Signal Processing and Robust Statistics for Fault Detection in Photovoltaic Arrays

by

Henry Braun

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2012 by the
Graduate Supervisory Committee:

Andreas Spanias, Chair
Cihan Tepedelenlioglu
Pavan Turaga

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

Photovoltaics (PV) is an important and rapidly growing area of research. With the advent of power system monitoring and communication technology collectively known as the "smart grid," an opportunity exists to apply signal processing techniques to monitoring and control of PV arrays. In this paper a monitoring system which provides real-time measurements of each PV module's voltage and current is considered. A fault detection algorithm formulated as a clustering problem and addressed using the robust minimum covariance determinant (MCD) estimator is described; its performance on simulated instances of arc and ground faults is evaluated. The algorithm is found to perform well on many types of faults commonly occurring in PV arrays. Among several types of detection algorithms considered, only the MCD shows high performance on both types of faults.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

List of Symbols

Symbol		Page
I_D	Diode current	5
I_{L}	Light-generated current	5
k_b	Boltzmann constant	5
I_{cell}	PV cell current	5
T	Temperature	5
R_{s}	Series resistance	4
R_{sh}	Parallel (shunt) resistance	4
I_{D}	Diode current	5
V_{cell}	PV cell voltage	3
I_0	Diode reverse saturation current	5
V_{D}	Diode voltage	5
q	Electron charge	5
n	Diode ideality factor	5
a	Modified ideality factor	5
N_s	Number of elements in series	5
I_{SC}	Short-circuit current	7
I_{MP}	Maximum-power current	7
w	White Gaussian noise	12
σ	Standard deviation	12
R_{arc}	Arc resistance	18
G_{arc}	Arc conductance	18
V_{arc}	Arc voltage	18
I_{arc}	Arc current	18
τ	Arc time constant	18
V_0	Steady-state arc voltage (Cassie model)	18
P_0	Steady-state arc power dissipation (Mayr model)	18
I_t	Cassie/Mayr transition point	19
G_{min}	Minimum arc conductance	19

Symbol		Page
T	Test statistic	22
γ	Detection threshold	22
$\mathbf{x}$	Detector input measurement vector	22
N	Number of PV modules in array	22
$\mathbf{x}_i$	Voltage/current measurement vector for ith module	23
$\boldsymbol{\mu}$	Population mean	23
$\mathbf{C}$	Covariance matrix	23
k	Neighborhood size	27
$\mathbf{c}$	classifier input vector	27
ϕ	Neuron activation function	28
ℓ	Iteration index	28
$\hat{\boldsymbol{\mu}}$	Estimated mean	33
$\hat{\mathbf{C}}$	Estimated covariance matrix	33
h	Subset size in MCD algorithm	33
α	Robustness parameter in MCD algorithm	33
β	Consistency factor in MCD algorithm	33

Chapter 1

INTRODUCTION

Photovoltaic (PV) electrical power generation is an active and growing area of academic research and industry. In 2009, the most recent year for which such statistics are available, a record high of nearly 1.3 peak Gigawatts of PV generation capacity were manufactured and shipped by US companies, a nearly 30% increase from 2008 [1]. Although rapid and continuous progress has been made in increasing efficiency and reducing cost of PV electric power generation, array level management of PV installations remains much the same as in previous decades. The family of technologies and protocols collectively known as the smart grid offer an opportunity to change this: with increased monitoring and communication among PV array components, significant improvements in overall array power production may be achieved.

As with other renewable energy sources, progress in PV is motivated primarily by a desire to reduce fossil fuel consumption. Increasing the share of renewables such as PV in electrical generation has clear environmental and political benefits: PV modules produce no greenhouse gasses during operation and relatively little during manufacturing, and do not require significant resources which must be imported from other countries. As such, governments around the world have subsidized the construction of PV arrays and funded research into PV technology.

Despite these advantages, however, PV technology faces several barriers which prevent it from being widely deployed. The most notable barrier is cost: in the US as of 2010, the average levelized cost of energy (LCOE) for PV electricity was $211/MWh, while the LCOE of coal was only $95/MWh [2]. PV has achieved cost parity with conventional energy only for a few special cases, such as very remote locations where fuel shipping costs are extremely high. The other barrier to widespread deployment of PV is its intermittence: PV array output is dependent on the weather, and with no effective way of storing energy, PV plants may cause stability and reliability issues for the electrical grid. Any technology which lowers cost or improves array reliability will accelerate the acceptance of PV into the mainstream of power generation. By replacing current ad-hoc methods of monitoring and fault detection and by dynamically optimizing array electrical configuration, the efficiency of PV arrays may be increased while decreasing the net cost per kWh of electricity from PV sources.

The work herein concerns the use of techniques based on information processing to improve the operation and management of PV arrays. Specifically, the problem of automated fault detection is described and several potential algorithms are considered. The specific case of a "smart PV module" which reports its state of operation and is capable of dynamically switching its electrical configuration is used. Current methods of fault detection are not rigorous, are slow, and rely on human operators. In [3] a faulty array which operated for roughly 1 month before being repaired is described. In [4] mean time to repair (MTTR) of several PV installations was calculated; all installations showed a MTTR of at least 19 days. The authors of [5], on the other hand, identify a MTTR of 3.3 days for a large-scale PV system; this is noted as being a very short time. With automated fault detection and mitigation, time to repair could be dramatically reduced, in some cases to under 1 hour. A detection algorithm based on robust statistics is presented and its performance on a variety of faults is evaluated; of the algorithms presented, only this approach achieves acceptable performance on a wide variety of faults. It is believed that this algorithm is the first of its kind, taking advantage of a smart PV array's capabilities to detect faults using statistical methods.

In Chapter 2, the basics of PV modules as electrical components is described. Chapter 3 then describes a variety of fault conditions occurring in a PV array and describes the array's behavior under these conditions. In Chapter 4, Several approaches to fault detection in PV arrays are described. Finally, Chapter 5 describes the performance of several of the algorithms on faults generated using a SPICE circuit simulator.

Chapter 2

OVERVIEW OF PHOTOVOLTAICS

Although this work deals with PV modules and arrays at a very high level, some understanding of the electrical behavior and characteristics of individual PV cells is required as a prerequisite. This chapter provides a brief overview of the PV cell as a circuit component. First, the single-diode equivalent circuit model of a PV cell is described and the current-voltage (I-V) characteristic is introduced in Section 2.1. Two popular and accurate models of PV module performance, developed at Sandia National Laboratory and the University of Wisconsin respectively, are presented in Section 2.2. Finally, the array-level topology of PV modules is described in Section 2.3.

2.1 Single-Diode Model and I-V Characteristics

Figure 2.1 shows the single-diode model, a simple and commonly used representation of a PV cell, module, or array [6]. As with a diode, the behavior of this circuit may be summarized by its current-voltage (I-V) characteristic. Figure 2.2 shows the simulated I-V and power-voltage (P-V) curves of a Sharp NT-175U1 PV module at standard test conditions (STC) of 1000 W/m^2 solar irradiance and 25 °C cell temperature. Note that the module achieves its maximum power output at 35.4 V and 4.95 A, achieved when the module is connected to a load of 7.15 Ω. This is known as the maximum power point (MPP) of a PV cell, module, or array. Modern inverters dynamically adjust the load they present to the array in order to maintain operation at the MPP, a process known as maximum power point tracking (MPPT). The study and development of MPPT algorithms is an active area of research but is beyond the scope of this work [7–10].

The component values of the single-diode model vary with environmental conditions, most notably temperature and solar irradiance. Light-generated current I_{L} is almost perfectly proportional to solar irradiance and is modeled as such. Temperature primarily affects the operation of the diode: I_{D} increases with increasing temperature, reducing the power output of the cell. Bypass diodes (not shown) are generally connected across several PV cells; this diode conducts large currents when the cell is reverse biased ($V_{\mathrm{cell}} < 0$), effectively bending the I-V curve upward at negative voltages and minimizing power dissipation and heating of the cell.

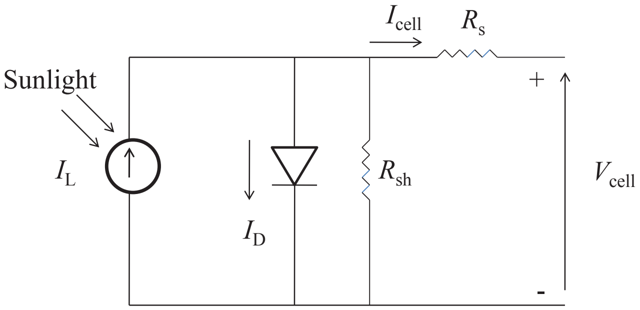


Figure 2.1: Single-diode model of a PV cell.

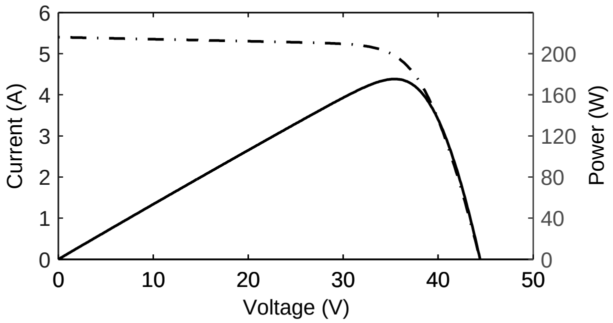


Figure 2.2: I-V and P-V curves of Sharp NT-175U1 module at STC.

2.2 PV Array Performance Models

Several models exist to predict the output of PV cells, modules and arrays as a function of environmental conditions. Two such models are presented here: the Sandia Labs model [3] and the 5-parameter model developed at the University of Wisconsin's Solar Energy Lab [6]. The Sandia model is generally more accurate, but requires extensive individual array measurements to fit model parameters to a given module. The 5-parameter model, on the other hand, is slightly less accurate but requires only data available from manufacturers' data sheets to derive model parameters.

Sandia Labs Model

The Sandia Labs PV performance model is a complex empirical model. It requires over 30 parameters to predict a PV module's I-V characteristic; these parameters must be calculated from measurements of the module or array over a variety of environmental conditions. The Sandia model's outputs are the voltage and current of the module at 5 points: Short circuit, Open circuit, maximum power point, and two intermediate points. A MATLAB implementation of the Sandia model is given in Appendix B.

5-Parameter model (University of Wisconsin)

The 5-parameter model is a simplified model developed at University of Wisconsin-Madison [6]. Unlike the Sandia model, its parameters can be fully derived from information on a PV module's datasheet. This is an important advantage since the Sandia model parameters for many more recently released modules are not available.

The 5-parameter model explicitly uses the single-diode PV cell model with parasitic resistances (Figure 2.1) to calculate the I-V characteristic. The diode is modeled using the Shockley equation:

$$I_{\mathrm{D}} = I_0(\exp\left(\frac{qV_{\mathrm{D}}}{nk_bT}\right) - 1) \tag{2.1}$$

where k_b is the Boltzmann constant, q is the electron charge, and T is the temperature in Kelvin. The behavior of the diode is then defined by I_0 and n. I_0 is known as the reverse saturation current; this is the current that the diode conducts when reverse biased. n is the diode ideality factor and primarily affects the diode threshold voltage.

The PV cell or module current I_{cell} is then given by

$$I_{\mathrm{cell}} = I_{\mathrm{L}} - I_0\left(\exp\left(\frac{V_{\mathrm{cell}} + I_{\mathrm{cell}}R_{\mathrm{s}}}{a}\right) - 1\right) - \frac{V_{\mathrm{cell}} + I_{\mathrm{cell}}R_{\mathrm{s}}}{R_{\mathrm{sh}}} \tag{2.2}$$

where$a = nk_bT/q$ is known as the modified ideality factor of the cell. In a single PV cell, n will typically take values between 1 and 2 as in the case of a silicon diode. In a module, however, n_{cell} will be multiplied by N_s, the number of modules in series, resulting in a much larger value of n and allowing the module to produce power at much higher voltages.

The variation of the circuit parameters in equation 2.2 is calculated by several physically and empirically justified means which are beyond the scope of this work. The primary drivers of PV module behavior are incoming irradiance, which is proportional to I_{L}, and module temperature, which acts through the temperature dependence of the Shockley diode equation.

2.3 Array topology

PV modules generally consist of many cells connected in series strings, with strings sometimes connected in parallel to increase current output. A similar connection scheme is used to form arrays, with series strings of modules connected in parallel to achieve the desired voltage and current. Larger arrays are divided into sub-arrays, with each sub-array feeding a different inverter. When all modules in an array are perfectly electrically matched, available array power is simply the power of each module multiplied by the number of modules and the entire array's I-V characteristic can be adequately represented by the single-diode model in Figure“ 2.1. However, when mismatch between modules exists, this is no longer accurate, and array maximum power may not be the sum of individual module maximum powers. Such mismatch is often due to a fault in the array.

Chapter 3

TYPES OF FAULTS OCCURRING IN PV ARRAYS

In an ideal solar array, array power is simply the sum of individual module powers. However, several conditions result in available DC power from the array being significantly below predicted levels. Some of these effects, such as mismatches, occur in all arrays at all times. Others are intermittent faults which are repaired as soon as they are detected. This chapter examines the effects of some of the most common non-ideal array behaviors.

3.1 Shading

Shading, the total or partial blockage of sunlight from a PV module surface, is a very serious concern in PV arrays [11–13]. It causes large performance drops and can even damage modules if not properly controlled. When a PV cell is shaded, its light-generated current I_L (see Figure 2.1) decreases, causing short circuit current I_SC and maximum power current I_MP to drop dramatically. In the extreme case of 100% shading, $I_\mathrm{SC} = I_\mathrm{MP} = 0$ A, and the PV cell acts as a diode. When a shaded cell is connected to unshaded cells in a series-parallel configuration, the shaded cell's maximum current I_SC is significantly less than the optimal current I_MP of the unshaded cells. Since each cell in the string must conduct the same current, the entire string is constrained to operate at the short-circuit current of the shaded cell, severely restricting the current and therefore power available from the remaining unshaded cells. A similar effect occurs at the module level. This effect is seen in Figure 3.2a, which shows the simulated I-V characteristics of a series string of two Sharp NT-175U1 PV modules at standard test conditions (STC) without bypass diodes, where one module has been shaded reducing its incoming irradiance to 200 W/m^2.

Bypass Diodes

In order to mitigate the effects of shading and other mismatches, bypass diodes are used (Figure 3.1). The bypass diodes D_B1 and D_B2 are connected such that under normal conditions ($V_\mathrm{cell} \geq 0$), they are reverse biased by the PV cell and conduct no current. However, when the PV cell itself is reverse biased by some external source ($V_\mathrm{cell} < 0$), the bypass diode activates, conducting large currents at low voltages and limiting the reverse voltage, and therefore power dissipation, of the shaded cell.

The heavy line in Figure 3.1 shows the dominant current flow in the case of severe shading of one cell in a 2-cell string. Shading causes I_{L2} to drop to a very low value while I_{L1} remains large. D_{B2}, the bypass diode of the shaded cell, is forward biased and conducts the additional current that is no longer able to flow through I_{L2}. Of the remaining diodes, D_{B1} and D_{C2} are reverse biased, and D_{C1} is forward biased but operates below its threshold voltage, and therefore conducts relatively small currents. In practice, a single bypass diode is usually connected across multiple cells, with similar effect.

Figure 3.2b shows the I-V characteristic of a series string of two Sharp NT-175U1 PV modules with a bypass diode for each module at standard test conditions (STC), one of which has been shaded by reducing its incoming irradiance to 200 W/m^2. In contrast to Figure 3.2a, the unshaded module is allowed to operate at or near its maximum power point (MPP). This reverse biases the shaded cell, activating its bypass diode. The effect of the bypass diode may be seen in the I-V curves of the individual modules: while the current of a module with no bypass diode is effectively limited to I_{SC} even at large reverse voltages, the addition of a bypass diode causes the combined I-V curve of the module and diode to bend sharply upward in the reverse voltage ($V_{\mathrm{cell}} < 0$) region.

The action of a bypass diode can be easily understood for a single string of two modules, but what about a full array? Figure 3.3 simulates the effect of a single shaded (500 W/m^2 irradiance) module in a 13×4 array of Sharp NT-175U1 modules at STC. Note that when bypass diodes are present, the effect of shading on output power is dramatically reduced. However, when bypass diodes are not present, current in the string where shading appears is severely limited, creating a disproportionately large drop in power relative to the shading effect. In addition to this problem, the shaded module is severely reverse biased ($V = -27V$) at the MPP while still conducting its maximum (short-circuit) current of 2.7 A. This large reverse bias and current causes the module to dissipate power, creating a hot spot which may cause damage to the array. Figure 3.3 shows another important point: Partial shading causes a disproportionately large drop in array output power relative to the decrease in irradiance. Even with bypass diodes, a 1% reduction in available input power resulted in a 3% reduction in output power due to partial shading. Without bypass diodes, this effect is even worse, causing a 12% drop in output

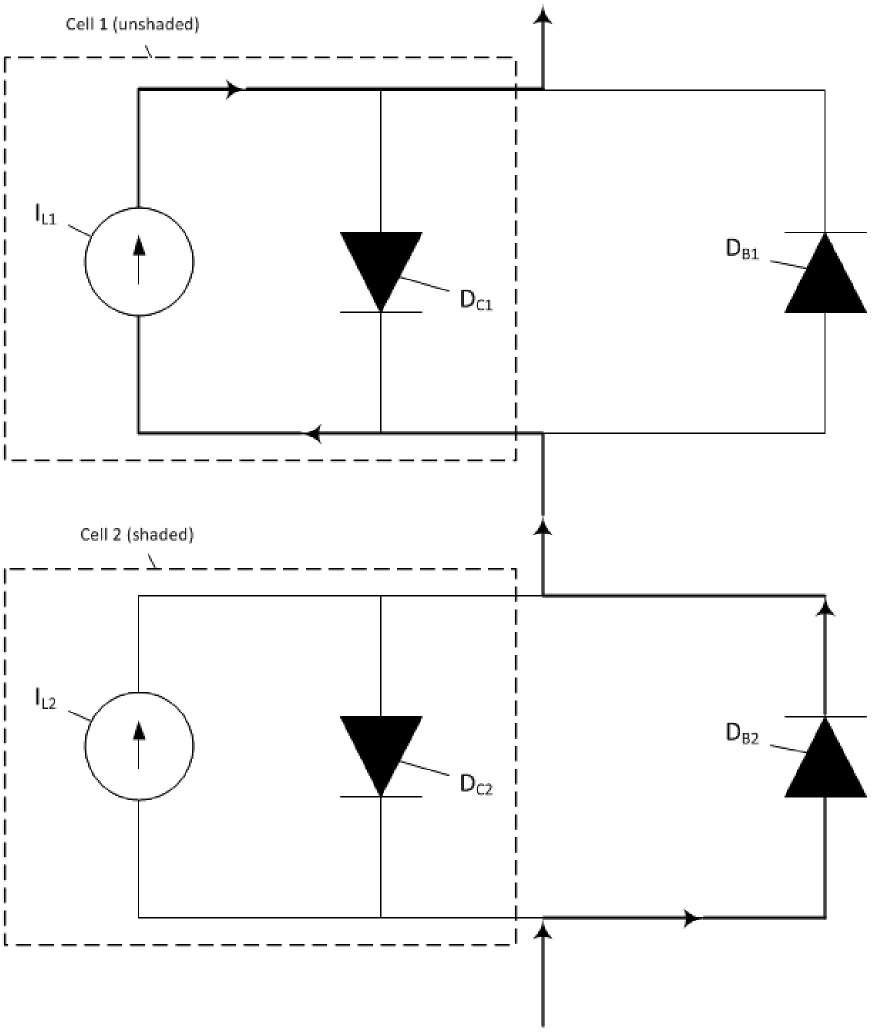
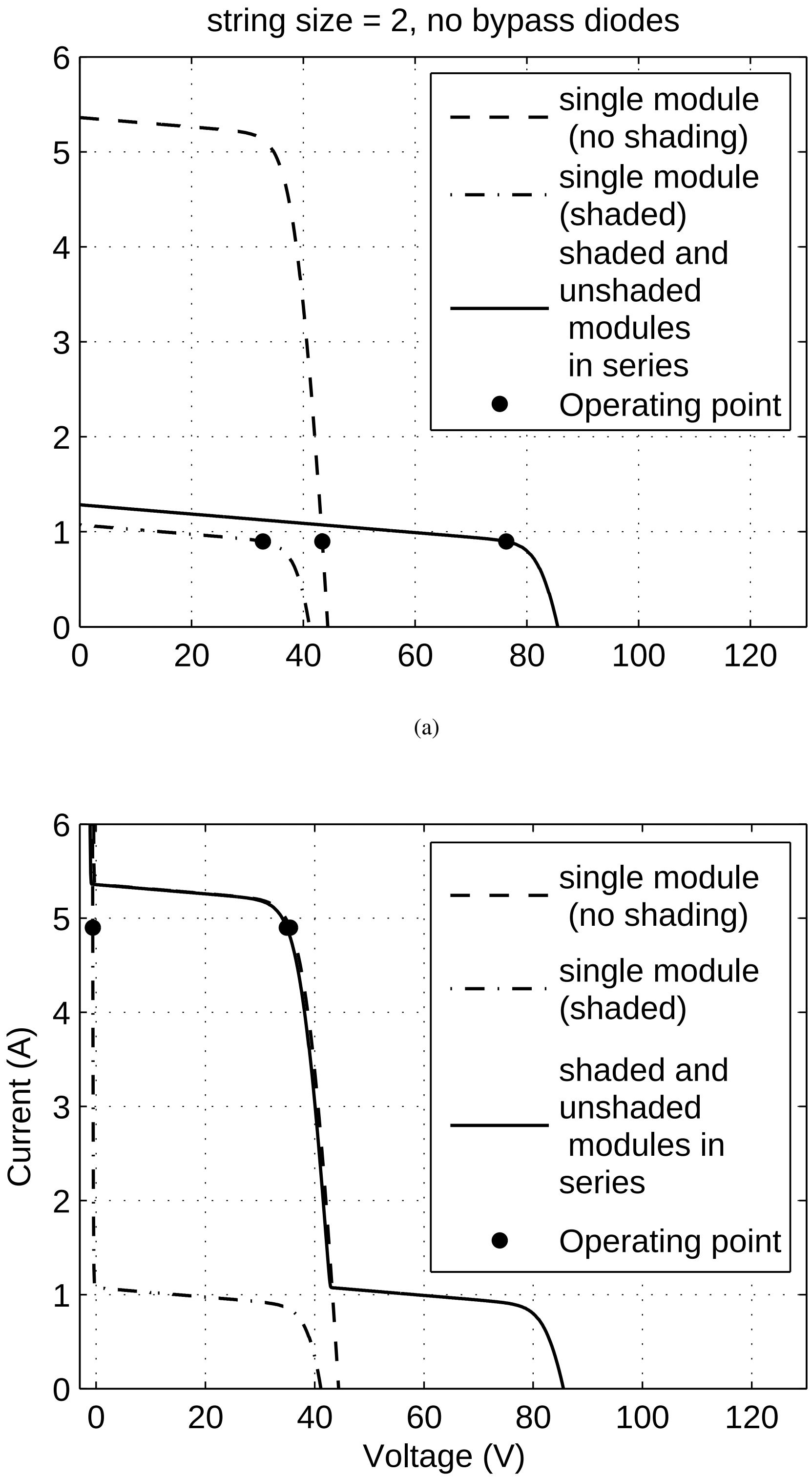


Figure 3.1: Schematic of bypass diode operation under shading conditions

power. For this reason, great care is taken in the design of PV arrays to minimize the time spent under shading conditions.

3.2 Module Mismatch

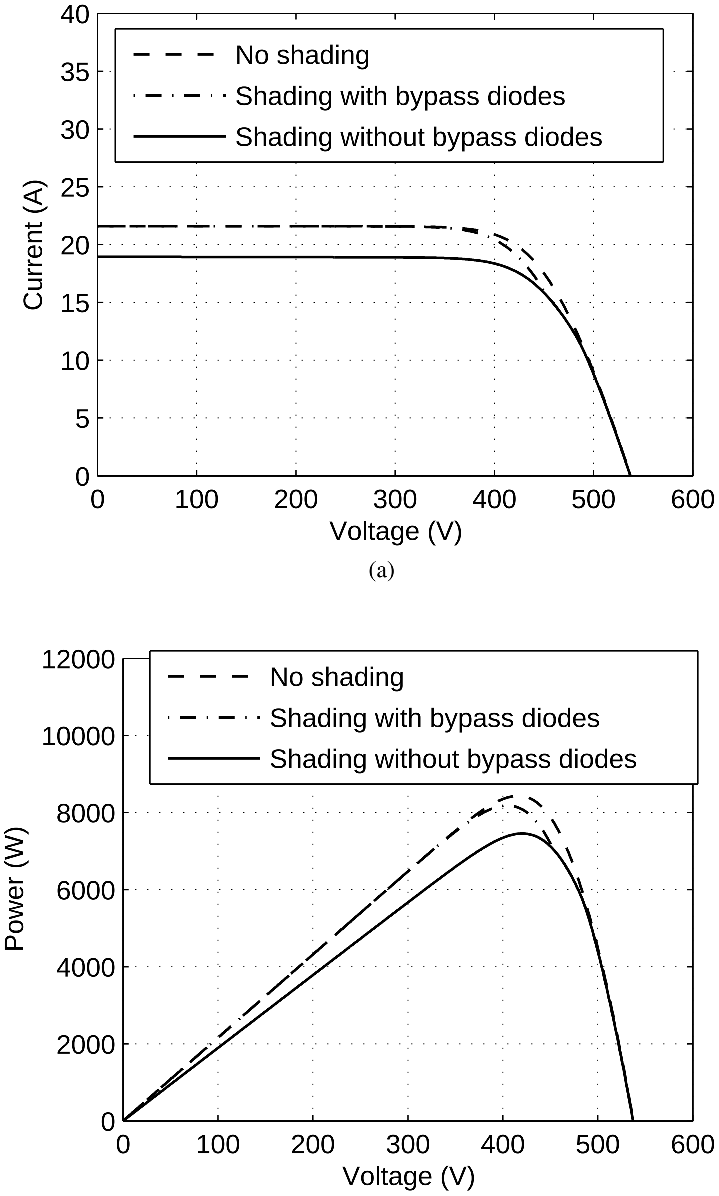
Every type of PV module has variable characteristics inevitably caused by process variation; the optimal current and voltage will not be the same for each module in an array at a given point in time. These variations have the effect of reducing the output of the array, since the current and voltage of a module are constrained by the array's electrical configuration. Module mismatch causes each module to operate at a suboptimal point on the Current-Voltage (I-V) curve, reducing the array's power output. While improvements in manufacturing processes have reduced the effect of these variations, they still must be considered [14]. This section



(a)

(b)

Figure 3.2: Effect of shaded (1/5 irradiance) module on IV curves for 2-module string (a) without and (b) with bypass diodes.



(a)

(b)

Figure 3.3: Effect of a single shaded (1/2 normal irradiance) module on (a) I-V curve and (b) P-V curve.

describes the effects of variation in each of the parameters of the single-diode model on the series-parallel (S-P) array configuration, in which modules are arranged into series "strings" that are then connected in parallel. In general, other configurations such as the Total Cross-Tied (TCT) configuration, in which layers of parallel-connected modules are connected in series, perform better [15], but are seldom implemented due to increased material costs.

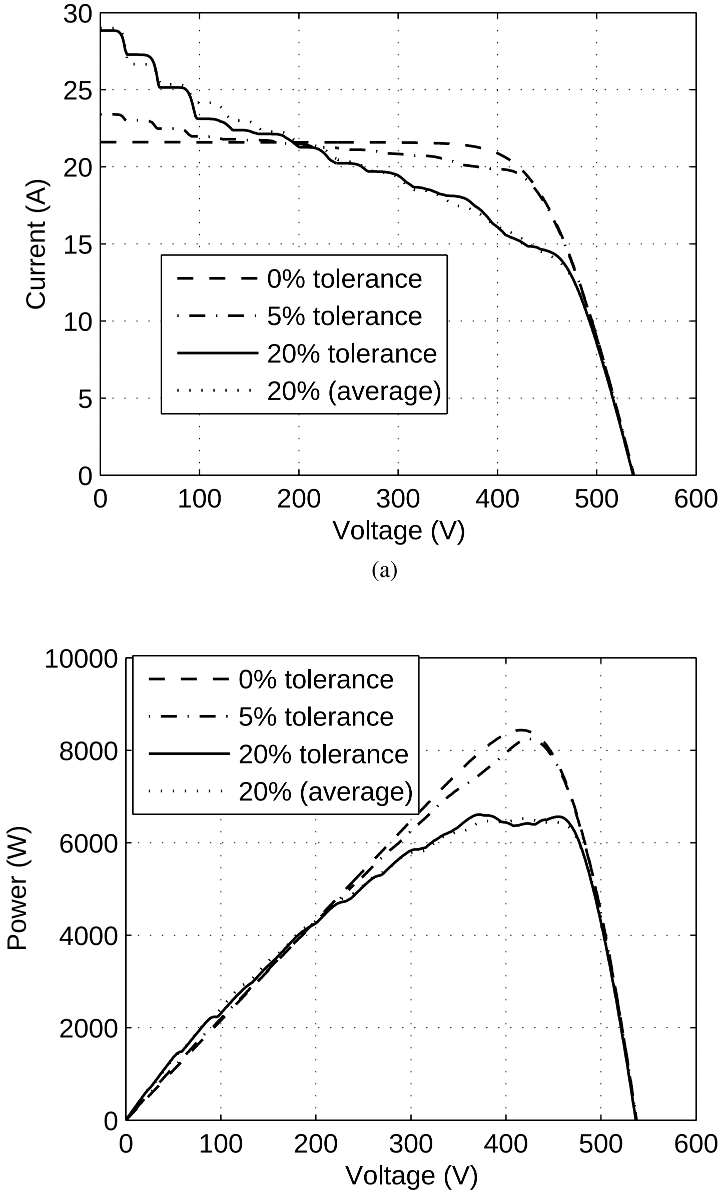
Mismatches in Current

One simple source of mismatch in a PV array is uncertainty in the light generated current I_L of each module, defined in Figure 2.1; this mismatch is especially harmful in the series-parallel configuration. The effects of additive zero-mean Gaussian variation in I_L are simulated for a 13 series, 4 parallel (13×4) array of Sharp NT-175U1 solar modules in Figure 3.4. Gaussian variation was selected because variation in $I_\mathbf{L}$ is expected to be a sum of many relatively independent factors; the author is not aware of any work which attempts to determine the distribution of modules' $I_\mathbf{L}$ values. The light current of the i-th module $I_\mathrm{L,i}$ was determined as follows:

$$I_{\mathrm{L},i} = I_\mathrm{L} + w(i) \tag{3.1}$$

where $w(i) \sim \mathcal{N}(0, \sigma^2)$, a Gaussian with mean 0 and variance σ^2, for the cases of $\sigma = 0.2 I_\mathrm{L}$ and $\sigma = 0.05 I_\mathrm{L}$. The 20% tolerance ($\sigma = 0.2 I_\mathrm{L}$) case has been selected in order to highlight the effect of mismatches; modern PV modules have a (not necessarily Gaussian) tolerance of I_L on the order of 3%–10%, causing a power loss of up to 4% relative to a perfectly matched array [14].

It can be seen that in every case the presence of mismatches reduces the power output of the array. This is due to the inherent limitations of the electrical configuration. Each module in a string must carry the same current, forcing the individual modules to operate at a slightly sub-optimal point on their I-V curve. A similar but less important effect occurs between strings, which must each develop the same voltage. An interesting effect is visible in the case of 20% standard deviation in the S-P configuration: The I-V curve develops corners, in turn causing local maxima in the power-voltage (P-V) curve. These corners appear when one or more modules becomes reverse biased, activating its bypass diodes and effectively turning it into a consumer of energy. At the maximum power point (MPP), the worst performing modules are effectively



(a)

(b)

Figure 3.4: Effect of mismatch in I_{L} on (a) I-V curve and (b) P-V curve.

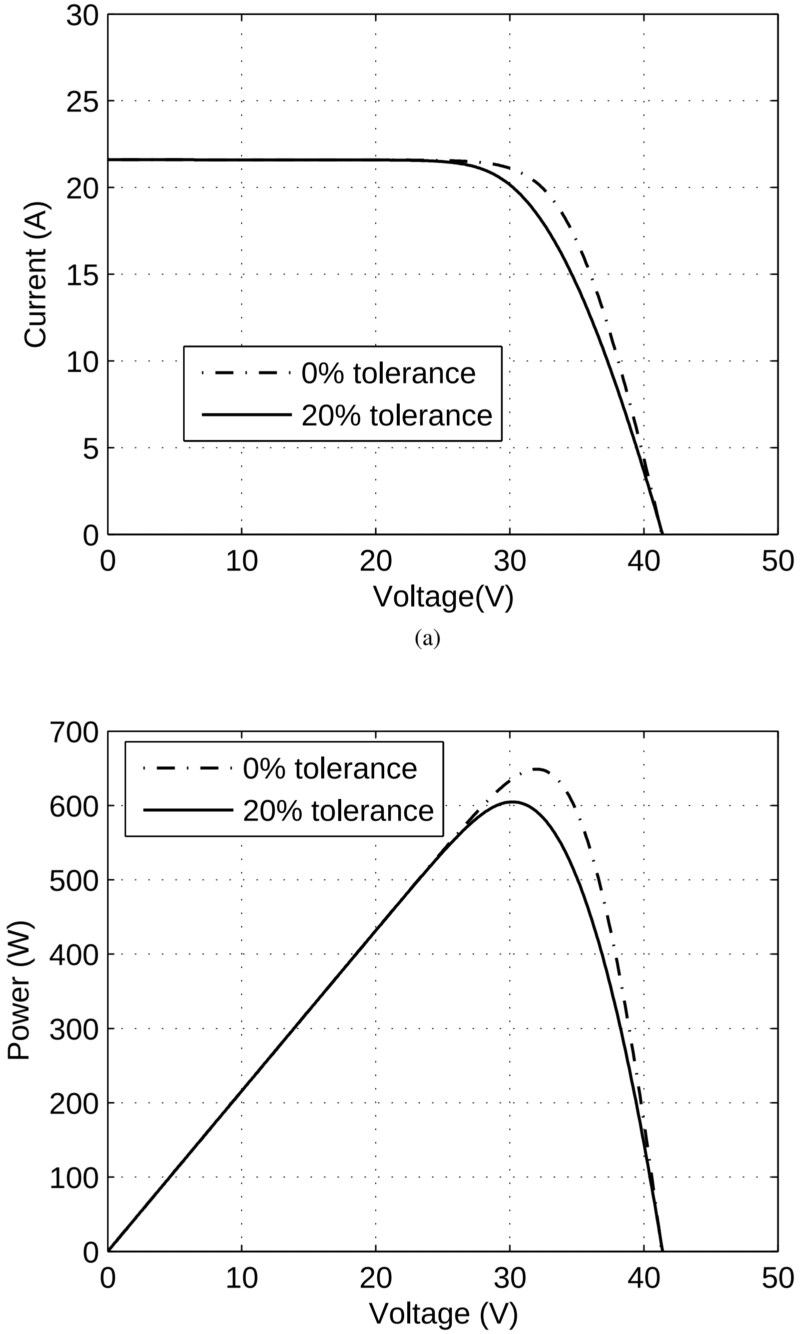
off-line due to mismatch. The effects of a zero-mean additive current mismatch are, on average, always harmful to the power output of the array. This is demonstrated in Figure 3.4b, which shows the averaged PV curve of 10 independent simulations of the $\sigma = 0.2I_{\mathrm{L}}$ case outlined above in (3.1). Note that while the presence of zero-mean variation in I_{L} does not affect the expected maximum power output of a single module, it lowers the expected maximum power of the array, in this case from 8.4 kW to 6.5 kW.

Mismatches in Diode Parameters

Uncertainty in diode parameters of saturation current I_0 and ideality factor n, defined in Chapter 2, leads to variation in the voltage of modules. Figure 3.5 shows the effect of a 20% tolerance in n and I_0 for the case of a 4-module array with all 4 modules in parallel. As in section 3.2, variation in parameters was modeled as additive Gaussian noise. This parallel configuration maximizes the effect of mismatch, since with a string size of 1, high-voltage modules are not able to compensate for low-voltage modules in the same string. In the S-P configuration, however, this variation has only a minimal effect on array power output. The voltage of a string is the sum of the voltages of its modules,and even if the maximum power voltage V_{MP} of individual modules varies significantly, the V_{MP} of a 13-module string will still have negligible variation. This is because when independent identical Gaussians are added the standard deviation of the sum grows much more slowly than the mean. Because of this, mismatches in module V_{MP} are much less important than mismatches in I_{MP}.

3.3 Module Soiling

Module soiling is the build-up of dirt on the surface of a PV module. Researchers have found that the effects of soiling are relatively small (2.3% loss of power) for directly incident light but become more significant for larger angles: an 8.1% loss was observed in a soiled module when light is incident from an angle of 56 degrees [16]. It was also found that bird droppings cause a much larger degrading effect than general soiling due to their complete blockage of light over a small area and can be treated as a partial shadow (see Section 3.1). If one module is soiled it will cause a mismatch in the effective irradiance of the modules, reducing the array performance.



(a)

(b)

Figure 3.5: Effect of mismatch in diode parameter n on (a) I-V curve and (b) P-V curve.

3.4 Ground Fault

A ground fault occurs when the circuit develops an unintentional path to ground. This results in lowered output voltage and power, and can be fatal if the leakage currents are running through a person. AC Ground fault circuit interrupters (GFCIs) are capable of detecting even very small (6 mA) leakage currents and opening a switch to stop current flow within 200 ms. Actual detection thresholds and response times vary widely by application, and there is generally a trade-off between response time and detection threshold. Typical AC GFCIs work on the principle of detecting mismatches between incoming and outgoing current:all wires going to and from a voltage source pass through the sensing opening of a current transducer. Under normal operation, all currents must pass through the transducer, meaning that the net current through the transducer is zero. Under fault conditions, some current does not pass through the detector, instead flowing through the faulty path and creating a non-zero net current through the transducer's opening. When this current is detected, a relay is quickly opened and all current flow is stopped [17] (Figure 3.7). GFCIs are a mature technology.

In a PV array, however, The existence of multiple power sources and potential fault locations means that a traditional GFCI which simply detects net current flow will be unable to detect many potential fault scenarios. Instead, ground fault detection in a PV array typically takes the form of a simple over-current detector with a detection threshold of 0.5 to 1 A. When a fault occurs, extra current in the grounding system is detected and the circuit is opened. GFCIs are mandated by the 2008 United States National Electric Code for all PV systems [18]. Figure 3.6 shows the effect of a severe ground fault on the currents and voltages of modules in a series-parallel configuration.

3.5 DC Arc Fault

Direct Current (DC) arcing is a spark across air or another dielectric and occurs in two forms: series and parallel arcs [19–21]. A series arc most often occurs when a connection breaks, leaving two conductors very near each other. Series arcs can occur in junction boxes, at the cable connections between modules and within modules. Parallel arcs can occur when two conductors of different voltage are near each other, such as when the insulation of two wires running parallel to one another is compromised. The current produced by a series arc is limited

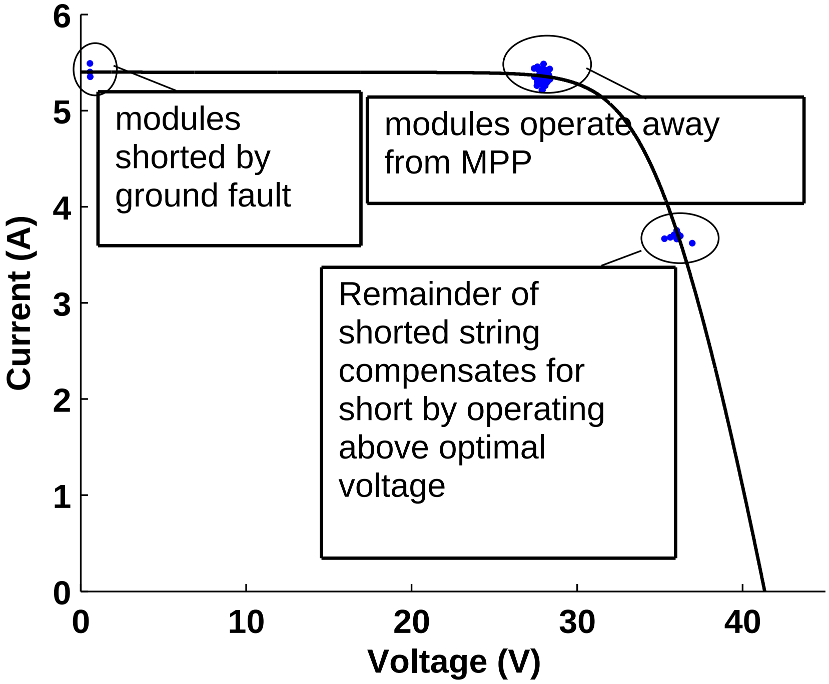


Figure 3.6: Module voltage & current during ground fault conditions.

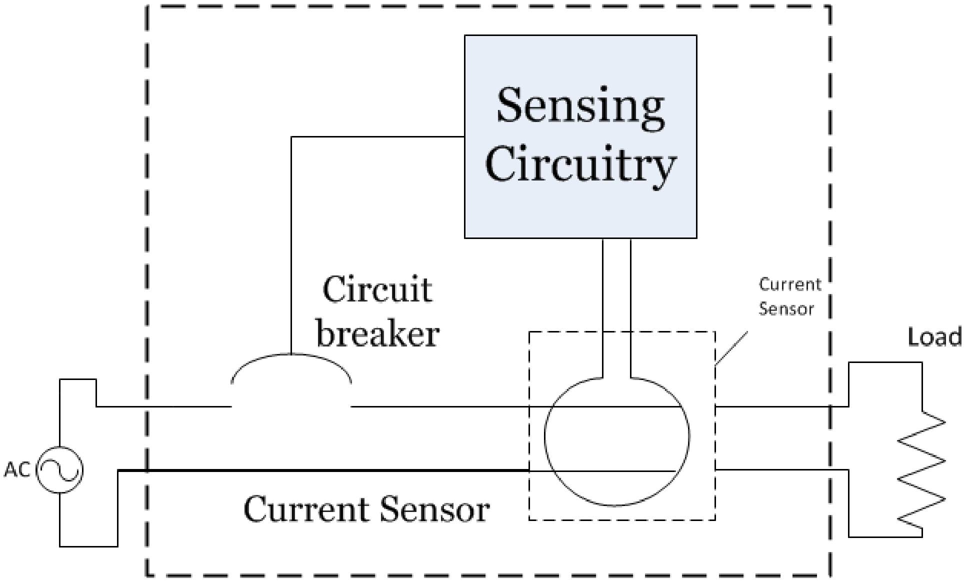


Figure 3.7: Simplified block diagram of GFCI operation.

by the load with which it is in series, while a parallel arc can consume as much current as a source is able to supply. Arcs can lead to inefficiency in array operation, frequently cause failure of bypass diodes [22], and can even cause fires. Even a relatively benign arcing incident is likely to disable the entire string in which the arc occurs.

Unlike other faults mentioned in this chapter, arc faults are an inherently transient phenomenon and cannot be fully modeled by a change in the DC behavior of an array. The DC behavior of arcs can be modeled as a nonlinear resistance [23], but more complex techniques are needed for transient behavior. Two intuitive and long-standing arc models are the Cassie [24] and Mayr [25] models, which describe an arc as a time-varying resistance. The Cassie model assumes that the arc is a cylinder of conductive plasma known as the arc column. Constant current density in the arc column, convective heat loss through its entire volume, and constant resistivity and energy storage density in the arc are also assumed. These assumptions result in an arc equation of the form

$$\frac{1}{R_{\text{arc}}(t)}\frac{dR_{\text{arc}}(t)}{dt} = \frac{1}{\tau}\left(1 - \frac{V_{\text{arc}}(t)^2}{V_0^2}\right) \tag{3.2}$$

where $R_{\text{arc}}(t)$ is the instantaneous arc resistance, τ is the arc time constant, $V_{\text{arc}}(t)$ is the voltage across the arc, and V_0 is the steady-state arc voltage. Equation 3.2 may equivalently be written in terms of the arc conductance $G_{\text{arc}}(t)$:

$$G_{\text{arc}}(t) = \frac{V_{\text{arc}}(t)I_{\text{arc}}(t)}{V_0^2} - \tau\frac{dG_{\text{arc}}(t)}{dt}. \tag{3.3}$$

where $I_{\text{arc}}(t)$ is the current through the arc and other quantities are defined as before. At steady state, the Cassie model implies a constant arc voltage regardless of current. While valid for large currents, the model breaks down at low currents and does not allow for ignition or extinction of an arc.

The Mayr model, on the other hand, is accurate at lower currents and accounts for arc extinction. It assumes constant power dissipation, P_0, from the edge of the arc column and assumes the conductance of the arc to be proportional to the energy stored in the arc column. The differential equation defining the Mayr arc model is

$$\frac{1}{R_{\mathrm{arc}}(t)}\frac{dR_{\mathrm{arc}}(t)}{dt} = \frac{1}{\tau}\left(1 - \frac{V_{\mathrm{arc}}(t)I_{\mathrm{arc}}(t)}{P_0}\right) \tag{3.4}$$

and may be equivalently re-written in terms of the conductance as

$$G_{\mathrm{arc}}(t) = \frac{I_{\mathrm{arc}}(t)^2}{P_0} - \tau\frac{dG_{\mathrm{arc}}(t)}{dt}. \tag{3.5}$$

At steady state, the Mayr model's I-V characteristic is a hyperbola. The model is not accurate at large currents where convective heat dissipation from within an arc becomes a significant effect.

Many related hybrid Cassie-Mayr arc models have been proposed [26] in an attempt to more completely describe the arc characteristics. In [23], a weighted sum of the Cassie and Mayr arc conductances is taken. The weights vary as a function of the arc current, with a very small minimum conductance G_{min} added to allow for self-ignition of the arc. The conductance of this hybrid model is given by:

$$\begin{aligned} G_{\mathrm{arc}}(t) = \quad & G_{min} + \left[1 - \exp\left(\tfrac{-I_{\mathrm{arc}}(t)^2}{I_t^2}\right)\right] \tfrac{V_{\mathrm{arc}}(t)I_{\mathrm{arc}}(t)}{V_0^2} \\ & + \exp\left(\tfrac{-I_{\mathrm{arc}}(t)^2}{I_t^2}\right) \tfrac{I_{\mathrm{arc}}(t)^2}{P_0} - \tau\tfrac{dG_{\mathrm{arc}}(t)}{dt}, \end{aligned} \tag{3.6}$$

where I_t is a constant representing the transition point between the two models and other parameters are defined as before. Another common hybrid model is the Habedank model, which defines a Cassie arc in series with a Mayr arc. Figure 3.8 shows the steady-state current-voltage characteristics of the Cassie and Mayr models given above, along with the Habedank model. Parameters were chosen as $G_{min} = 10^{-8}$ S, $P_0 = 30$W, and $V_0 = 28$ V. The parameter τ does not affect the steady-state I-V characteristic.

3.6 High-resistance Connections

A high-resistance connection is a small region of high resistance through which current must flow, causing unintended heating and power loss. High-resistance connections may be formed within a module during the manufacturing process, during array assembly if connections are not made tightly, or over time due to factors such as thermal stresses [27]. A high-resistance

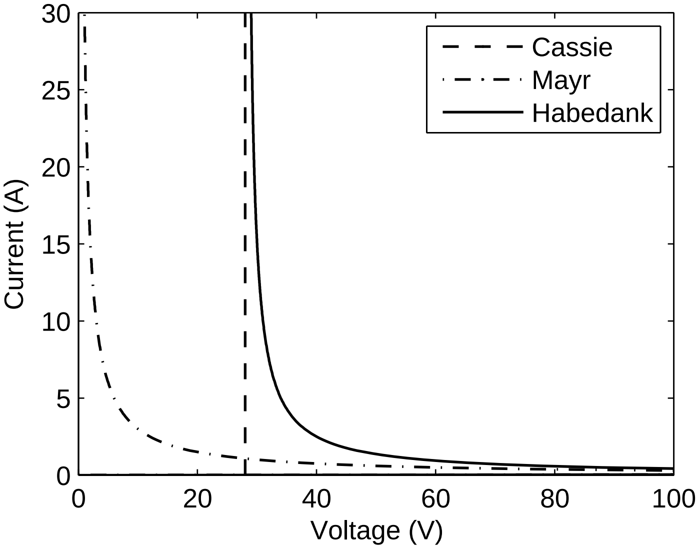


Figure 3.8: Comparison of Cassie, Mayr, and hybrid models

connection may eventually separate fully, leading to a series arc fault or open circuit. High-resistance connections increase the effective series resistance of a module, leading to reduced power output. In extreme cases, a bad connection may heat up enough to be hazardous, leading to the colloquial name "glow fault" to refer to such a connection.

3.7 Inverter Failure

Inverters are generally the weakest link in a PV array, and inverter failures are the most common cause of PV array outages. The failure of switching transistors is the most common cause of inverter failure, although degradation of electrolytic capacitors also plays a large role [28]. Several approaches are taken to mitigate the risk of inverter failure. These include designs which limit the use of electrolytic capacitors, selection of over-rated switching transistors, and the use of multiple, smaller inverters in place of a single large one. Thermal stresses are the most significant factor leading to inverter failure, so the design of cooling systems is extremely important for inverter reliability. Many modern inverters report their operating status, including efficiency, to a server for logging and monitoring. This information better allows an array operator to quickly identify and repair faulty inverters.

3.8 Islanding

The addition of arrays to the power grid poses a safety risk known as islanding. Islanding occurs when a fault in the wider electrical grid causes a power outage, but a PV array does not turn off. In this situation, a small “island” forms in the grid, receiving electrical power from the PV array. This island of energized wires poses a threat to repair technicians seeking to fix the fault. Modern inverters are, for the most part, equipped with anti-islanding features which detect when the AC grid goes down and shut off the inverter. In addition to this safeguard, many electrical companies require that residential arrays connected to their grid come equipped with a manual disconnect switch, to be opened by a power company technician before beginning work on nearby parts of the grid.

3.9 Summary

The preceding sections have given an indication of the types of faults a PV array experiences and the effect they have on array IV curves. Throughout the cases examined it was found that faults occurring in individual modules have far-reaching effects on array behavior: PV modules are not independent units whose performance can be easily compared to one another. It is this interdependence of module output in the series-parallel configuration which makes the problem of detecting, localizing, and diagnosing faults non-trivial.

Chapter 4

DETECTION ALGORITHMS

In this chapter, several methods for detection of faults are considered. First, previous work from the PV area is described. Then, basic classical distance-based methods are examined. Next, the distance-based methods are adapted to the PV area by the introduction of robust statistics. Finally, several other methods of detection, such as machine learning based methods, are briefly considered.

4.1 Basic Detection Theory

In general, all binary detectors may be viewed as comparing a test statistic $T(\mathbf{x})$ to a threshold γ in order to arrive at a decision on the presence (s_1) or absence (s_0) of the condition to be detected:

$$T(\mathbf{x}) \gtrless_{s_0}^{s_1} \gamma \tag{4.1}$$

In a binary detector there is an inherent trade-off between detection rate and false alarm rate. By setting $\gamma = \infty$, no false alarms will be detected, but all true instances of the event will also go unnoticed. Likewise, by setting $\gamma = -\infty$ the event will always be detected, whether or not it actually occurred. Between these two extremes, different detectors may be superior in different applications.

4.2 Problem statement

The problem of identifying the presence of a fault in a PV array may be formulated as an outlier or anomaly detection problem: the presence of an outlier in I-V data indicates a fault. Anomaly detection is broadly defined as the practice of identifying subsets of data which do not exhibit the same patterns as the rest of the set. It is an active area of research with several successful commercial applications. The most notable commercial successes are probably fraud detection algorithms, which monitor financial transaction data for events such as identity theft, and intrusion detection algorithms, which attempt to identify unauthorized use of computer networks. With the availability of higher - resolution data from PV modules and inverters, an opportunity exists for anomaly detection methods to be adapted and applied to PV technology.

For purposes of this discussion, it will be assumed that the algorithms described accept pairs of voltage and current measurements from N individual PV modules as input, denoted

$\mathbf{x}_i(t) = \begin{bmatrix} V_i(t) & I_i(t) \end{bmatrix}^T$, where $1 \leq i \leq N$ is the index of an individual module and t is time. Since values of V and I change quickly with weather variation, only data from a single time t is considered in the algorithms; dependence on t will be omitted from the following discussion. Thus, the input data may be visualized as a set of points in I-V space as in Figures 5.5 and 5.2. Most of the algorithms presented are not restricted to 2-dimensional $\mathbf{x}_i$ and can easily be adapted to incorporate a wider variety of measurements, for instance module temperature.

4.3 Likelihood Ratio Test

The extremely well-known Neyman-Pearson theorem [29] states that in a two-class detection problem, the optimal detector is one whose test statistic is based on the ratio of the likelihoods of the two classes. That is, given two possible classes, s_0 and s_1, and a received observation vector $\mathbf{x}$ with corresponding PDFs $p(\mathbf{x}|s_0)$ and $p(\mathbf{x}|s_1)$ the test statistic

$$T(\mathbf{x}) = \frac{p(\mathbf{x}|s_1)}{p(\mathbf{x}|s_0)} \gtrless_{s_0}^{s_1} \gamma \tag{4.2}$$

will achieve the highest possible detection rate for a given false alarm probability. For the problem of fault detection in PV arrays s_0 represents normal operation and s_1 represents the presence of a fault.

Because it is proven to be optimal, the likelihood ratio test is a natural first choice for implementation of a detection algorithm. However, in this case, at least two complications prevent its implementation. First, although $p(\mathbf{x}|s_0)$, the PDF of the observed data under no-fault conditions, may reasonably be approximated as a Gaussian random variable, no such approximation of $p(\mathbf{x}|s_1)$ may be feasibly developed. Second, the distribution of $p(\mathbf{x}|s_0)$, even if known to be Gaussian, depends on unknown parameters $\boldsymbol{\mu}$ and $\mathbf{C}$, its mean and covariance matrix.

The second issue may be dealt with in at least two ways. First, $\boldsymbol{\mu}$ and $\mathbf{C}$ may be estimated from measurements of environmental conditions such as incoming solar irradiance or ambient temperature. If this is done, the effect of model mismatch and measurement error on the estimates of $\boldsymbol{\mu}$ and $\mathbf{C}$ must be considered; they are random variables correlated to the true values of the parameters.

Another way to deal with the unknown parameters is to apply the generalized likelihood ratio test (GLRT) to the problem. The GLRT extends the test in 4.2 to the case where PDFs depend on an unknown parameter vector $\boldsymbol{\theta}$. Its test statistic and decision rule are given by

$$T(\mathbf{x}) = \frac{p(\mathbf{x}|\hat{\boldsymbol{\theta}}, s_1)}{p(\mathbf{x}|\hat{\boldsymbol{\theta}}, s_0)} \gtrless_{s_0}^{s_1} \gamma \tag{4.3}$$

where $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate of the parameter vector $\boldsymbol{\theta}$. Unlike the Neyman-Pearson detector in equation 4.2, the GLRT is not an optimal test statistic. However, it often shows good performance and is widely used [29].

The lack of a good model for $p(\mathbf{x}|s_1)$, the distribution of the data under fault conditions, is a much more serious problem. Until measurements from PV array monitoring systems are available, it is not feasible to estimate this PDF. Although the likelihood ratio test cannot be implemented for this problem, one of the detectors considered for simulations in Chapter 5 is based on $p(\mathbf{x}|s_1)$.

4.4 Distance-based test Statistics

Most of the test statistics considered in this section rely on some measure of distance from the center of a distribution. Anomalies will have a large distance, and with the selection of an appropriate threshold a detector may be designed. This section describes several common distance-based detection schemes and describes their advantages and disadvantages.

The Euclidean Distance

Euclidean distance is the traditional measure of distance from geometry, extended to n-dimensional space. In our case, $n = 2$, with current and voltage comprising the dimensions of the data. The Euclidean distance between two points $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as

$$d_{EU}(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_2 - \mathbf{x}_1\|_2 = \sqrt{(\mathbf{x}_2 - \mathbf{x}_1)^T(\mathbf{x}_2 - \mathbf{x}_1)}. \tag{4.4}$$

In an anomaly detection context, $T_i(\mathbf{x}_i) = d_{EU}(\mathbf{x}_i, \boldsymbol{\mu}_x)$, where $\boldsymbol{\mu}_x$ is the mean of the data, may be used as a test statistic to determine whether $\mathbf{x}_i$ is an anomaly. $T(\mathbf{x}_i, ...\mathbf{x}_N) = \max_i(T_i(\mathbf{x}_i))$ is then a useful and intuitive test statistic to determine whether or not any anomalies are present.

Euclidean distance is extremely intuitive and is an appropriate distance metric for a few well-scaled problems where the elements of $\mathbf{x}_i$ are relatively uncorrelated. However, Euclidean distance is not scale-invariant. If one element of $\mathbf{x}_i$ measures current on the order of amps and another measures voltage on the order of hundreds of volts, the Euclidean distance will depend almost entirely on the difference in voltage between two data points, while ignoring potentially significant data from current measurements. Mahalanobis distance, discussed in Section 4.4, is one attempt to correct for these effects.

The Mahalanobis Distance

Mahalanobis distance is one of a family of distance metrics which attempt to overcome the problems of the Euclidean distance by replacing the ℓ_2-norm with a quadratic norm. In the case of the Mahalanobis distance, the quadratic norm is defined by the covariance matrix $\hat{\mathbf{C}}$ of the dataset:

$$d_{\mathrm{MA}}(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_2 - \mathbf{x}_1)^T \hat{\mathbf{C}}^{-1} (\mathbf{x}_2 - \mathbf{x}_1)}. \tag{4.5}$$

Compared with the Euclidean distance, Mahalanobis distance offers several important advantages. It is scale invariant - in fact, any non-singular linear transformation may be applied to the data set without altering $d_{\mathrm{MA}}(\mathbf{x}_1, \mathbf{x}_2)$: by recalculating $\hat{\boldsymbol{C}}$, the effects of the transformation will be canceled out. Another effect of this invariance is that correlations between the elements of $\mathbf{x}_i$ are automatically compensated for. Taking the Mahalanobis distance between two points is equivalent to taking the Euclidean distance after the dataset has undergone centering, decorrelation and normalization procedures.

Mahalanobis distance is commonly used to measure $d_{\mathrm{MA}}(\mathbf{x}_i, \hat{\boldsymbol{\mu}})$, the distance between the sample centroid $\hat{\boldsymbol{\mu}}$ and a point $\mathbf{x}_i$ in the dataset. $d_{\mathrm{MA}}(\mathbf{x}_i, \hat{\boldsymbol{\mu}})$ then provides a test statistic to determine whether $\mathbf{x}_i$ is an outlier. As before, $\max_i(d_{\mathrm{MA}}(\mathbf{x}_i, \hat{\boldsymbol{\mu}}))$ may be used to determine the presence or absence of outliers. Note also that for the case of 1-dimensional data ($m = 1$), $d_{\mathrm{MA}}(\mathbf{x}_i, \hat{\boldsymbol{\mu}})$ simplifies to the well-known z-score statistic. Mahalanobis distance functions well as an outlier detector when the distribution of the sample data is Gaussian (for instance) with few outliers, but is not accurate for highly skewed or multimodal distributions, or for sample sets containing a large proportion of contaminated data points (outliers). This last effect occurs because the Mahalanobis distance is not robust - the true centroid and covariance are

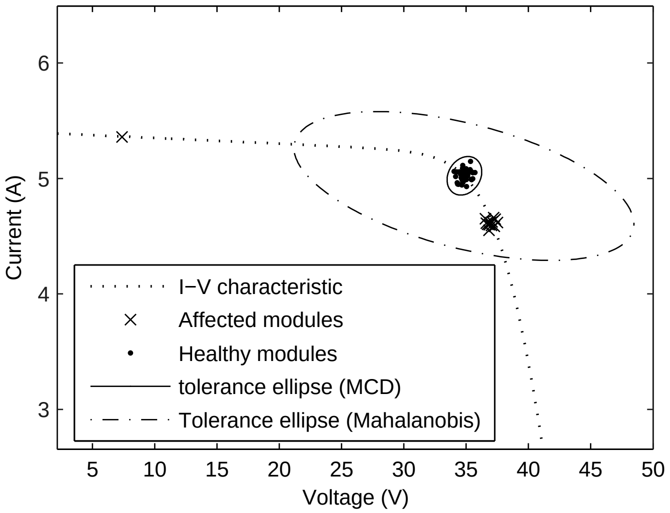


Figure 4.1: Demonstration of masking effect on I-V measurements on an array simulation under ground fault conditions The 99.7 % tolerance ellipse is plotted for each case; the MCD estimator shows little or no masking, while the classical estimator is rendered virtually useless by masking due to multiple clustered outliers.

quickly "masked" by the contaminated data, creating misleading results (Figure 4.1) [30]. One method for dealing with skewed and multimodal data is the k-nearest neighbor approach [31, 32], discussed in Section 4.5. In order to mitigate the masking effect, several robust estimators are used in place of $\boldsymbol{\mu}$ and $\mathbf{C}$; one popular choice is the Minimum Covariance Determinant (MCD) estimator, discussed in Section 4.4.

MCD Estimator

The MCD estimator is a multivariate measure of a dataset's center and variation characteristics that possesses good resistance to data contamination by outliers. First introduced by Rousseeuw [33], the MCD seeks to determine the subset of h points whose covariance matrix has the smallest possible determinant. The parameter h is an integer chosen such that $0.5N < h < N$, where N is the sample size. The MCD estimator has a high breakdown point: it can withstand $N - h$ contaminated observations with only finite deviation from the uncontaminated MCD. The MCD estimator is prohibitively expensive to compute for large samples ($N > 100$) and high-dimensional data. However, the FAST-MCD algorithm [30] finds approximate results with acceptable complexity by iteratively modifying the subset of h points such that its covariance determinant decreases with each iteration, rather than testing every possible subset to find the optimal one. If the outputs of the MCD estimator are used in place of $\mathbf{C}$ and $\boldsymbol{\mu}$ in the Mahalanobis distance, a more robust test statistic for outlier detection is produced, as shown in Figure 4.1.

4.5 k-Nearest Neighbor Methods

The approaches previously mentioned in this section implicitly assume a non-skewed distribution of inputs. Put another way, they assume that the probability density function (PDF) of the distribution from which observations are taken has elliptical level sets which are all centered around the same point. However, in real applications, the underlying PDF of the observations may be skewed; in fact it may be completely unknown. In this case, the k-nearest neighbor method may be used to classify data with only a labeled training set, and without developing any model of the distributions of the classes [34].

The operation of a k-NN classifier is highly intuitive. Training sets $A, B, ...$ containing vectors $\mathbf{a}_i, \mathbf{b}_i,\ i = 1, 2, ...N$ are defined, as is neighborhood size k. Then, an observation $\mathbf{c}$ is presented for classification. Its nearest neighborhood, the set of k training vectors closest to $\mathbf{c}$, is determined using, for instance, one of the distance metrics discussed above. The vectors in the nearest neighborhood are then allowed to vote on the classification of $\mathbf{c}$, and $\mathbf{c}$ is assigned to class A or B based on which class is most represented in its nearest neighborhood [35].

The approach outlined above is effective when high-quality training data for all classes are available. However, in an anomaly detection problem, data is most often available only for the class of normal observations: anomalous events may be extremely rare and may not be clustered into predictable spatial groups. A simple modification of the algorithm presented above transforms the k-nearest neighbor algorithm into an outlier detector. Only one class A is used, representing the non-anomalous observations. A data point $\mathbf{c}$ is once again presented for classification as a normal or anomalous observation and its nearest neighborhood is calculated as before. However, rather than allowing the nearest neighborhood to vote on the class of $\mathbf{c}$, the distance from each of the nearest neighbors to $\mathbf{c}$ is calculated and the sum of these distances is used as test statistic to determine if $\mathbf{c}$ is an outlier. Depending on the application, a fixed number of anomalies may be detected in each pass, or a threshold on distance may be set, above which an observation is determined to be anomalous [31].

4.6 Machine Learning methods

The field of machine learning is concerned with the development and implementation of computational models which determine their behavior based on an input dataset. Learning machines include feedforward neural networks, support vector machines (SVMs), self-organizing maps (SOMs) and genetic algorithms. Machine learning techniques have found applications in pattern recognition, control, and data mining, among other areas. This section describes two popular machine learning algorithms, the feed-forward neural network trained with back-propagation and the support vector machine (SVM), and their application in the area of fault detection. The methods presented below require a training dataset containing a wide sample of both faulty and non-faulty data. Such a dataset is not currently available for solar arrays. However, once monitoring systems become more commonplace and data on fault occurrences becomes available, examples of faulty arrays could feasibly be generated using circuit simulators.

Feed-Forward Neural Networks

The basic feed-forward neural network, also known as a multilayer perceptron (MLP), is arguably the most well-known example of a neural computational model today. In the MLP, simple units known as neurons are connected in a layered topology: each neuron performs a weighted sum of all the neurons in the previous layer and calculates its output based on its activation function $\phi(x)$. In addition to the input and output layers of the network, one or more hidden layers of varying size are used. Arguably, the most common activation function used is the tan-sigmoid function, also known as the hyperbolic tangent. The tan-sigmoid function is one of several activation functions which allows the MLP to act as a universal approximator; that is, with a sufficient number of neurons in the hidden layer, the MLP can map any input-output relationship with arbitrary precision. The popularity of the MLP is due in large part to this fact, since earlier neural network models were restricted in the types of functions they could approximate.

Feedforward neural networks are generally trained by some variation of backpropagation. Many approaches to backpropagation exist; their specifics are beyond the scope of this section, but all follow the same basic approach outlined below. During the ℓ-th iteration, a pair of input and output vectors $\mathbf{x}(\ell)$ and $\mathbf{d}(\ell)$ are presented to the network for training. The network output $\mathbf{y}(\ell)$ and error $\mathbf{e}(\ell) = \mathbf{d}(\ell) - \mathbf{y}(\ell)$ are calculated, and the error energy is used as an objective function to optimize network weights using some variant of gradient descent. A more thorough treatment of backpropagation is given in [36].

For the specific case of a classifier for PV arrays, inputs may reflect the current, voltage, temperature of modules or of the array as a whole, measurements of weather data, measurements of grid behavior, etc. Input data may be taken from only the most recent available samples or past samples may be included as their own inputs. The input vector $\mathbf{x}(k)$ may be pre-processed in order to reduce its dimensionality, for instance by principal component analysis (PCA). Outputs may represent the presence or absence of a fault, the type of fault, or its location, among other things. With proper guidance and a good choice of training sets, the MLP is able to map the input vector with information on present and past array status to an output vector giving information on the performance of the array. In the simplest implementation, a neural network would be trained on a two-class dataset, with classes representing faulty and non-faulty arrays.

Although powerful, there are several notable issues associated with the MLP computational model, including the existence of locally optimal network weights and issues of network underfitting and overfitting. Arguably, the most serious of these is the existence of local minima in the objective function used to optimize network weights. This implies that gradient descent and related optimization methods are likely to settle on a globally suboptimal set of weights $\mathbf{w}$. Fortunately, in practice many of the local minima are very nearly optimal and the user need not exhaustively search for the global optimum. The problem is also mitigated by the variation of gradient descent step size in an annealing-based approach. Step size is initially chosen to be large, ensuring that the backpropagation fully explores the space of possible weights. It is then decreased, allowing the algorithm to “fine tune” the weights to reflect local details of the error surface.

One problem with the MLP, and with most machine learning paradigms, is the difficulty of generating a detailed, comprehensive and noise-free set of input-output pairs for training. In order to demonstrate other issues encountered in MLP training, a simple example from PV is

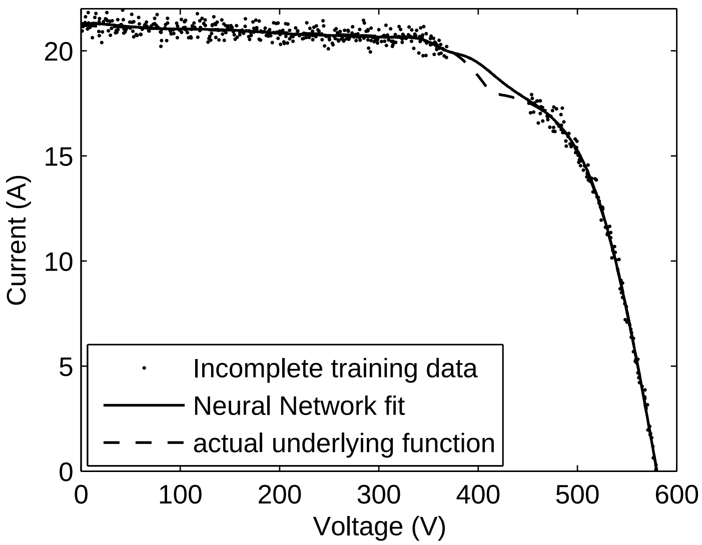


Figure 4.2: Effect of gaps in training data on Neural network training.

considered based on the work in [37]. In [37] a neural network is used to approximate the current of a PV array as a function of temperature, irradiance, and load voltage. By holding irradiance and temperature constant, a single-input single-output neural network may be used to predict the I-V curve of the array. A partially shaded array's I-V curve is approximated in order to better show the effects of overfitting and underfitting.

Ideally, the training set contains a large number of vectors throughout the space of all possible inputs. If gaps or low-density regions exist in the training data, the trained MLP is likely to be a poor approximator of the underlying function for input vectors that are not adequately represented in the training set. This is not a problem unique to machine learning — the MLP often generalizes better than more traditional function approximation techniques. However, the MLP is commonly used with very high-dimensional input data and complex non-linear functional relationships, bringing the issue of generalization to greater prominence. Figure 4.2 shows the effect of a gap in training data on the simple example of an I-V curve of a shaded PV array. When a section of the training data is removed, the network is unable to approximate the function over that region.

A closely related issue is optimal sizing of the MLP. If the MLP's weights are not regularized in any way or it contains too many neurons, overfitting is virtually guaranteed to occur.

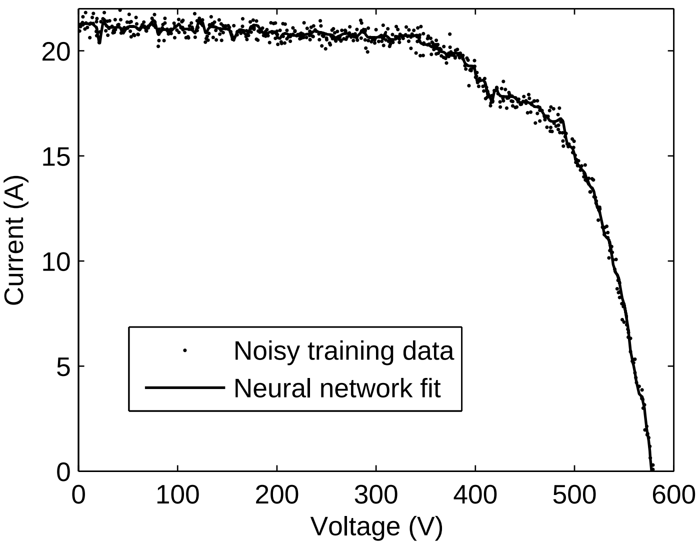


Figure 4.3: Overfitting due to large network size with no regularization.

On the other hand, if too few neurons or overly agressive regularization is used, underfitting is the result. As intuition would suggest, overfitting occurs when the network has too many degrees of freedom and approximates a functional relationship between input noise and output, when in reality none exists. Underfitting, on the other hand, occurs when the network has too few degrees of freedom and is therefore unable to accurately approximate the underlying function. Figure 4.3 shows the effect of overfitting on the approximation of a function in noisy training data.

Although potentially serious, these issues do not prevent the MLP from being a useful tool. Performance is usually not extremely sensitive to network size or regularization parameters and a wide range of network topologies may produce good results.

Support Vector Machines

Another highly successful machine learning model is the support vector machine, or SVM. The SVM is similar to the feed-forward neural network in that its operation is defined by weight matrices which determine the input-output mapping. Unlike the multilayer perceptron, however, the SVM's performance metric is a convex function. As such, the uniquely globally optimal weights for an SVM are relatively easily determined using convex optimization techniques. Like the multi-layer perceptron, the SVM may be used as a classifier or in order to approximate a function. However, the vast majority of successful SVM applications have been in classification problems; this application is treated below.

The SVM classifier operates on the principle of finding the optimal separating hyperplane between two datasets. This optimal hyperplane is defined as the plane which maximizes the margin of separation between the two classes . The points at which this minimum distance is achieved are known as support vectors.

The above-mentioned definition of the optimal separating hyperplane is, of course, meaningful only for two linearly separable classes. Vapnik et. al [38] have performed extensive work in generalizing the SVM to non-separable datasets and non-linear boundaries. The problem of optimally dividing two non-separable classes is approached by assigning a "soft margin" which penalizes the misclassification of data while still maximizing the margin of separation between the correctly classified data. Nonlinear class boundaries are introduced by expanding input vectors to a higher dimensionality using a nonlinear kernel function. By appropriately choosing the SVM kernel, arbitrarily complex classification boundaries may be defined and the SVM becomes a universal approximator in the same sense as the MLP introduced in Section 4.6.

4.7 Final detector: MCD-based detection

From the algorithms considered and described above, only the MCD-based distance metric is both able to be feasibly implemented in a PV array and sufficiently powerful to achieve high detection rates with few false positives. The k-nearest neighbor and machine learning based methods described all require a comprehensive labeled dataset for training of the algorithm. No such dataset is currently available and is unlikely to become available in the near future. creating it would first require the deployment of a comprehensive and accurate fault detection system — an interesting chicken-and-egg problem.

The classical Euclidean and Mahalanobis distance based outlier detection schemes, on the other hand, are not sufficiently powerful to achieve high detection rates with low false positives. Neither are robust and the true behavior of the data is quickly masked by the multiple clustered outliers present in PV array faults. The same is true for the single-class k-nearest neighbor outlier detector described in Section 4.5, since its results depend on distances as well.

Because of these problems with masking, a robust classification algorithm which estimates the center and covariance of the data using the MCD estimator was used.

The algorithm computes a test statistic $T(\mathbf{x}_1, ..., \mathbf{x}_N)$ given by

$$T(\mathbf{x}_1, ..., \mathbf{x}_N) = \max_i \left(\sqrt{(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_x)^T \hat{\mathbf{C}}_x^{-1} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_x)} \right) \tag{4.6}$$

in which $\hat{\boldsymbol{\mu}}_x$ and $\hat{\mathbf{C}}_x$ are estimates of sample mean and covariance matrix computed using the MCD estimator. The MCD (discussed below) identifies the $N/2$ most tightly clustered observations from the set of $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and uses these values to estimate $\boldsymbol{\mu}_x$ and $\mathbf{C}_x$. Finally, $T(\mathbf{x}_1, ..., \mathbf{x}_N)$ is compared against a threshold γ and if $T(\mathbf{x}_1, ..., \mathbf{x}_N) > \gamma$ a fault is determined to be present. The threshold γ is currently chosen to achieve a desired false alarm rate.

The test statistic was selected to measure the maximum distance in I-V space between a module and the center of the modules' distribution. The distance measure used is a quadratic distance closely related to the well-known Mahalanobis distance, but uses different estimators for the mean and covariance matrix.

Selection of estimators for $\hat{\boldsymbol{\mu}}_x$ and $\hat{\mathbf{C}}_x$ is non-trivial: faults in PV arrays often cause the appearance of multiple clusters of module data in I-V space as in Figure 5.5. These clusters prevent the use of conventional estimators of sample mean and covariance, since clusters of faulty modules and strings will mask the characteristics of the non-faulty data. To combat this masking effect, several robust estimators exist which discard large numbers of outlying observations and return estimates only of the cluster to which the majority of observations belong.

The minimum covariance determinant (MCD) estimator was used for this purpose. The MCD estimator [30] first determines the subset $S \subset X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ of h observations (with $N/2 \leq h \leq N$) such that the determinant of the sample covariance matrix $|\hat{\mathbf{C}}_S|$ is minimized. This step effectively discards $N - h$ points as outliers; the subset size is also commonly given as $\alpha = h/N$, the fraction of data which will be kept when performing estimation. $h = N/2$ ($\alpha = 0.5$) was chosen for maximum robustness. Next, the estimated mean $\hat{\boldsymbol{\mu}}_x = \hat{\boldsymbol{\mu}}_S$ and covariance matrix $\hat{\mathbf{C}}_x = \beta \hat{\mathbf{C}}_S$ are calculated, where

$$\beta = \frac{\mathrm{med}_i \left((\mathbf{x}_i - \hat{\boldsymbol{\mu}}_S)^T \hat{\mathbf{C}}_S^{-1} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_S) \right)}{m} \tag{4.7}$$

is a consistency factor to correct for non-anomalous points excluded from S. med is the sample median and m is the population median of a χ_2^2 random variable. This is used because if $\mathbf{x}_i$ are 2-dimensional and normally distributed, $(\mathbf{x}_i - \boldsymbol{\mu}_x)^T \mathbf{C}_x^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_x)$ will be χ_2^2 distributed. Determining the exact subset S is prohibitively expensive to compute for large N. However, the more recent FAST-MCD estimator [30] computes good approximate results with acceptable complexity.

4.8 Summary

In this section, the problem of detecting faults in PV arrays was defined, and several methods for general fault detection systems were described. Except for the MCD-based distance metric, all the methods surveyed were not practical for use in the PV array fault detection problem. Some were not sufficiently robust, while others required large training datasets which are not currently available. The preferred detection algorithm is described in Section 4.7; It uses the robust MCD estimator to determine the centroid and covariance of the cluster of PV module measurements in I-V space (Figure 4.1) and identifies outliers based on their quadratic distance from the centroid.

Chapter 5

DETECTION PERFORMANCE

The performance of the detection algorithms described in Chapter 4 was evaluated using Monte Carlo analysis. This was necessary because the relevant statistics of the algorithms are not easily calculated. In particular, the MCD-based method is not easily mathematically analyzed. Performance of detectors was evaluated using the receiver operating characteristic (ROC) as a performance metric. This section discusses the Monte Carlo simulation procedure, the relevance of the ROC as a summary of detector performance, and the behavior of the different detectors under ground and arc fault conditions.

5.1 Monte Carlo Simulations

Monte Carlo simulations are a powerful method of estimating the behavior of random processes when an analytical solution is not feasible. Several variations of the technique exist, but all involve generating a large number of realizations of the process to be studied and estimating its parameters from these realizations. In this case, 10^5 realizations of both the no-fault case and the faulty case were generated for each fault type examined. Measurement noise was modeled as additive white Gaussian noise (AWGN) with variance $0.13\ \mathrm{V}^2$ and $0.0025\ \mathrm{A}^2$ for voltage and current, respectively; these values correspond to a standard deviation of 1% of the true (noiseless) value. Errors in voltage and current measurement were assumed to be independent of one another; in measurements from Paceco Corp's prototype monitoring system this appears to be the case.

5.2 Receiver Operating Characteristic

The receiver operating characteristic (ROC) is used to summarize the performance of a detector. It plots the probability of false alarm on the x-axis and the probability of detection on the y-axis. In this way, the ROC reflects the trade-off between sensitivity and specificity of a detector and gives performance over the full range of detection thresholds.

Since no accurate data are available on the relative frequency and severity of different faults, it is not feasible to determine the ROC of the detector as a whole. Instead, simulations were carried out individually for different examples of faults. Simulations for ground and arc faults are discussed in the following sections.

5.3 Test Statistics

6 different test statistics were evaluated in the Monte Carlo simulations and their performance was compared. The detectors studied were the MCD-based detector with two different robustness parameters α, Mahalanobis distance, likelihood of observation under a Gaussian distribution, and Z-score of both module voltage and current. The MCD and Mahalanobis distances are discussed in detail in Chapter 4. The test statistic for likelihood of observation under a Gaussian is given by

$$T(\mathbf{x}) = \frac{1}{\prod_i p(\mathbf{x}_i|\hat{\boldsymbol{\mu}}, \hat{\mathbf{C}})} \tag{5.1}$$

where $\hat{\boldsymbol{\mu}}$ and $\hat{\mathbf{C}}$ are the sample mean and covariance, respectively, of the data, and $p(\mathbf{x}_i|\hat{\boldsymbol{\mu}}, \hat{\mathbf{C}})$ is the likelihood of vector $\mathbf{x}_i$ given the sample mean and covariance matrix of the data.

5.4 Fault modeling

The case examined for simulation was that of a 13-series 4-parallel array, comprising 52 individual Sharp NT-175U1 modules. This configuration was chosen because it matched that of a nearby PV array on which a prototype monitoring system has been installed. Simulation of all faults was performed using De Soto's 5-parameter model, described in Section 2.2 [6]. Simulations were performed at standard test conditions (STC) of $1000\mathrm{W/m^2}$ direct normal irradiance, airmass 1.5, and $25^\circ\mathrm{C}$ cell temperature. Circuit component parameters for the Sharp-NT-175U1 module at STC were calculated using the Engineering Equation Solver (EES) software as described in [6]. Table 5.1 gives the circuit parameters generated by EES. The LTSpice netlist implementation of the NT-175U1 module is given in Appendix A.

Two types of faults were examined: ground faults and DC series arc faults. Ground faults were simulated by connecting a resistor between circuit ground and the fault location (Figure 5.4). Arc faults were simulated using the steady-state Cassie arc model (Section 3.5), which assumes a constant arc voltage drop (Figure 3.8).

Table 5.1: Component parameters for 5-parameter model of Sharp NT-175U1 module at STC.

name	value	description
I_s	1.685×10^{-10} A	diode reverse saturation current
I_ph	5.419 A	light-generated current
n	71.1025	diode ideality factor
R_s	0.7294 Ω	series resistance
R_sh	202.9 Ω	shunt (parallel) resistance

5.5 Detector Performance

DC Series Arc Fault

Series arc fault simulation was performed using the Cassie arc model for steady-state arc voltage of 5 V, corresponding to a relatively small arc, but one which is fully capable of destroying a PV module. Fig. 5.2 shows a scatter plot of the faulty array's modules in I-V space. It may be seen from figure 5.2 that the arc fault does not show the multiple clustered outliers common to other faults in PV arrays (see Section 4.1) Fig. 5.1 shows the ROC of the MCD-based detector operating on this fault. It can be seen that the detector achieves a detection rate of 98% at a false alarm rate of only 0.01%.

Figure 5.3 compares the ROC of the MCD-based detector to that of the other detectors described in Section 5.3. Since the fault does not result in clustered outliers in I-V space, there is in this case no benefit to the use of robust statistics; in fact, the MCD-based algorithm underperforms the non-robust Mahalanobis distance. This is because the MCD algorithm discards $J/(1-\alpha)$ of the data points before performing estimation, leading to higher variance in $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$, the estimates of mean and covariance. When α is increased from 0.5 to 0.75, fewer I-V pairs are discarded and the algorithm performance is improved.

Ground Fault

A 100 Ω ground fault across 4 modules was simulated. Figure 5.4 shows the LTSpice schematic of this fault scenario. Figure 5.5 shows the location of every module in the array in I-V space. Unlike in the previously examined case of an arc fault, multiple clustered outliers are present. As expected, their presence in the data causes the robust statistics-based MCD algorithm to dramatically outperform the non-robust Mahalanobis distance. This is shown in Figure 5.6, which plots the ROC curves of several detectors operating on the data shown in Figure 5.5.

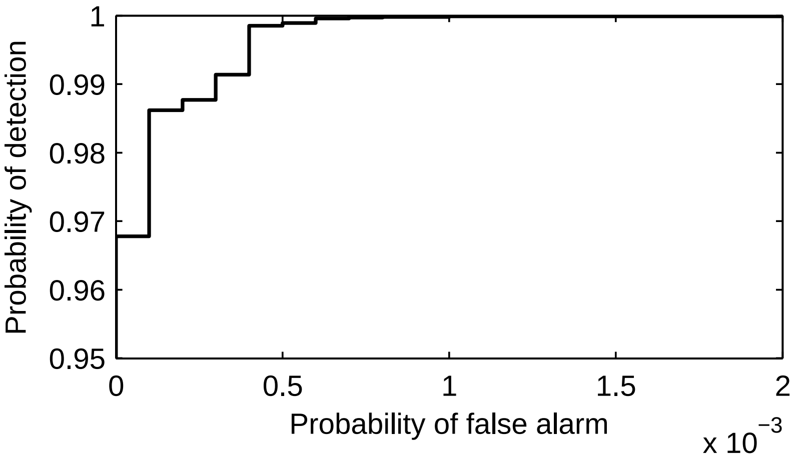


Figure 5.1: ROC of MCD-based detector under 5V Cassie model series arc fault.

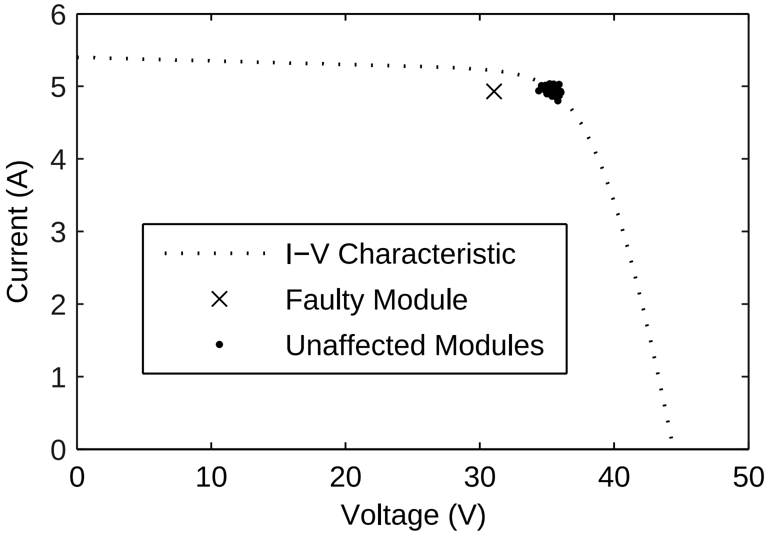


Figure 5.2: Location of PV modules in I-V space for 5V Cassie model series arc fault.

5.6 Summary

In this chapter, the results of Monte Carlo simulations of ground and DC series arc faults was examined. The receiver operating characteristic (ROC) for several detection algorithms was derived and their detection performance was examined. Only one method, the MCD-based detector, showed acceptable performance on all fault types, achieving a 98% detection rate with 0.01% false alarm rate.

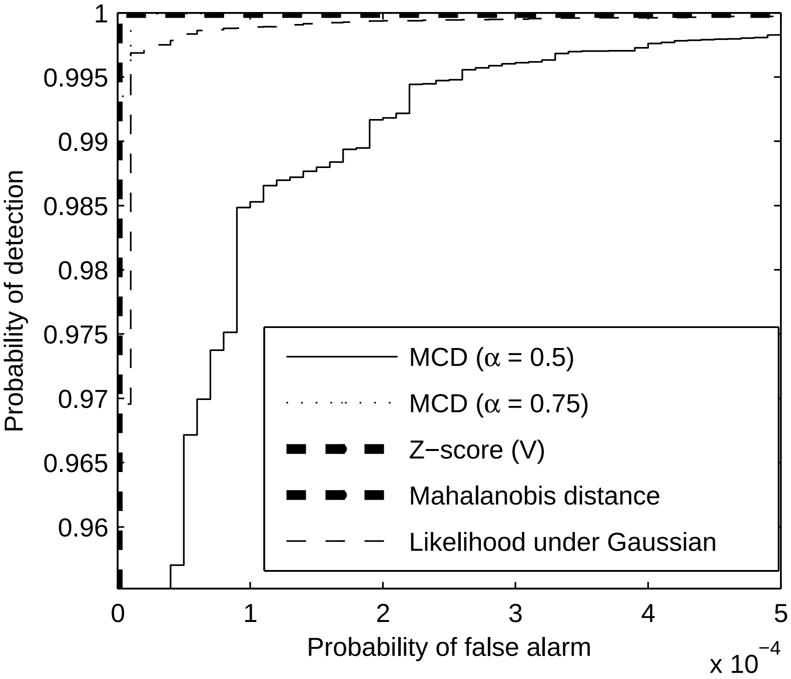


Figure 5.3: Comparison of performance of MCD-based detector with other test statistics on series arc fault.

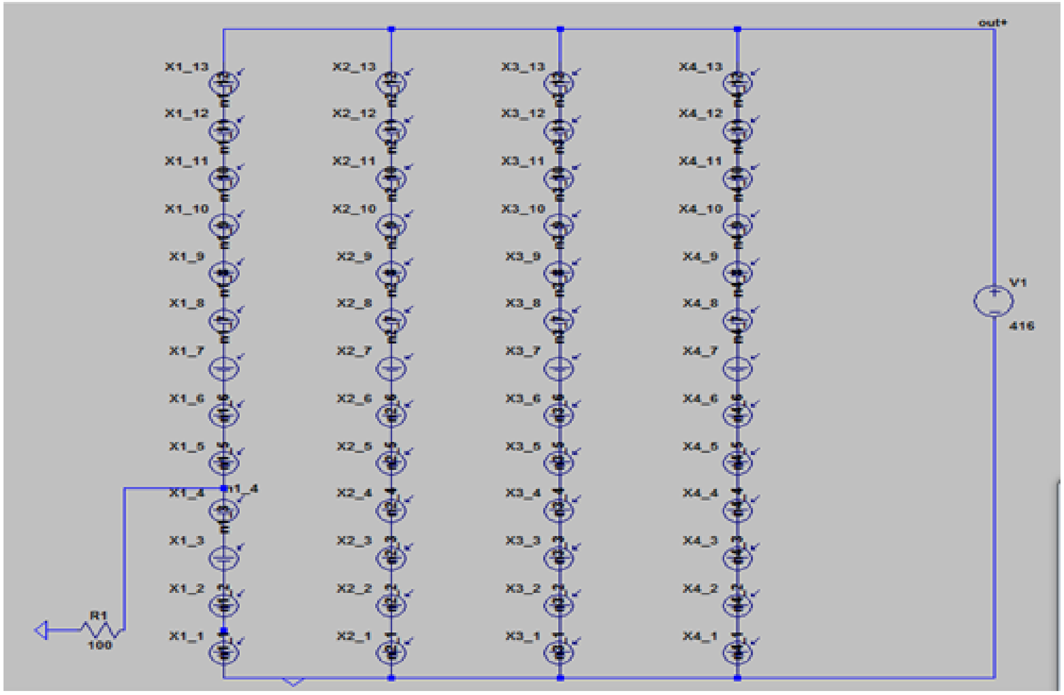


Figure 5.4: Schematic of ground fault simulation.

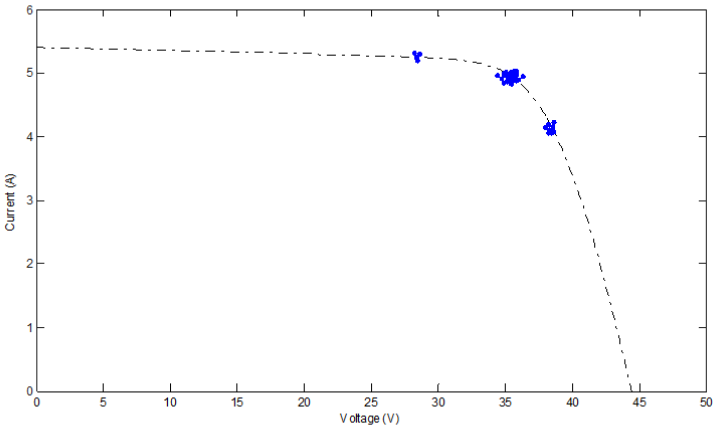


Figure 5.5: Location of PV modules in I-V space for $100\ \Omega$ ground fault across 4 modules.

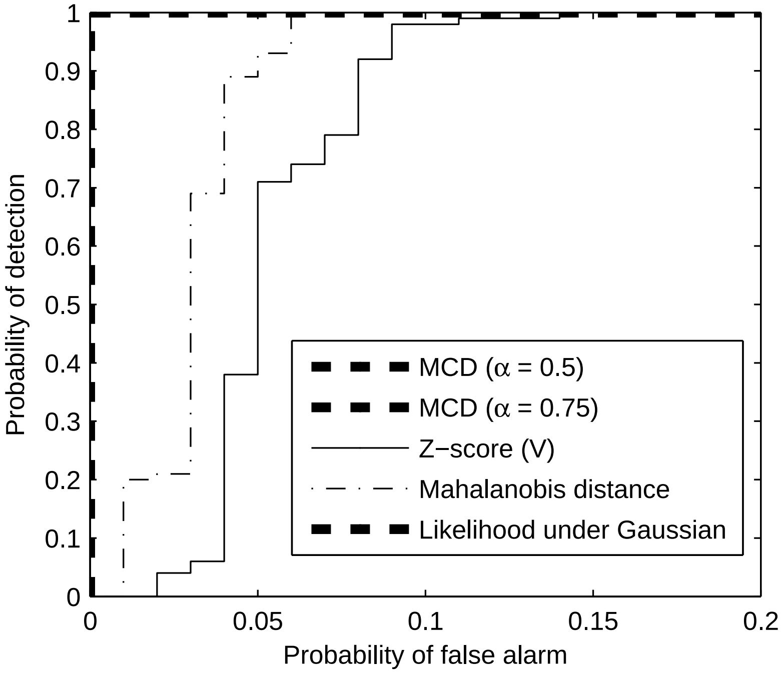


Figure 5.6: Comparison of performance of MCD-based detector with other test statistics on ground fault.

Chapter 6

CONCLUSION

A "smart PV array" may be equipped with monitoring and control capabilities in order to achieve an increased operating efficiency and fault tolerance. Signal processing and pattern recognition techniques are used to monitor photovoltaic arrays and detect and respond to faults with minimal human involvement. In this document, the problem of automated PV array fault detection is presented and several potential detection algorithms are described. The algorithms are then tested on simulated DC series arc and ground faults using Monte Carlo analysis techniques. Only one algorithm, the MCD-based distance threshold, was found to exhibit good behavior on both types of faults.

First, Chapter 2 described the high-level electrical behavior of PV modules and arrays, and presented two commonly used models for simulation of PV modules. One of these, the 5-parameter model, was used extensively for modeling of faults in later chapters.

Chapter 3 described several common faults occurring in PV arrays. Shading, mismatch due to variation between modules, ground faults, and DC arc faults are all described. In general, most faults look similar, with a single-point fault affecting the entire series string in which the fault occurs, and potentially the entire array. The effect of this non-independence is twofold: multiple clustered outliers cause non-robust estimation techniques to fail, and increase the power lost due to a fault.

Chapter 4 described several approaches to fault detection and their potential for implementation in a PV array setting. Classical likelihood ratio based methods and statistical outlier detectors such as the Mahalanobis distance and k-nearest neighbor methods were presented. Machine-learning based methods were also described. Likelihood-based methods, some forms of the k-nearest neighbor algorithm, and machine learning methods all require a more detailed and accurate statistical model of PV array faults than is currently available. Mahalanobis distance, on the other hand, suffers from "masking" due to the non-independent nature of PV modules under fault conditions, as does the unsupervised version of the k-nearest neighbor algorithm. Only the MCD-based method, described in Section 4.4, overcomes all these difficulties.

Chapter 5 presented the results of Monte Carlo analysis for PV array ground and arc faults generated using a SPICE circuit simulator. Results were given in the form of ROC curves, which summarize the trade-off between sensitivity and specificity in a detector. Only the MCD-based algorithm showed good performance on both fault types, although several other detectors met or exceeded its performance on one type or the other.

This work is believed to be the first attempt to use high-resolution measurements from a PV array monitoring system for fault detection. Other methods in the literature perform detection only at the array level [39], or at a reduced resolution [40]. It is hoped that the MCD-based detection algorithm presented herein will be deployed as part of a comprehensive PV array monitoring system, including data visualization and dynamic reconfiguration (switching) capabilities. Once deployed, the data from this array may be used to further improve fault detection capabilities, increasing PV array energy production and lowering the cost of energy for array operators and their customers.

REFERENCES

[1] P. Wong, "Solar photovoltaic cell/module manufacturing activities 2009," US Energy Information Administration, Tech. Rep., Jan. 2011.

[2] J. Conti and P. Holtberg, "Levelized cost of new generation resources in the annual energy outlook 2011," US Energy Information Administration, Tech. Rep., Dec. 2010.

[3] D. King, J. Kratochvil, and W. Boyson, "Photovoltaic array performance model," Sandia National Laboratory, Tech. Rep., 2004.

[4] A. Maish, C. Atcitty, S. Hester, D. Greenberg, D. Osborn, D. Collier, and M. Brine, "Photovoltaic system reliability," in *Photovoltaic Specialists Conference, 1997., Conference Record of the Twenty-Sixth IEEE*, Sep. 1997, pp. 1049 –1054.

[5] K. Otani, T. Takashima, and K. Kurokawa, "Performance and reliability of 1 MW photovoltaic power facilities in AIST," in *Photovoltaic Energy Conversion, Conference Record of the 2006 IEEE 4th World Conference on*, vol. 2, May 2006, pp. 2046 –2049.

[6] W. De Soto, S. Klein, and W. Beckman, "Improvement and validation of a model for photovoltaic array performance," *Solar Energy*, vol. 80, no. 1, pp. 78 – 88, Aug. 2006.

[7] J. Yang, H. Bae, J. Lee, and B. Cho, "A simplified series-parallel structure for the regulated peak power tracking (RPPT) system," in *Applied Power Electronics Conference and Exposition, 2008. APEC 2008. Twenty-Third Annual IEEE*, 2008, pp. 160 –166.

[8] C.-L. Shen, Y.-E. Wu, and F.-S. Liu, "A double-linear approximation algorithm to achieve maximum-power-point tracking for PV arrays," in *Power Electronics and Drive Systems, 2009. PEDS 2009. International Conference on*, 2009, pp. 758 –763.

[9] T. Esram and P. Chapman, "Comparison of photovoltaic array maximum power point tracking techniques," *Energy Conversion, IEEE Transactions on*, vol. 22, no. 2, pp. 439 –449, Jun. 2007.

[10] J. Blanes, A. Garrigos, J. Carrasco, E. Avila, and E. Maset, "Maximum power point estimator for photovoltaic solar arrays," in *Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean*, 2006, pp. 889 –892.

[11] H. Patel and V. Agarwal, "MATLAB-Based modeling to study the effects of partial shading on pv array characteristics," *Energy Conversion, IEEE Transactions on*, vol. 23, no. 1, pp. 302 –310, Mar. 2008.

[12] D. Nguyen and B. Lehman, "Modeling and simulation of solar PV arrays under changing illumination conditions," in *Computers in Power Electronics, 2006. COMPEL '06. IEEE Workshops on*, July 2006, pp. 295 –299.

[13] V. Quaschning and R. Hanitsch, “Numerical simulation of current-voltage characteristics of photovoltaic systems with shaded solar cells,” *Solar Energy*, vol. 56, no. 6, 1996.

[14] F. Spertino and J. Akilimali, “Are manufacturing I-V mismatch and reverse currents key factors in large photovoltaic arrays?” *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 11, pp. 4520 –4531, Nov. 2009.

[15] N. D. Kaushika and N. K. Gautam, “Energy yield simulations of interconnected solar PV arrays,” *Power Engineering Review, IEEE*, vol. 22, no. 8, pp. 62 –62, Aug. 2002.

[16] R. Hammond, D. Srinivasan, A. Harris, K. Whitfield, and J. Wohlgemuth, “Effects of soiling on PV module and radiometer performance,” in *Photovoltaic Specialists Conference, 1997., Conference Record of the Twenty-Sixth IEEE*, 29 1997, pp. 1121 –1124.

[17] R. C. Quick, “Ground fault circuit interrupter—design and operating characteristics,” *Industry Applications, IEEE Transactions on*, vol. IA-11, no. 1, pp. 50 –55, 1975.

[18] *NFPA 70: National Electrical Code*, NFPA Std., 2008.

[19] G. Gregory and G. Scott, “The arc-fault circuit interrupter: an emerging product,” *Industry Applications, IEEE Transactions on*, vol. 34, no. 5, pp. 928 –933, 1998.

[20] G. Gregory, K. Wong, and R. Dvorak, “More about arc-fault circuit interrupters,” *Industry Applications, IEEE Transactions on*, vol. 40, no. 4, pp. 1006 – 1011, 2004.

[21] M. Naidu, T. Schoepf, and S. Gopalakrishnan, “Arc fault detection scheme for 42-V automotive dc networks using current shunt,” *Power Electronics, IEEE Transactions on*, vol. 21, no. 3, pp. 633 –639, May 2006.

[22] H. Haeberlin and M. Kaempfer, “Measurement of damages at bypass diodes by induced voltages and currents in PV modules caused by nearby lightning currents with standard waveform,” in *23rd European Photovoltaic Solar Energy Conference*, 2008.

[23] K.-J. Tseng, Y. Wang, and D. Vilathgamuwa, “An experimentally verified hybrid cassie-mayr electric arc model for power electronics simulations,” *Power Electronics, IEEE Transactions on*, vol. 12, no. 3, pp. 429 –436, may 1997.

[24] A. M. Cassie, “Arc rupture and circuit serverity: A new theory,” CIGRE, Tech. Rep., 1939.

[25] O. Mayr, “Beiträge zur theorie des statischen und des dynamischen lichtbogens,” *Electrical Engineering (Archiv fur Elektrotechnik)*, vol. 37, pp. 588–608, 1943, 10.1007/BF02084317. [Online]. Available: http://dx.doi.org/10.1007/BF02084317

[26] G. Idarraga Ospina, D. Cubillos, and L. Ibanez, “Analysis of arcing fault models,” in *Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES*, aug. 2008, pp. 1 –5.

[27] N. Bosco, “Reliability concerns associated with PV technologies,” National Renewable Energy Laboratory, Tech. Rep., Apr. 2010.

[28] F. Chan and H. Calleja, “Reliability: A new approach in design of inverters for PV systems,” in *International Power Electronics Congress, 10th IEEE*, oct. 2006, pp. 1 –6.

[29] S. Kay, *Fundamentals of Statistical Signal Processing: Detection theory*, ser. Prentice Hall signal processing series. Prentice-Hall PTR, 1998. [Online]. Available: http://books.google.com/books?id=vA9LAQAAIAAJ

[30] P. J. Rousseeuw and K. v. Driessen, “A fast algorithm for the minimum covariance determinant estimator,” *Technometrics*, vol. 41, no. 3, pp. 212–223, Aug. 1999.

[31] V. Hautamaki, I. Karkkainen, and P. Franti, “Outlier detection using k-nearest neighbour graph,” in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, Aug. 2004, pp. 430 – 433.

[32] F. Angiulli and C. Pizzuti, “Fast outlier detection in high dimensional spaces,” in *Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, T. Elomaa, H. Mannila, and H. Toivonen, Eds. Springer Berlin / Heidelberg, 2002, vol. 2431, pp. 43–78.

[33] P. Rousseeuw, “Multivariate estimation with high breakdown point,” *Mathematical statistics and applications*, vol. 8, pp. 283–297, 1985.

[34] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” *Journal of Biomedical Informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[35] M. Hellman, “The nearest neighbor classification rule with a reject option,” *Systems Science and Cybernetics, IEEE Transactions on*, vol. 6, no. 3, pp. 179 –185, Jul. 1970.

[36] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Prentice Hall, 2009.

[37] A. Al-Amoudi and L. Zhang, “Application of radial basis function networks for solar-array modelling and maximum power-point prediction,” *Generation, Transmission and Distribution, IEEE Proceedings*, vol. 147, no. 5, pp. 310 –316, Sep. 2000.

[38] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, pp. 273–297, 1995, 10.1007/BF00994018.

[39] S. Vergura, G. Acciani, V. Amoruso, and G. Patrono, “Inferential statistics for monitoring and fault forecasting of PV plants,” in *ISIE 2008. Industrial Electronics, 2008, IEEE International Symposium on*, Jun. 2008, pp. 2414 –2419.

[40] H. Zhiqiang and G. Li, “Research and implementation of microcomputer online fault detection of solar array,” in *Computer Science Education, 2009. ICCSE ’09. 4th International Conference on*, Jul. 2009, pp. 1052 –1055.

Appendix A

ADDITIONAL RESULTS

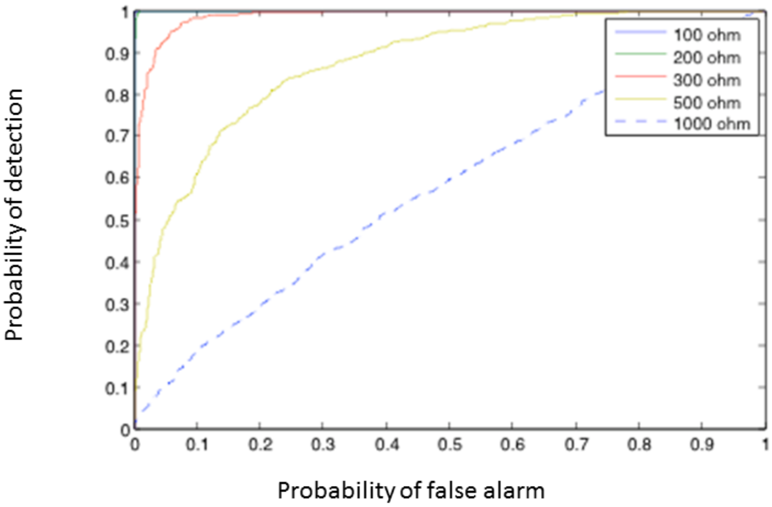


Figure A.1: Comparison of MCD-based detector performance under varying ground fault resistance.

This section contains additional simulation results which were not mentioned in the main body of the paper.

Figure A.1 shows a comparison of MCD-based detector performance as ground fault resistance varies. This result provided an early indication that the MCD-based detection algorithm could show high performance on plausible fault types.

Figure A.2 shows a comparison of MCD-based detector performance as arc fault resistance increases. In early simulations, arcs were modeled as series resistances rather than as Cassie model constant-voltage arcs. As with Figure A.1 above, this result provided an early indication that the MCD-based detection algorithm could show high performance on plausible fault types.

In Figure A.3, a measurement snapshot from Paceco Corp's prototype monitoring system is shown. Each point represents a single PV module. The blue line is the Sandia Model's predicted I-V characteristic given temperature and irradiance data from a weather station. Data such as this guided the early development of the algorithms. First, the predicted I-V characteristic is highly unreliable due to low-quality irradiance measurements; this led to a focus on methods which did not depend on accurate environmental data. Second, the plot shows two probable minor arc faults due to a design flaw in an early version of the monitoring hardware. The presence of multiple data clusters in I-V space caused by these faults led to a focus on robust statistics for detection.

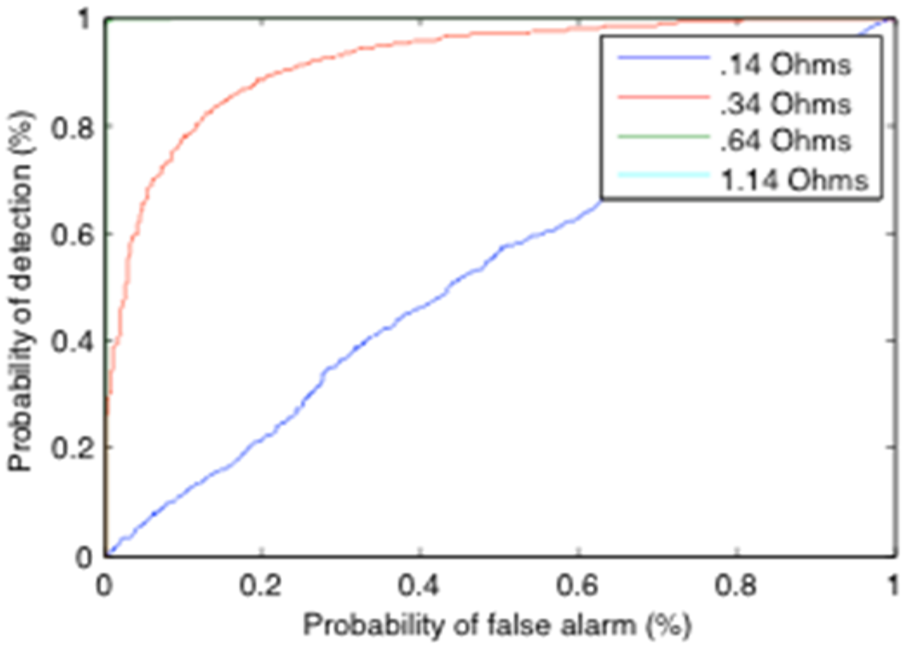


Figure A.2: Comparison of MCD-based detector performance under varying arc resistance.

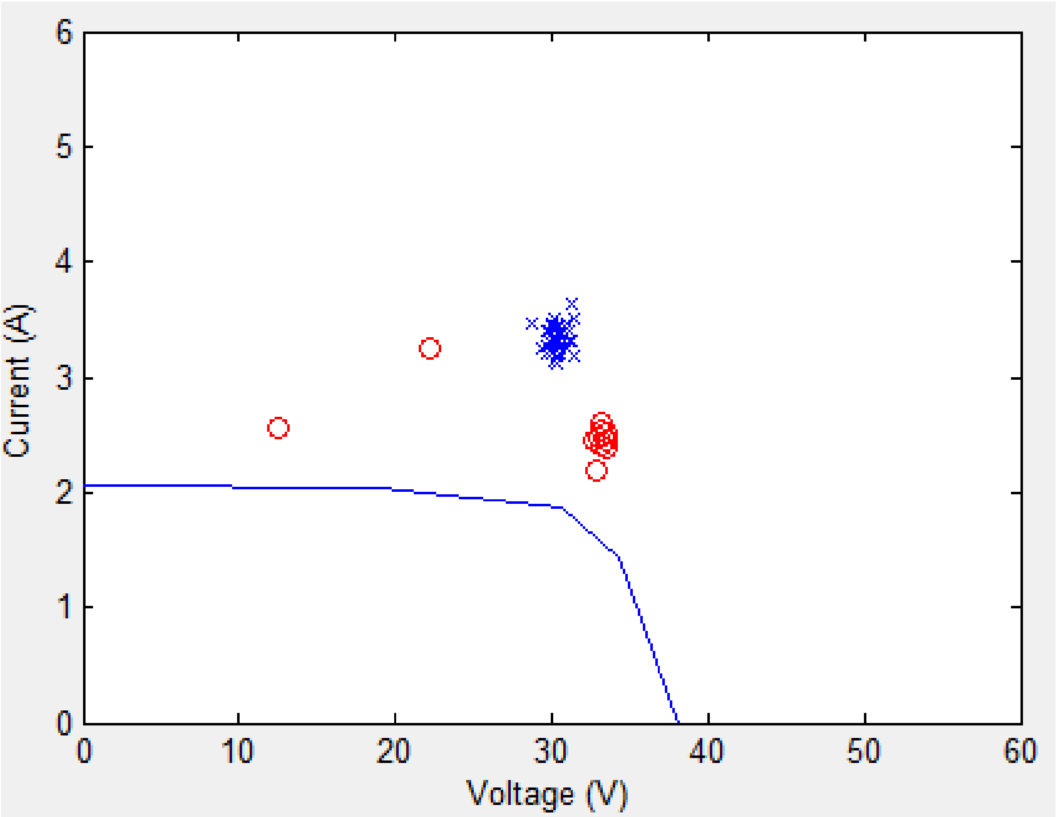


Figure A.3: Measurement Snapshot from Paceco Corp's prototype monitoring system.

Appendix B

MATLAB CODE AND SPICE NETLISTS

B.1 MATLAB implementation of Sandia PV performance model

NT_175U.m: Generate module parameters for Sharp NT-15U PV module and generate default environmental inputs

```
%set module parameters for Sharp NT-175U panel, Sandia model
function [modelParams arrayParams envParams] = nt_175u()
clear modelParams %envParams

modelParams.name = ’Sharp NT-175U1’;
modelParams.vintage = 2007;
modelParams.area = 1.3;
modelParams.material = ’c-Si’;
modelParams.series_cells = 72;
modelParams.parallel_strings = 1;
modelParams.Isco = 5.40;
modelParams.Voco = 44.4;
modelParams.Impo = 4.95;
modelParams.Vmpo = 35.4;
modelParams.aIsc = .000351;
modelParams.aImp = -.000336;
modelParams.C0 = 1.003;
modelParams.C1 = -.003;
modelParams.BVoco = -.151;
modelParams.mBVoc = 0;
modelParams.BVmpo = -.158;
modelParams.mBVmp = 0;
modelParams.n = 1.323;
modelParams.C2 = .001;
modelParams.C3 = -8.711;
modelParams.A = [.931498305 .059748475 -.010672586 .000798468
   -2.23567E-5];% actual NT-175u parameters
%modelParams.A = [0.921940714 0.070891738 -0.01427241
   0.001170898 -3.37053E-5]; %parameters from other C-Si panel
modelParams.B = [1 -.002438 .0003103 -1.246E-5 2.112E-7 -1.359E
   -9];
modelParams.dTc = 3;
modelParams.fd = 1;
modelParams.a = -3.56; %actually alpha in King’s paper
modelParams.b = -.075; %actually beta
modelParams.C4 = .992;
modelParams.C5 = .008;
modelParams.Ixo = 5.32;
modelParams.Ixxo = 3.51;
modelParams.C6 = 1.128;
modelParams.C7 = -.128;
modelParams.e0 = 1000;
modelParams.To = 25;
```

```
%set parameters for De Soto model


% %set environmental conditions:
envParams.airmass = 1;
envParams.aoi = 0; %angle of incidence
envParams.T_ambient = 25;
envParams.T_cell = 30;
envParams.irradiance = 1000;
envParams.P_diffuse = 0;

%set parameters of the array:

%Array tracking type. 0 for fixed angle, 1 for single-axis, 2
   for
%dual-axis.
arrayParams.trackType = 0;

%direction of tilt (maximum decrease) in degrees.  0 is North,
   90 is east.
%ignored for tracking type 2 (dual-axis).
arrayParams.tiltDirection = 180;

%FIXED tilt angle in degrees.  0 is lying flat on the ground,
   90 is
%standing on edge.  ignored for tracking type 2 (dual-axis).
   for type 1
%(single-axis), this is the angle when panels are stowed
   horizontally.
arrayParams.tiltAngle = 0;

arrayParams.size = 72;  %number of modules in array
arrayParams.moduleType = ’Sharp NT-175U1’; %type of module used

arrayParams.maxVoltage = 600; %maximum voltage before inverter
   shutdown
arrayParams.minVoltage = 300; %minimum voltage before inverter
   shutdown
arrayParams.maxCurrent = 55;  %conductor ampacity at SRC (25 C)
arrayParams.tempCoeffAmpacity = -.85/100; %temperature
   coefficient of ampacity

arrayParams.nSer = 12;
arrayParams.nPar = 6;
arrayParams.resistance = 1;
arrayParams.utcOffset = -7;
```

get_IV_curve.m: Generate I-V points given structures containing environmental conditions and module parameters

```
%calculates a solar panel IV curve based on the Sandia solar
   module
%performance model.

function [V I] = get_IV_curve(env, model)
k = 1.38066E-23; %Boltzmann’s constant, J/K
q = 1.602E-19; % charge of electron, coulombs

%calculate airmass dependence of model:
f_airmass = max(0,polyval(fliplr(model.A),env.airmass));

%calculate angle of incidence dependence:
f_aoi = polyval(fliplr(model.B),env.aoi);

%calculate temperature difference from SRC:
delta_T = env.T_cell - model.To;

%calculate short-circuit current Isc
Isc = model.Isco * f_airmass * (f_aoi * env.irradiance + model.
   fd * env.P_diffuse)/model.e0 ...
    * (1 + model.aIsc*delta_T);

%calculate effective irradiance Ee
Ee = Isc / (model.Isco * (1 + model.aIsc*delta_T));

%calculate maximum-power current Imp
Imp = model.Impo * (model.C0*Ee + model.C1 * Ee^2)*(1 + model.
   aImp*delta_T);

%calculate "thermal voltage" Vt
Vt = model.n * k * (env.T_cell + 273.15)/q; %T_cell converted
   from C to K

%calculate open-circuit voltage Voc
BVoc = model.BVoco + model.mBVoc*(1- Ee); %temperature
   coefficient as function of effective irradiance
Voc = max(0,model.Voco + model.series_cells*Vt*log(Ee) + BVoc *
    delta_T);

%calculate maximum power voltage Vmp
BVmp = model.BVmpo + model.mBVmp*(1- Ee); %temperature
   coefficient as function of effective irradiance
Vmp = max(0,model.Vmpo + model.C2*model.series_cells*Vt*log(Ee)
    + model.C3*model.series_cells*(Vt * log(Ee))^2 + BVmp *
   delta_T);

%calculate additional currents Ix and Ixx
```

```
Ix = model.Ixo * (model.C4 * Ee + model.C5 * Ee^2)*(1 + model.
   aIsc*delta_T);
Ixx = model.Ixxo * (model.C6 * Ee + model.C7*Ee^2)*(1+ model.
   aImp*delta_T);

%combine points into vectors
V = [0 Voc/2 Vmp (Voc + Vmp)/2 Voc];
I = [Isc Ix Imp Ixx 0];
if V(2) > V(3)
    V = [0 Vmp (Voc + Vmp)/2 Voc];
    I = [Isc Imp Ixx 0];
end
if Voc == 0
    V = 0;
    I = 0;
end
```

B.2 5-parameter model: LTSpice netlists

```
* C:\Users\hcbraun\Documents\research\LTspice\nt_175u.asc
D_cell N001 MINUS nt_175u_model
BÂğI_phot MINUS N001 I={I_ph}
R_par N001 MINUS 202.9
R_ser PLUS N001 .7294
D1 MINUS PLUS MBR20100CT
.model D D
.lib C:\Program Files (x86)\LTC\LTspiceIV\lib\cmp\standard.dio
.model nt_175u_model D(Is ={Is} N = {N})
.param Is_ref 1.685E-10
.param tol_Is_ref 0.0
.param N_ref 71.1025
.param tol_N_ref 0.0
.param I_ph_ref 5.419
.param tol_I_ph_ref 0.0
.param T_ref 298.15
.param T 298.15
.param S_ref 1000
.param M_ref 1.5
.param S 1000
.param M 1.5
.param Is Is_ref * (1+gauss(tol_Is_ref)) * (T/T_ref)**3 * exp
   (1/kb * (E_g_ref/T_ref - E_g/T))
.param N {N_ref * (1 + gauss(tol_N_ref)) * T/T_ref}
.param E_g_ref 1.121
.param E_g E_g_ref * (1 - .0002677 * (T - T_ref))
.param kb 8.617343E-10
.param I_ph S/S_ref * M/M_ref * (I_ph_ref * (1 + gauss(
   tol_I_ph_ref)) + a_isc*(T - T_ref))
.param a_isc .0029
```

```
.param test  (I_ph_ref * (1 + gauss(tol_I_ph_ref)) + a_isc*(T -
    T_ref))
.backanno
.end
```

B.3 MATLAB implementation of MCD algorithm

```
function [mcdout] = mcd_fast(xnan, varargin)
%simplified re-implementation of P. Rousseeuw’s FAST-MCD
   algorithm.

% Inputs:
% xnan            (required) Input data matrix.  Rows are
   observations,
%                 columns are variables. NaNs and infs are
   permitted but will be ignored.
% alpha           robustness parameter.  Must be a scalar
   between 0.5 and 1.
% h               number of observations to keep when doing
   estimation.
%                 h = alpha * number of observations. if both h
    and alpha are
%                 given, whichever one is specified last will
   be used.
% ntrial          number of initial guesses to start from when
   doing optimization. default = 500.

% Outputs:
% mcdout              structure with estimation results
%     .center         Estimated data centroid.
%     .cov            Estimated data covariance matrix.
%     .h              Subset size parameter h from above.
%     .alpha          Robustness parameter from above.
%     .rd             Robust distances of the dataset.



%nDims: dimensionality of data (number of columns in x);
%nObs: number of observations (number of rows in x);
%nStarts:  number of random starting points (default 500)
%nIter:  number of iterations to decrease the covariance matrix


%define constants:
nDims = size(xnan,2);
nObs = size(xnan,1);
h = round(0.75 * nObs); %defalt value; can be overriden with ’h
   ’ argument
nStarts = 500; %default value; can be overriden with the ’
   ntrial’ argument
```

```
nIter = 2; %number of c-steps to do when selecting the 10 best
   candidates for additional steps
nSetsToKeep = 10;
reweightTol = 0.975; %what significance level to cut off at
   when re-weighting

%parse the variable arguments:
for i = 1:2:nargin - 1
    argName = varargin{i};
    switch argName
        case ’h’
            h = varargin{i + 1};
        case ’alpha’
            alpha = max(varargin{i + 1}, 0.5);
            h = ceil(nObs * alpha);
        case ’ntrial’
            if isinteger(varargin{i+1}) && varargin{i + 1} > 0
                nStarts = varargin{i + 1};
            else
                warning(’non-permissible value for argument "
                   ntrial".  Using default value of 500.’);
            end
        otherwise
            warning([’unrecognized argument type: "’ varargin{i
               } ’".’]);
    end
end

%check for permissibility of h:
if mod(h,1) ~= 0 %if h is not an integer
    h = round(h);
    warning(’non-integer set size "h", rounding to nearest
       integer.’);
end
if 2 * h<= nObs
    h = floor(nObs / 2) + 1;
    warning([’argument "h" is too low.  Using minimum value of
       ’ int2str(h) ’ instead.’]);
end
if h > nObs
    h = nObs;
    warning([’argument "h" is too high.  Using maximum value of
        ’ int2str(h) ’ instead.’]);
end

%check for NaNs and infs in data and ignore them if needed.
nanMaskBig = ~isfinite(xnan); %find the NaNs and infs
```

```
nanMask = logical(sum(nanMaskBig, 2)); %find all the rows with
   a NaN or inf
x = xnan(~nanMask,:); %discard rows with NaNs or Infs
%nNans = sum(nanMask(:));
N = nObs - sum(nanMask(:)); %number of GOOD (not NaN or inf)
   observations

if N < h || N < nDims;
    error('mcd_fast:dataTooSmall','Insufficient observations
       for estimation.');
end

%pick nStarts combinations at random, and
%for each trial combination, iterate for 2 steps:

det_C = inf;
det_C_best10 = inf * ones(nSetsToKeep,1);
weights_best10 = false(N,nSetsToKeep);
dists = zeros(1,N);
dists_final = zeros(1,N);
for i = 1:nStarts

    %find the starting point
    indices = randperm(N);
    weights = false(N,1);
    weights(indices(1:h)) = true;
    %iteratively decrease for two steps
    for j = 1:nIter %only repeat nIter times even if there is
       no convergence

        %calculate its mean and covariance matrix for the
           current weights
        Chat = cov(x(weights,:));
        muhat = mean(x(weights,:),1);
        %check for the condition of singular covariance matrix:
        if cond(Chat) > 1E8
            mu = muhat;
            C = Chat;
            dists_final(:) = inf;
            weights_final = weights;

            mcdout = mcd_results(mu,C,h,dists_final, nanMask);

            warning('Encountered singular covariance matrix;
               exiting early.');
            return
        end
        dists = d_quad(x, Chat, muhat);

        %xc = x - ones(nObs,1)*muhat; %center the data

        %L = chol(Cinv); %cholesky decomposition, L is upper
           triangular, L’*L ==Cinv
        %dists = sum((L * xc’).^2,1);

        [~, sortOrder] = sort(dists); %1st h elements of
           sortorder are indices of the new set.
        newWeights = false(N,1);
        newWeights(sortOrder(1:h)) = true;

        weights(:) = newWeights(:);
    end
    det_Chat = det(Chat); %if this new estimation is in the top
        10, keep it:
    [winnerThreshold winnerIndex] = max(det_C_best10);
    if det_Chat < winnerThreshold
        %C = Chat;
%         det_C = det_Chat;
%         mu = muhat;
%         dists_final = dists;
%         weights_final = weights;
        det_C_best10(winnerIndex) = det_Chat;
        weights_best10(:,winnerIndex) = weights;
    end

end

%iterate the best 10 results until convergence:
for i = 1:nSetsToKeep
    weights = weights_best10(:,i);
    %iterate until convergence:
    flag = true;
    while flag
        Chat = cov(x(weights,:));
        muhat = mean(x(weights,:),1);

        dists = d_quad(x, Chat, muhat);

%         xc = x - ones(nObs,1)*muhat; %center the data
%         Cinv = Chat^-1;
%
%         L = chol(Cinv); %cholesky decomposition, L is upper
   triangular, L’*L ==Cinv
%         dists = sum((L * xc’).^2,1);
```

```
        [~, sortOrder] = sort(dists); %1st h elements of
           sortorder are indices of the new set.
        newWeights = false(nObs,1);
        newWeights(sortOrder(1:h)) = true;

        if all(newWeights == weights)
            flag = false;
        else
            weights(:) = newWeights(:);
        end
    end

    det_Chat = det(Chat); % if the newly calculated value is
       best, keep it:
    if det_Chat < det_C
        C = Chat;
        det_C = det_Chat;
        mu = muhat;
        dists_final = dists;
        weights_final = weights;
    end
end


%apply correction factor to make the median go in the right
   place:
%corFac = median(dists_final)./chi2inv(0.5, nDims);
dist_sort = sort(dists_final);
corFac = dist_sort(h) / chi2inv(h/nObs, nDims);

C =  corFac .* C; %MCDCOV is slightly different, it goes by
   quantile rather than the median.

%do re-weighting:
dists_final = dists_final / corFac;
%all(dists_final == d_quad(x,C,mu))
weights_final = dists_final <= chi2inv(reweightTol, nDims);
C = cov(x(weights_final,:));
mu = mean(x(weights_final,:),1);
dists_final = d_quad(x, C, mu);

mcdout = mcd_results(mu,C,h,dists_final, nanMask);

function mcdout = mcd_results(mu, C, h, dists, nanMask)
nObs = size(nanMask,1);
%build results structure:
mcdout = struct(...
    ’center’,mu,...
    ’cov’,C,...
    ’cor’,[],...
    ’h’,h,...
    ’alpha’, h./nObs,...
    ’md’,[],...
    ’rd’, nan(nObs,1),...
    ’cutoff’,[],...
    ’flag’,[],...
    ’method’,[],...
    ’plane’,[],...
    ’classic’,[],...
    ’class’,[],...
    ’X’,[]);
mcdout.rd(~nanMask) = sqrt(dists);

% mcdout.center = mu;
% mcdout.cov = C;
% mcdout.cor = [];
% mcdout.h = h;
% mcdout.alpha = h./ nObs;
% mcdout.md = [];
% mcdout.rd = sqrt(dists);
% mcdout.cutoff = [];
% mcdout.flag = [];
% mcdout.method = [];
% mcdout.plane = [];
% mcdout.classic = [];
% mcdout.class = [];
% mcdout.X = [];



function dists = d_quad(x, C, mu) %returns distance squared,
   not distance!
nObs = size(x,1);
xc = x - ones(nObs,1)*mu; %center the data
Cinv = C^-1;

L = chol(Cinv); %cholesky decomposition, L is upper triangular,
    L’*L ==Cinv
dists = sum((L * xc’).^2,1);
```

B.4 Monte Carlo simulation code

roc_arc.m: Perform Monte Carlo simulation and generate ROC curves, given an input CSV file containing noisless data

```
%roc_arc

filename = ’star_gfault_100ohm_4modules.csv’;
```

```
error_pct = 1;
sigmaI = .01; %amps
sigmaV = .01; %volts
nTrials = 100;
nSer = 13;
nPar = 4;
alpha = 0.5;
thresh_mcd = zeros(1,2*nTrials);
thresh_mcd2 = thresh_mcd;
thresh_zscore_I = thresh_mcd;
thresh_zscore_V = thresh_mcd;
thresh_mahal = thresh_mcd;
thresh_likelihood = thresh_mcd;
label = thresh_mcd;

[volts0, amps0] = ltspice2matlab(filename, nSer, nPar, 0,0);
dims = size(volts0);

for i = 1: nTrials
    %generate examples of the faulty case
    i
    volts = volts0 + error_pct*.354*randn(dims);
    amps = amps0 + error_pct*.0495 * randn(dims);
%        scatter(volts, amps)
%        xlim([0 60]); ylim([0 6]);
%        pause(.02);

    indata = [volts(:) amps(:)];
    mcdout = mcdcov(indata,'alpha',alpha,'plots',0);
    thresh_mcd(i) = max(mcdout.rd);

    mcdout = mcdcov(indata,'alpha',0.75,'plots',0);
    thresh_mcd2(i) = max(mcdout.rd);

    thresh_zscore_I(i) = max(abs(zscore(amps(:))));
    thresh_zscore_V(i) = max(abs(zscore(volts(:))));
    thresh_mahal(i) = max(mahal(indata,indata));
    thresh_likelihood(i) = 1/likelihood_gauss(indata);

    label(i) = 1;

end

%generate examples from the non-faulty case
filename = 'STAR_baseline.csv';
[volts0, amps0] = ltspice2matlab(filename, nSer, nPar, 0,0);

for i = nTrials+1 : 2*nTrials
```

```
    i
    volts = volts0 + error_pct*.354*randn(dims);
    amps = amps0 + error_pct*.0495 * randn(dims);

    %[volts, amps] = ltspice2matlab(filename, nSer, nPar,
       sigmaI, sigmaV);
    indata = [volts(:) amps(:)];
    mcdout = mcdcov(indata,’alpha’,alpha,’plots’,0);
    thresh_mcd(i) = max(mcdout.rd);

    mcdout = mcdcov(indata,’alpha’,0.75,’plots’,0);
    thresh_mcd2(i) = max(mcdout.rd);

    thresh_zscore_I(i) = max(abs(zscore(amps(:))));
    thresh_zscore_V(i) = max(abs(zscore(volts(:))));
    thresh_mahal(i) = max(mahal(indata,indata));
    thresh_likelihood(i) = 1/likelihood_gauss(indata);

    label(i) = 0;

end

[rocX_mcd,rocY_mcd] = get_roc(thresh_mcd,label);
[rocX_mcd2,rocY_mcd2] = get_roc(thresh_mcd2,label);
[rocX_zscore_I,rocY_zscore_I] = get_roc(thresh_zscore_I,label);
[rocX_zscore_V,rocY_zscore_V] = get_roc(thresh_zscore_V,label);
[rocX_mahal,rocY_mahal] = get_roc(thresh_mahal,label);
[rocX_likelihood,rocY_likelihood] = get_roc(thresh_likelihood,
   label);

plot(rocX_mcd,rocY_mcd,rocX_mcd2,rocY_mcd2,rocX_zscore_I,
   rocY_zscore_I,...
    rocX_zscore_V,rocY_zscore_V,rocX_mahal,rocY_mahal,
       rocX_likelihood,...
    rocY_likelihood);
legend(’MCD (\alpha = 0.5)’, ’MCD (\alpha = 0.75)’,’Z-score (I)
   ’,’Z-score (V)’,’Mahalanobis distance’,’Likelihood under
   Gaussian’);
xlabel(’Probability of false alarm (%)’);
ylabel(’Probability of detection (%)’);
```

get_roc.m: Generate ROC curve given classifier output and ground truths

```
function [rocX, rocY] = get_roc(inData, groundTruths)
nClass0 = sum(~groundTruths);
nClass1 = sum(groundTruths);

[data] = sortrows([inData(:) groundTruths(:)],-1);

rocX = zeros(length(inData)+1,1); rocY = rocX;
```

```
for i = 1:size(data,1)
    %get number of false positives:
    rocX(i + 1) = rocX(i) + ~data(i,2);
    %get number of true positives:
    rocY(i + 1) = rocY(i) + data(i,2);

end

%normalize to get percentages:
rocX = rocX/nClass0;
rocY = rocY/nClass1;
```

B.5 MATLAB implementation of Sandia inverter performance model

```
function [efficiency Pac] = sandia_inverter(inverter, Vdc, Pdc)

Idc = Pdc / Vdc;
A = inverter.P_dco * (1 + inverter.C1 * (Vdc - inverter.V_dco))
   ;
B = inverter.Pso * (1 + inverter.C2 * (Vdc - inverter.V_dco));
C = inverter.Co * (1 + inverter.C3 * (Vdc - inverter.V_dco));

Pac = (inverter.P_aco/(A-B))*(Pdc - B) - C*(A-B)*(Pdc - B) + C
   *(Pdc - B)^2;

if Vdc > inverter.MPPThigh || Vdc > inverter.Vdcmax || ...
        Vdc < inverter.MPPTlow || Idc > inverter.Idcmax || Pac
           > inverter.P_aco
    Pac = 0;
end

efficiency = Pac / Pdc;
```