core CPU ($\frac{7}{1+1} = 3.5$).

Performance results of an old version of our code on the fastest hardware are also included. It is considerably less stable because of a less careful search for the interval of sign change and uses the function *fzero* built into MATLAB for the root finding. At $N_V = 100$, the execution time of this routine may still be acceptable on high end CPUs, but our optimized routine can fit the curve with $N_V = 900$ in the same time. We also tried to generate an absolute worst case of an unoptimized approach by calling *fzero* with just the initial J_{Trial} as starting guess. However, this approach was not able to generate a single fit, as the root search quickly aborted because of encountering $-\infty$ as function value.

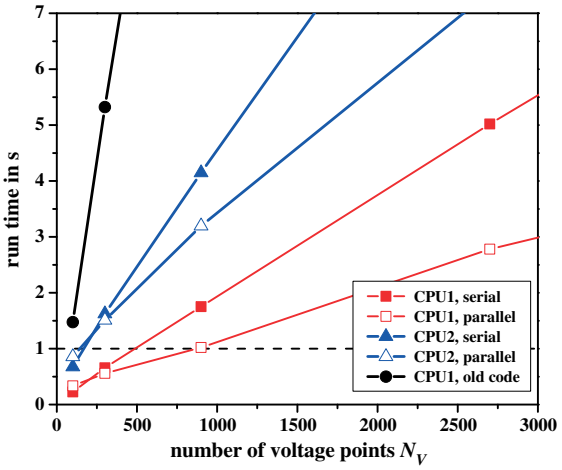

Figure 4. Run times in seconds for exemplary curve fit (Figure 3) with different numbers of voltage points, averaged over five runs. Plot of some of the data in Table II. The dashed line drawn at 1 s indicates where performance is acceptable for "real-time" applications. Refer to Section 4 for details of the different configurations.

5. CONCLUSION

We demonstrated that with our highly optimized current calculation algorithm, solar cell J–V curve fitting using the full two-diode model with six fit parameters is possible within a few seconds for J–V curves featuring a few hundred voltage points, even on readily available older hardware. This high performance negates the need for any simplifications in the two-diode model or estimation approaches and may even allow implementing extensions of the two-diode model (e.g., [23–25]).

For interactive fitting with many voltage points, the parallelized LM search is favorable, whereas for automated batch processing, a higher efficiency can be achieved by running as many serial searches for different cells in parallel as the machine supports threads. The faster each

fit is, the less hardware is needed for a given cell throughput. Using our algorithm routine, two-diode parameter extraction at in-line speeds becomes feasible. This additional information can be used to enhance cell classification and matching, in order to improve module efficiency.

Our current calculation algorithm can be used in conjunction with an LM optimization or any other fit routine. It has proven to be very stable in internal testing, even for rather unusual *J–V* curves. With the parameters given in this paper, the performance is close to optimal for this algorithm, and it has been shown where and how much additional performance can be gained by tuning the parameters at the cost of precision or stability. Further optimization is possible by using a faster programming language than MATLAB.

ACKNOWLEDGEMENTS

The authors are indebted to E. F. R. Stegemann (now with Q-Cells) for the "old" version of the code, which formed a solid basis to improve upon. Careful reading of the manuscript by B. Berghoff and the reviewers is greatly appreciated as well as enduring beta-testing by N. Wilck and M. Thore. Further thanks go to the German Federal Ministry of Education and Research (BMBF) under Grant No. 03SF0352A, "SINOVA" for financial support and a steady supply of experimental solar cells with particularly ill-behaved *J–V* characteristics, requiring hardening our algorithm.

REFERENCES

1. Wolf M, Rauschenbach H. Series resistance effects on solar cell measurements. *Advanced Energy Conversion* 1963; **3**(2): 455–479. DOI: 10.1016/0365-1789(63)90063-8
2. Wolf M, Noel GT, Stirn RJ. Investigation of the double exponential in the current–voltage characteristics of silicon solar cells. *IEEE Transactions on Electron Devices* 1977; **24**(4): 419–428. DOI: 10.1109/T-ED.1977.18750
3. Stutenbaeumer U, Belayneh M. Equivalent model of monocrystalline, polycrystalline and amorphous silicon solar cells. *Renewable Energy* 1999; **18**: 501–512. DOI: 10.1016/S0960-1481(98)00813-1
4. Greulich J, Glatthaar M, Rein S. Fill factor analysis of solar cells' current-voltage curves. *Progress in Photovoltaics: Research and Applications* 2010; **18**: 511–515. DOI: 10.1002/pip.979
5. Enebish N, Agchbayar D, Dorjkhand S, Baatar D, Ylemj I. Numerical analysis of solar cell current–voltage characteristics. *Solar Energy Materials & Solar Cells* 1993; **29**: 201–208. DOI: 10.1016/0927-0248(93)90035-2
6. Ishaque K, Salam Z, Taheri H. Simple, fast and accurate two-diode model for photovoltaic modules. *Solar Energy Materials & Solar Cells* 2011; **95**: 586–594. DOI: 10.1016/j.solmat.2010.09.023
7. Meiying Y, Xiaodong W, Yousheng X. Parameter extraction of solar cells using particle swarm optimization. *Journal of Applied Physics* 2009; **105**(9): 094502–094510. DOI: 10.1063/1.3122082
8. Macabebe EQB, Sheppard CJ, van Dyk EE. Parameter extraction from *I–V* characteristics of PV devices. *Solar Energy* 2011; **85**: 12–18. DOI: 10.1016/j.solener.2010.11.005
9. Jervase JA, Bourdoucen H, Al-Lawati A. Solar cell parameter extraction using genetic algorithms. *Measurement Science and Technology* 2001; **12**: 1922–1925. DOI: 10.1088/0957-0233/12/11/322
10. Chegaar M., Azzouzi G., Mialhe P. Simple parameter extraction method for illuminated solar cells. *Solid State Electronics* 2006; **50**: 1234–1237. DOI: 10.1016/j.sse.2006.05.020
11. Bouzidi K, Chegaar M, Bouhemadou A. Solar cells parameters evaluation considering the series and shunt resistance. *Solar Energy Materials & Solar Cells* 2007; **91**: 1647–1651. DOI: 10.1016/j.solmat.2007.05.019
12. AlRashidi MR, AlHajri MF, El-Naggar KM, Al-Othman AK. A new estimation approach for determining the *I–V* characteristics of solar cells. *Solar Energy* 2011; **85**: 1543–1550. DOI: 10.1016/j.solener.2011.04.013
13. Bayhan H, Bayhan M. A simple approach to determine the solar cell diode ideality factor under illumination. *Solar Energy* 2011; **85**: 769–775. DOI: 10.1016/j.solener.2011.01.009
14. Suckow S. 2-Diode Fit. NanoHUB.org, URL: https://nanohub.org/resources/14300
15. Goetzberger A, Knobloch J, Voss B. *Crystalline Silicon Solar Cells*. John Wiley & Sons: Chichester, 1998, ISBN 0471971448
16. Sah CT, Noyce RN, Shockley W. Carrier generation and recombination in *P–N* junctions and *P–N* junction characteristics. *Proceedings of the IRE* 1957; **45**: 1228–1243. DOI: 10.1109/JRPROC.1957.278528
17. Breitenstein O. Nondestructive local analysis of current–voltage characteristics of solar cells by lock-in thermography. *Solar Energy Materials & Solar Cells* 2011; **95**: 2933–2936. DOI: 10.1016/j.solmat.2011.05.049
18. Pletzer TM, Stegemann EFR, Windgassen H, Suckow S, Bätzner DL, Kurz H. Gettering in multicrystalline silicon wafers with screen-printed emitters. *Progress in Photovoltaics: Research and Applications* 2011; **19**(8): 946–953. DOI:10.1002/pip.1099
19. Pletzer TM, Thore M, Suckow S, Mayer B, van Mölken J, Safiei A, Windgassen H, Bleidiessel R, Kurz H. Efficiency increase of lossy solar cells

by laser post-processing and detailed analysis of the current-voltage characteristic. *Proceedings of the 37th IEEE Photovoltaic Specialist Conference* 2011; 001662–001667. DOI: 10.1109/PVSC.2011.6186275

20. Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics* 1963; **11**(2): 431–441.
21. More JJ. The Levenberg–Marquardt algorithm: implementation and theory. *Lecture Notes in Mathematics* 1978; **630**: 105–116. DOI: 10.1007/BFb0067700
22. Burgers AR, Eikelboom JA, Schönebecker A, Sinke WC. Improved treatment of the strongly varying slope in fitting solar cell *I–V* curves. *Proceedings of the 25th IEEE Photovoltaic Specialist Conference* 1996; 569–572. DOI: 10.1109/PVSC.1996.564070
23. Kurobe K, Matsunami H. New two-diode model for detailed analysis of multicrystalline silicon solar cells. *Japanese Journal of Applied Physics* 2005; **44**(12): 8314–8321. DOI: 10.1143/JJAP.44.8314
24. Nishioka K, Sakitani N, Uraoka Y, Fuyuki T. Analysis of multicrystalline silicon solar cells by modified 3-diode equivalent circuit model taking leakage current through periphery into consideration. *Solar Energy Materials and Solar Cells* 2007; **91**: 1222–1227. DOI: 10.1016/j.solmat.2007.04.009
25. Kassis A, Saad M. Analysis of multi-crystalline silicon solar cells at low illumination levels using a modified two-diode model. *Solar Energy Materials and Solar Cells* 2010; **94**: 2108–2112. DOI: 10.1016/j.solmat.2010.06.036