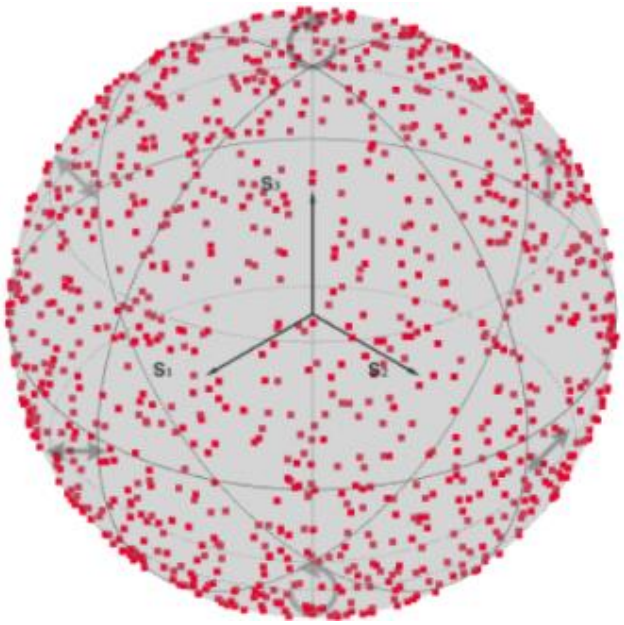
APPLICATION NOTE

All-States and Matrix Methods for PDL or PER

Polarization Scrambler and Stabilizer

Polarization controllers like the N7785C and N7786C can repeatably run long sequences of polarization states, synchronized to other instruments with triggering signals. The N7786C can also set and stabilize chosen polarization states. These can be used to improve accuracy measuring polarization dependent loss (PDL) or polarization extinction ratio (PER). The principles and programming are described in this note.



KEYSIGHT TECHNOLOGIES

Table of Contents

Standard PDL Measurement Methods

There are two general methods for measuring the polarization dependent insertion loss (PDL) of a fiberoptic component, which are standardized in the IEC document 61300-3-2 as the all-states method and the Mueller Matrix method.

In the all-states method, the optical test signal to the device under test is varied through a sufficiently large sampling of all possible polarization states such that the relation between the maximum and minimum optical power measured at the device output represents the PDL with the necessary accuracy.

For the Mueller Matrix method, a small set of known polarization states, typically 4 or 6 states, are applied to the DUT and matrix analysis of the output power levels is used to determine the maximum and minimum power that would be found by scanning all of the possible states. This is generally faster and is well suited for spectral measurements with continuously swept tunable lasers. However, this method requires instrumentation that accurately sets specific states of polarization and somewhat more advanced control programs for derivation of the results.

For components with high polarization extinction ratio (PER) like polarizers and polarizing beam splitters, the standard Mueller Matrix method also typically has insufficient accuracy to measure extinction beyond about 20 dB, because the lowest power level is not measured directly and the calculated value is very sensitive to small variations in the measured power levels and their corresponding polarization states. An extension of the method, used in method 5 of the application note for polarization alignment, gets past this limitation by calculating and then setting the polarization for the minimum power. https://www.keysight.com/us/en/assets/3120-1564/application-notes/Polarization-Alignment-Methods.pdf.

Easy programming with new polarization controllers

This application note is an adaptation of an earlier paper to match the functionality and SCPI command set of the N778-C generation polarization controllers.

A follow-up note details using advanced functionality for also aligning polarization.

The all-states method is a straight-forward way to measure components with little wavelength dependence, so that only one or a few wavelengths need to be measured. Fiberoptic couplers, splitters and isolators are typical components to test this way. Similarly, tests of polarization beam splitters and other devices designed for high PER benefit from this method.

The implementation of the all-states PDL method described in this note exploits the special characteristics of the Keysight N7785C synchronous scrambler, also available in the N7786C and N7788C, which can be programmed for repeatable stepping through a sequence of polarization states at high speed while producing synchronization triggers. This can be used to shorten total measurement time, allow optimized detector averaging times, and normalize the results to remove the polarization dependence of the setup from the results, delivering average insertion loss (IL) at the same time.

The implementation of the Mueller Matrix method in this note takes advantage of the built-in real-time polarization monitoring of the N7786C to provide fast stabilized setting of the chosen polarization states as well as to measure the optical power output from the polarization controller to the device under test (DUT) at the same time as the power output from the DUT is measured. This provides very accurate and stable measurements, especially for devices with very low PDL.

Apparatus

The synchronized all-states PDL test setup is shown schematically in Fig. 1. A stable polarized signal, like from a DFB or Fabry-Perot laser, is applied to the synchronous scrambler. The N7785C is the best match for this application, but the N7786C and N7788C can also be used in the same way. An optical power meter that logs a series of power measurements, synchronized with triggers from the polarization scrambler, completes the instrumentation.

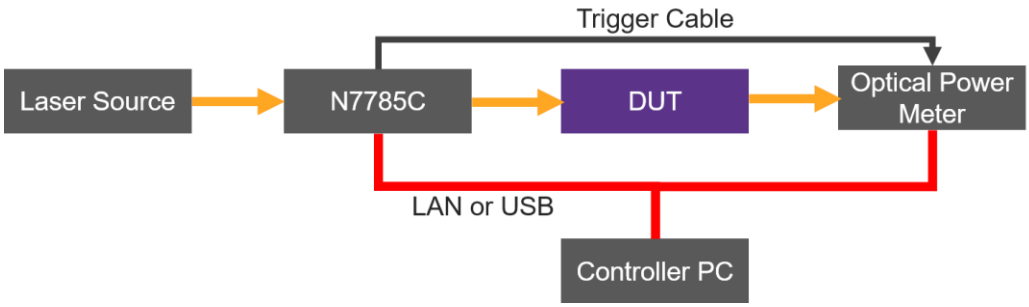


Figure 1. Schematic setup for synchronized all-states PDL measurement.

The setup is similar for the Mueller Matrix method described later but uses the N7786C and does not require hardware triggering to the optical power meter.

The power meter can be selected to best match the measurement need. The uncertainty contribution from polarization dependence of the detector and other instrumentation after the DUT, like switches or splitters, can generally not be eliminated with a reference measurement, because the polarization states at the output of the DUT are not the same as during the reference without DUT. Therefore, the specified polarization dependence of the power meter should be lower than the desired PDL measurement uncertainty. For the lowest PDL, the N7747C or N7748C optical power meters or the 81624C remote power head are recommended. Either can log up to 1M samples with an averaging time that can be set down to 100 µs. For measuring high PER, where a sequence with many samples is needed to approach the lowest power, the 25 µs or shorter averaging times of the N7742C, N7744C and N7745C multiport power meters reduces the needed measurement time. The higher dynamic range during logging for these instruments also supports the measurement of PER beyond 30 dB.

The optical fiber connection between the laser source and the scrambler should be stable so that the state of polarization is repeatable. A polarization maintaining fiber can be used to improve this stability if the source has PMF output. For adjustable wavelength, a tunable laser like the N7779C can be used. Multiport measurements, such as on PON splitter devices, can be made easily using one or more 8-port N7745C optical power meters to measure all ports simultaneously.

All-States Measurement Description and Parameter Choices

The synchronous scrambler is configured with a sequence of polarization settings that provide sufficient coverage of the Poincaré sphere. These settings can be randomly distributed, since the scrambler switches very rapidly to any other state of polarization (SOP). The instrument is programmed to itself generate the sequence of random states. For measurement of low PDL values, it is usually sufficient to use a hundred settings and even 1000 settings can usually be scanned in a sufficiently short time. For measurement of high PER values, like 20 dB or above, a sequence of 10k to 20k samples should be used.

The power meter is configured to log the power for each of the polarization settings, so the length of this logging sequence is the same as for the polarization sequence. In many applications, a fast averaging time for the power readings can be chosen, like 100 µs or even 25 µs if supported by the power meter. However to reduce noise for measuring PDL values substantially below 0.1 dB, it may be necessary to increase the averaging time and possibly to use the coherence control modulation of the laser source. In the case of coherence control, a long averaging time like 10 ms should be chosen to average out the power variations due to the modulation. The power meter is configured to make a single measurement for each trigger applied to the external input from the polarization scrambler.

The repetition rate of the polarization sequence is then set to give a switching period that accommodates the averaging time of the power meter plus enough time for the SOP and the new power reading to settle. The triggering to the power meter is delayed for this settling time. The switching period must be long enough that the power meter averaging time is elapsed before the next SOP switch. At least 20 µs holdoff is needed to allow the new state of SOP to be reached. The optical power meter may require longer settling time, depending on the analog bandwidth of the detection circuit. In the case of the N7742C, N7744C and N7745C, the bandwidth is matched to the chosen averaging time, so a recommendation is for a trigger period twice as long as the selected averaging time, with the trigger hold-off set to 40% of the trigger period, allowing 10% buffer between the end of averaging and the next polarization step. This assumes using an averaging time not shorter than 25 µs. After thus configuring the sequence operation it is advisable to run the sequence once before measuring, so that each successive sequence after this has the same starting condition.

Now a reference measurement can be made by connecting the optical signal that will later be applied to the DUT directly to the power meter or otherwise to the path that will be connected to the DUT output. Then run the sequence and save the series of power values. These will usually have a range of about 0.5 dB, primarily due to the polarization dependence of the scrambler, and can be used to remove this influence from the results.

These reference results can be used for normalizing measurements over the next minutes to hours, depending on the PDL accuracy needed and the environmental stability of the setup. For example, the state of polarization input to the controller may change over time, which changes the SOP pattern produced by the sequence. The validity of the reference can be easily checked by making the same connection and repeating the measurement. The difference in the traces can be reviewed, most simply by using the same calculation used to find PDL from the reference and measurement traces, shown below. If the result exceeds the desired uncertainty, then a new reference is needed. This same procedure can be used to check the repeatability and noise level directly after the reference measurement, for example to determine whether the coherence control of the laser source is needed.

Then insert the DUT into the setup and run the sequence again to measure its polarization dependence. These results can be saved as a trace for later evaluation or used immediately to reduce the results to the PDL value, as described below. The sequence can be further repeated on this DUT or successive devices.

To obtain the PDL result from the reference trace P_i^R and the device trace P_i^D, where the index *i* ranges from 1 to *N*, the length of the sequence, first calculate the normalized transmittance trace *T* as

$$T_i = \frac{P_i^D}{P_i^R} \quad , \qquad \text{(Equation 1)}$$

where all values *P* are expressed in linear power units, such as mW.

The PDL (or PER) is then calculated from the minimum and maximum of the *N* values in the trace T_i according to

$$PDL = 10log\left(\frac{\max(T_i)}{\min(T_i)}\right) \quad . \qquad \text{(Equation 2)}$$

Making the same evaluation for different lengths or segments of the sequence can be used to investigate the sequence length needed for reliable measurements.

When a reference measurement is made in this way, the polarization averaged insertion loss (IL) can also be calculated, according to

$$IL = 10log\left(\frac{\max(T_i)+\min(T_i)}{2}\right) \quad . \qquad \text{(Equation 3)}$$

Programming the all-states method

The commands used for a basic program to make these measurements are described here. For full details on use of the commands, please refer to the respective programming guides. In this example, the laser source is not programmed but assumed to provide a stable input signal to the N7785C.

First it is good to start from a known condition by presetting both the N7785C instrument ‘A’ and the power meter instrument ‘B’. Especially for the polarization controller, it may sometimes be necessary to increase the VISA timeout for this step beyond the 2 s default value, like to 5 s. The repeatablity of the polarization sequence may be better if the preset is not performed immediately before generating and using the sequence.

Label	Command	Example	Comment
A0 B0	SYSTem:PRESet	SYST:PRES;*OPC?	preset for known start conditions; use OPC? to wait for completion

Then the power meter or power meters can be set up for triggered logging for each power meter *n* before activating the logging. Note that on N774x multiport instruments there is common triggering for some or all of the ports, so it can be necessary to set all ports, including ports not used for the measurement, to the same averaging time and triggering mode.

Label	Command	Example	Comment
B1	:SENSe[n]: POWer:RANGe:AUTO	SENS1:POW:RANG:AUTO 0	disable automatic range switching
B2	:SENSe[n]:POWer:RANGe	SENS1:POW:RANG 10DBM	set the best power range for the maximum power level
B3	:SENSe[n]: POWer:GAIN:AUTO	SENS1:POW:RANG:AUTO 0	disable automatic gain switching if using random scrambling with N774x instruments that support this.
B4	:TRIGger[n]:INPut	TRIG1:INP SME	logs a single sample for each input trigger
B5	:SENSe[n]: POWer:WAVelength	SENS1:POW:WAV 1550nm	Optionally set the wavelength of the laser source to provide calibrated absolute power measurements; not required for IL and PDL
B6	:SENSe[n]: FUNCtion:PARameter:LOGGing	SENS1:FUNC:PAR:LOGG 1000,100US	Configure the number of samples and averaging time for the sequence

Next the polarization controller is configured.

Label	Command	Example	Comment
A1	:PCONtroller:GEN:SCRAmble? or :PCONtroller:GEN:RANDom?	PCON:GEN:SCRA? 1000 or PCON:GEN:RAND? 1000,1000	Generate a random scramble sequence (or a 'random walk') with chosen length (and step scale)
A2	:PCONtroller:SAVE:ARB?	PCON:SAV:ARB?	Optionally save the sequence internally for later loading the same sequence
A3	:PCONtroller:SEQuence:DCOMpensation	PCON:SEQ:DCOM 1	For best polarization stability, the default drift compensation should be active
A4	:PCONtroller:REPetition	PCON:REP 1	The sequence will be run once after starting the sequence function
A5	:PCONtroller:SEQuence:RRATe	PCON:SEQ:RRAT 5	Example 5 kHz switching to allow 100 µs averaging time plus 100 µs for polarization switching
A6	:PCONtroller: SEQuence:HOLDoff	PCON:SEQ:HOLD 2560	Delays the trigger to the power meter by 80 µs for settling the new SOP power reading
A7	:PCONtroller: SEQuence:SMODe	PCON:SEQ:SMOD 2	The sequence will run when a trigger is received
A8	:TRIGger:CONFiguration	TRIG:CONF 1	default triggering connections
A9	:PCONtroller:STARt	PCON:STAR	Enable the scrambler
A10	*OPC?	*OPC?	This should be polled until it returns '1' and the controller is ready to be triggered
A11	:TRIGger	TRIG 1	Trigger an initial sequence to get a repeatable starting condition for succeeding sequences

Here the program should wait for the expected duration of the sequence, in this example for 1000 states at 5 kHz then 200 ms, before continuing. Add at least 100 ms before continuing, to be sure the sequence finishes. In the succeeding runs of the sequence below, this is assured by the status of the power meter logging.

Now the setup is ready to make a reference measurement by connecting the output from the N7785C to the power meter.

Label	Command	Example	Comment
B7	:SENSe[n]:FUNCtion:STATe	SENS1:FUNC:STAT LOGG,STAR	Logging of each power meter is activated and waiting for triggers
B8	:SENSe[n]:FUNCtion:STATe?	SENS1:FUNC:STAT?	this query responds with LOGGING_STABILITY,PROGRESS when the power meter is ready;
A12	:PCONtroller:STARt	PCON:STAR	Enable the scrambler
A13	*OPC?	*OPC?	This should be polled until it returns '1' and the controller is ready to be triggered
A14	:TRIGger	TRIG 1	
		wait	Again wait for the duration of the sequence
B9	:SENSe[n]:FUNCtion:STATe?	SENS1:FUNC:STAT?	this query is repeated until the response changes from LOGGING_STABILITY,PROGRESS to LOGGING_STABILITY,COMPLETE
B10	:SENSe[n]:FUNCtion:RESult?	SENS1:FUNC:RES?	This query is used for each power meter to upload the data arrays; the data format is described below

This uploads the reference trace for Eqn. 1 above. The resulting data are an array of 32-bit optical power values from each power meter. The data are transmitted in binary block format, as described in the Programming Guide. Expressing this binary data as real arrays requires attention to the byte ordering. The first few bytes represent ASCII characters. The first is the symbol '#', followed by a digit that gives the remaining number of bytes to interpret as characters. These give a number that tells how many bytes of data follow. For example, '#3808'... indicates 808 bytes of data are contained, corresponding to 202 32-bit power values. Each value is transmitted least significant byte first (LSBfirst, little-endian, 'Intel byte order').

The Command Expert instrument command sets or the 816x Plug&Play driver simplify programming with automatic reading of binary blocks, providing the correct formatting. Or using direct IO with VEE, LSB can be selected for the byte ordering in the Instrument Manager under Advanced Instrument Properties and the binary block format can be selected for automatic parsing during the read operation. Other programming environments use other ways to format the input data.

In some cases, it may be most convenient to handle reading and parsing of binary blocks at the lower VISA level. That is supported by the VISA COM I/O with its IFormattedIO488 interface, setting the property InstrumentBigEndian to ‘false’ and using the READIEEEBlock method. For the power meters, the parameter ‘type’ is set to BinaryType_4. When programming with python, pyvisa can be used and also provides formatting for reading and writing binary values with the appropriate byte ordering.

Then the DUT can be connected in the optical path, like in Fig. 1, and the steps from B7 to B10 repeated to obtain the device trace for Eqn. 1. This allows the calculation of PDL and IL. Further DUT measurements can be made in the same way, using the same reference data for as long as the stability of the system maintains sufficient validity of these data.

When the program is closed, it is good to stop the power meter logging function and the polarization controller operation.

- SENS1:FUNC:STAT LOGG,STOP
- PCON:STOP

Characteristics of typical measurement examples

Quick PDL measurements

The fast SOP switching of the synchronous scrambler allows a fast sampling rate for a short measurement time. The 100 µs averaging time, which is the minimum for the optical power meters with lowest polarization dependence, can be used with a 5 kHz repetition rate, so a sequence of 1000 states runs in 200 ms. A section of such a sequence is shown in Figure 2, to observe the repeatability of the sequence 10 minutes after the reference. The corresponding PDL calculation in this run gave 0.03 dB.

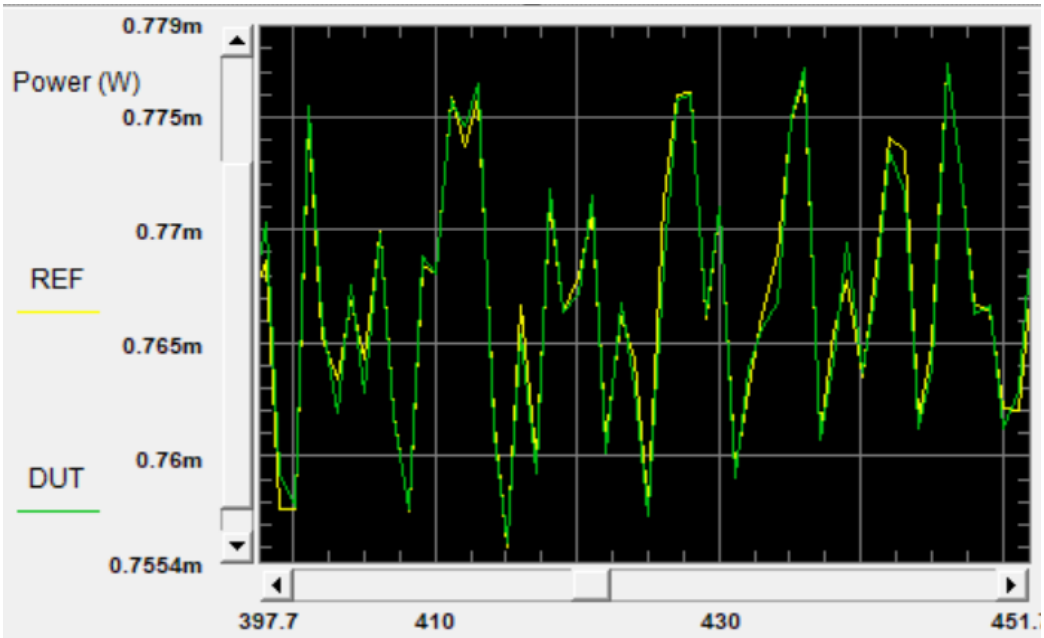


Figure 2. Repeatability of a sequence using 100 µs averaging time, after 10 minutes. The yellow trace it the original reference and the green trace is the succeeding measurement.

For measuring PDL values up to 1 dB, about 100 samples are sufficient for the min-max PDL to come within 10% of the full PDL value. So a good measurement sequence runs in 20 ms.

Very low PDL

For measuring PDL values significantly below 0.1 dB, the repeatability mentioned above is a limitation. In this case it is useful to increase the power meter averaging time. Using 10 ms averaging time with a stable setup has been seen to give repeatability corresponding to less than 0.005 dB over times of 10 minutes or more. The 10 ms averaging time also supports use of the coherence control function of the laser sources, if needed to avoid interference effects due to reflections in the setup. Again, for these values, good measurements are obtained with sequence lengths of about 100.

In practice, it may be difficult to reliably get such stable repeatability. With the extended functionality of the N7786C for stabilization and monitoring, the Mueller Matrix method described later provides a more dependable approach.

High polarization dependence – PER Test

The range of high extinction ratio measurements is limited by the determination of the lowest transmission value. When using a random pattern of SOP, this is improved by using many samples and having minimum SOP variation during the averaging time of the sample. This latter condition is an advantage of the polarization switching rather than continuous scanning. To assure measurements above 30 dB PER, a minimum of 20k samples is recommended. To minimize the measurement time, short averaging times can be used. For example, using 100 µs averaging time and 5 kHz repetition, the 20k sequence requires 4 s.

Reducing the averaging time to 25 µs can be used for faster measurement. But for example if the rate is increased to 10 kHz, the 17 kHz bandwidth of the 81636B power meter can limit the reliable results to about 27 dB. For higher PER it is better to reduce the repetition rate to 5 kHz for 25 µs averaging time, or the N7744C and N7745C can be used at 10 kHz or even 20 kHz at 25 µs averaging, due to the higher analog bandwidth.

As an example of the dependence on sequence length, Figure 3 shows how the minimum value of transmission through a linear polarizer changes during a particular 50k sequence. On this graph, the value 0.001 corresponds to measuring 30 dB PER. The graph shows that this sequence had at least 2 points below this value within 20k samples.

The impact of power meter bandwidth limitation can be reduced by avoiding large power changes from sample to sample. This can be done using the 'random walk' scrambling pattern, mentioned in step A1. But then the sequence must be long enough to give good coverage of all possible SOP. The parameter pair 10000, 1000 for number of points and scale of the steps may be a good choice.

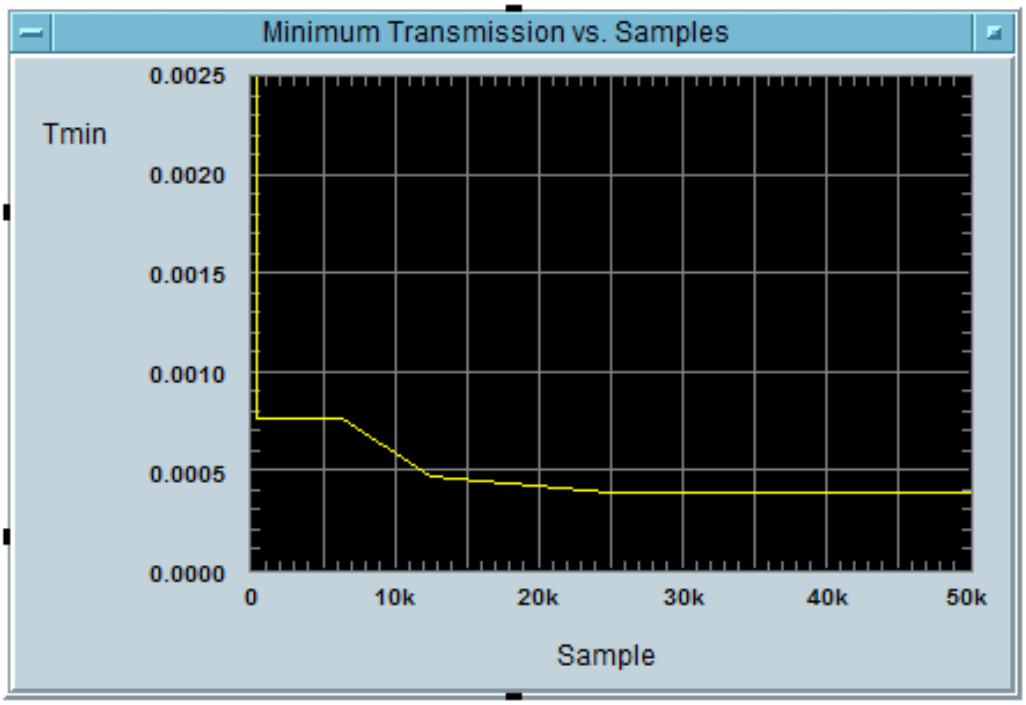


Figure 3. Development of the T_{min} value over 50k samples

Estimates for required sequence length

Over all possible states of polarization, which can be represented as points on the surface of a Poincaré sphere, the transmittance varies within the full range T_{min} to T_{max} . The corresponding states of polarization, Z_{min} and Z_{max} , represent two poles of the sphere that can be considered as the endpoints of the z axis, which we scale here from 0 to 1. The transmittance at any SOP is determined by the projection of its position on the sphere onto the z axis, and the value T scales linearly with z between T_{min} and T_{max} , that is

$$T(z) = z(T_{max} - T_{min}), 0 \leq z \leq 1 \quad . \qquad \text{(Equation 4)}$$

The density of surface area on the sphere and thus of polarization states is uniform along z between Z_{min} and Z_{max}.

A finite sequence of states will generally cover a shorter measured range, not reaching the extreme values, and result in a lower measured PDL. Thus, the uncertainty of the measured PDL value depends on the length of the sequence. The influence of this underestimation of the range on the calculation is somewhat different for the cases of low PDL or high PER as discussed below.

For measurements of high polarization dependence, like PER above 20 dB, the influence on the result is asymmetric with respect to a gap in sampling close to Z_{min} and Z_{max}. For example at 20 dB, T_{min} = 0.01 × T_{max} . A 1% sampling gap at the maximum only underestimates the PER by about 0.04 dB, but the same gap at the minimum causes a 3 dB underestimate. Therefore, the sufficiency of a sequence length in this asymmetric case is dominated by the probability of sampling within a given distance from Z_{min}. The probability that at least one of N samples is with the range $0 \leq z \leq a$ is also the probability that not all of the samples fall in the range $a \leq z \leq 1$. Assuming a random sampling with uniform distribution along z , this probability is given by

$$P = 1 - (1 - a)^N \qquad \text{(Equation 5)}$$

So, for the 20 dB PER requirement example, we could choose a value 0.002 for a. This leaves an underestimate of about 0.8 dB at 20 dB and a 21 dB PER will give at least a 20 dB result. As shown in Figure 4, a sample length of 3000 or more can provide this coverage with very high confidence.

To achieve a similar 1 dB level of uncertainty for a 30 dB measurement, a 10-times smaller value of a at 0.0002 must be used. As shown in the figure, this would indicate using a sequence of about 30k samples. A compromise could be a = 0.0005, which allows a 3 dB underestimate at 33 dB PER and can be covered by about 12k samples with very high confidence.

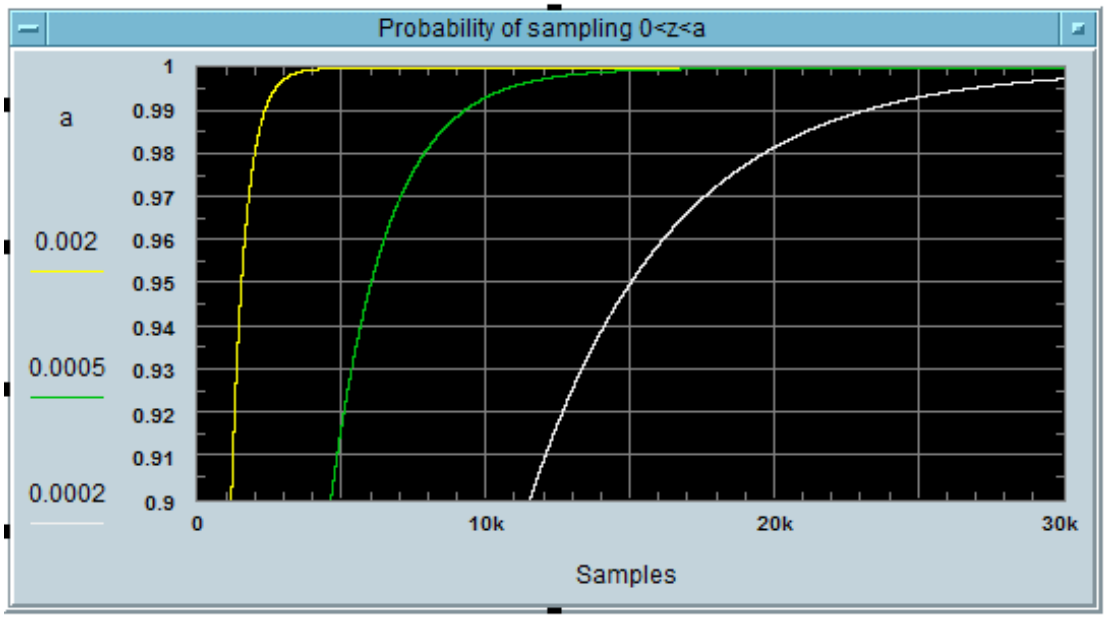


Figure 4. Estimate of the confidence of sampling T_{min} within a fraction a.

Please note that the programming procedure described above for randomizing the sequence of polarization states cannot assure a truly uniform distribution. But this is a reasonable approximation for these estimates of sequence length. The validity of the approximation can be checked by measuring a polarizer known to have PER higher than about 35 dB. Increasing the sequence length could be used to compensate for any nonuniformity.

The second case, measuring the polarization dependence of devices with low PDL, has a nearly symmetric dependence on sampling gap from Zmin or Zmax . Based on Eqn. 2, it can be found that if the sampled values cover 90% of the full range, leaving a 10% gap, then for PDL values below about 1 dB, the PDL values in dB will also be underestimated by about 10%. For these PDL values, the position of the measured range between Tmin and Tmax does not change the calculated result significantly.

In this case, the probability that the samples cover at least the range r is the probability that the sample with lowest z is in the range 0 ≤ z1 ≤ 1-r and that the sample with highest z is in the range z1+r ≤ z2 ≤ 1. This results in the following equation.

$$P = 1 - r^N - \int_0^{1-r} (1-x)^{N-1} (r+x)^N \, dx \qquad \text{(Equation 6)}$$

Note that without the integral term, this equation is identical to Equation 5 when r =1− a . The additional term is added because the target range can be shifted up or down by each sample. The development of this confidence measure with sample length is shown in Figure 5, for several values of r .

As shown in the figure, a high confidence for less than 10% PDL (r = 0.9) uncertainty can be reached with about 75 samples. An uncertainty due to coverage of 1% could be reached with about 750 samples, but then the noise and repeatability limits of the power measurements will likely dominate the total uncertainty.

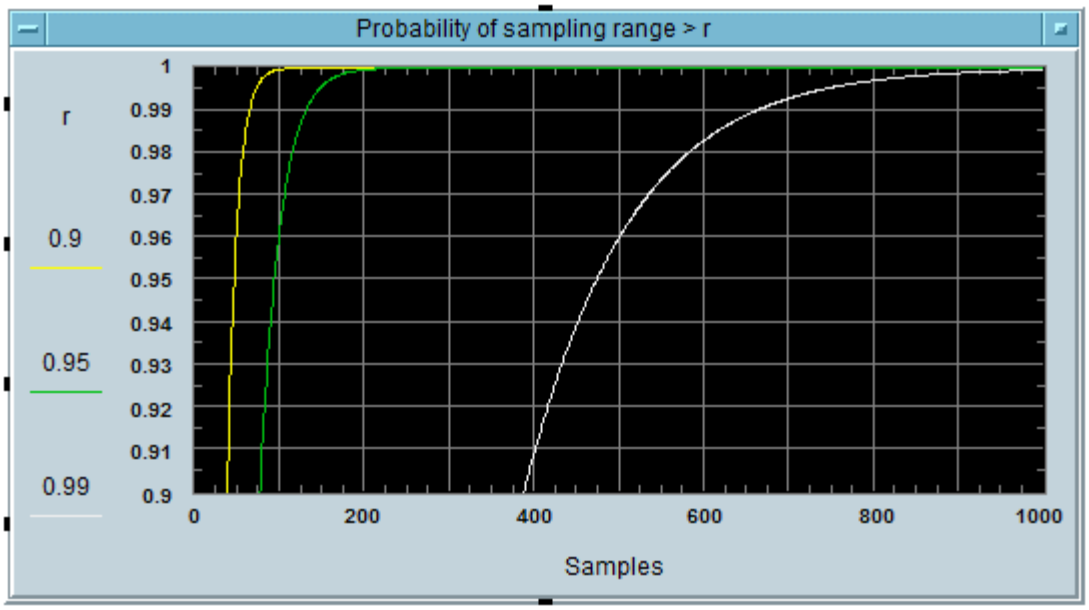


Figure 5. Estimate of the confidence of sampling T within a fractional range r.

Mueller Matrix Measurement Description and Programming

This method is based on measurements from a small set of SOP with known orientation to each other although usually arbitrary orientation to the device axes. Most commonly, this is done using 4 SOP with a simple relationship in the Stokes vector space. The N7786C stabilizer function can be used to set these states, at which the signal level is then measured, usually first without the DUT as reference power values and then including the DUT. From these values, the IL and PDL can be calculated.

Besides accurately setting and stabilizing the SOP, the N7786C also allows real-time monitoring of its output power, which is then applied to the DUT, so that both the input and output signals of the DUT are measured at the same time. This corrects the PDL results for any fluctuations in power level between the reference and the DUT measurement, that may be caused by power or SOP drift between the laser source and the N7786C. This is especially valuable for dependable measurements of PDL below 0.01 dB.

The setup is again like in Figure 1, but uses the N7786C and usually does not use the trigger connections, since the synchronization is not critical and can be handled by the program. The N7788C can be used in a similar way, either by adding an external optical splitter to tap part of the output signal from the polarization controller to the input port for the polarimeter.

The measurements result in 4 reference power values PiR and 4 device power values PiD , expressed in linear units like mW, for the states with Stokes vectors (s1,s2,s3) indexed shown in the equation 7 as:

Index 1. (1,0,0)

Index 2. (-1,0,0)

Index 3. (0,1,0)

Index 4. (0,0,1) (Equations 7)

When the above mentioned power monitoring of the N7786C output is used for normalization, the above values P_i^R and P_i^D are each determined as the ratio of the raw power value at the power meter (p_i^R and p_i^D) to the power value measured with the N7786C (m_i^R and m_i^D) at the same time. For example: $P_1^R = p_1^R / m_1^R$.

From these, the 4 top-row elements of the Mueller matrix are calculated.

$$m_1 = \frac{1}{2}\left(\frac{P_1^D}{P_1^R} + \frac{P_2^D}{P_2^R}\right)$$

$$m_2 = \frac{1}{2}\left(\frac{P_1^D}{P_1^R} - \frac{P_2^D}{P_2^R}\right)$$

$$m_3 = \frac{P_3^D}{P_3^R} - m_1$$

$$m_4 = \frac{P_4^D}{P_4^R} - m_1$$ (Equations 8)

Further calculation uses an intermediate term for the relative degree of polarization dependence, here labeled *d*.

$$d = \sqrt{{m_2}^2 + {m_3}^2 + {m_4}^2}$$ (Equation 9)

The maximum and minimum values for *T*, used in Eqns. 2 and 3 to calculate IL and PDL are given by:

$$T_{max} = m_1 + d$$

$$T_{min} = m_1 - d$$ (Equation 10)

Programming the Mueller Matrix method

As always, the procedure can start with a preset, if needed to establish a known condition. Otherwise the only critical setting in preparation is the wavelength setting for the polarimeter calibration.

Label	Command	Example	Comment
A0 B0	:SYSTem:PRESet	SYST:PRES;*OPC?	Preset for known start conditions, if needed; use OPC? to wait for completion
A1	:POLarimeter:GAIN	POL:GAIN 0	Set gain level as needed for input signal level
A2	POLarimeter:WAVelength	POL:WAV 1550NM	Calibrate polarimeter to the input signal wavelength
A3	:STABilizer:STABilize	STAB:STAB 1	Activate stabilizer function

Here the program should wait for 2 seconds for the activation process. Then the stabilizer can remain activated for the rest of this procedure.

Before or during this wait, the optical power meter(s) can be configured, so that the N7786C power monitor values can be read during the same time as the power meter measurement. These settings are best applied to all ports of a multiport power meter to assure the intended timing.

Label	Command	Example	Comment
B1	:SENSe[n]: POWer:RANGe:AUTO	SENS1:POW:RANG:AUTO 0	disable automatic range switching
B2	:SENSe[n]:POWer:RANGe	SENS1:POW:RANG 10DBM	set the best power range for the maximum power level (this could be reset for later power readings if needed for power readings that vary over more than 30 to 40 dB
B3	:SENSe[n]:POWer:ATIMe	SENS1:POW:ATIM 0.1	set the averaging time
B4	:INITiate[n]: CONTinuous	INIT1:CONT 0	disable continuous triggering
B5	:SENSe[n]: POWer:WAVelength	SENS1:POW:WAV 1550nm	Optionally set the wavelength of the laser source to provide calibrated absolute power measurements; not required for IL and PDL

The polarimeter of the N7786C is sampling very fast when the stabilizer is active, to provide timely feedback, so the effective averaging time of the readings used for power monitoring are usually shorter than the averaging times chosen for the power meter to achieve low noise levels. Since low noise is equally desirable for the power monitor readings, it is best to read these values multiple times, during the power meter averaging time, and then use the average of those values as m_i^R and m_i^D. To do this, the power meter is configured to wait for a trigger to begin averaging and then after waiting for the averaging time to elapse, the value is uploaded with the FETCH command.

Label	Command	Example	Comment
A4	:STABilizer:SOP	STAB:SOP s_1,s_2,s_3	Enter the target SOP
			wait at least 300ms for this to stabilize
B6	:TRIGger	TRIG 1	trigger the start of the averaging time
A5	:POLarimeter:POWer?	:POL:POW?	Optionally used for power monitoring. This should be repeated over the duration of the power meter averaging time to produce an average monitor power. (If monitoring is not used, A5 can be omitted and B6 and B7 replaced with a READ command that performs both.)
			Wait for any remaining averaging time, including about 10 ms buffer before fetching the power meter reading.
B7	:FETCh[n]:POWer?	FETC1:POW?	Read the measured power

Steps A4 to B7 are repeated for each of the 4 states, initially without a DUT to give the 4 reference power values. If the setup is stable, these values can be used over an extended time, for example measuring multiple devices. Then a device is inserted and the 2 steps are again repeated 4 times. With the measurement results, the desired Stokes parameters to align the polarization are calculated as above.

With FW version 3.0 or higher, instead of waiting for a fixed delay in A4, the command STAB:STAB:DIFF? can be used to poll the current offset from the target SOP. A good response to consider the polarization converged is when it is below “0.005”. This can also be repeated a few times to be sure that the SOP has settled.

Once the procedure is finished, the stabilizer can be deactivated to allow other uses.

Label	Command	Example	Comment
A6	:STABilizer:STABilize	STAB:STAB 0	Deactivate stabilizer function

Appendix: Python Programming Examples

The following pages show two python script programming examples. The first is for programming the all-states algorithm. This version was tested using an N7786C polarization synthesizer and N7745C multiport power meter. The second example shows the Mueller Matrix method including power monitoring with the N7786C. The examples also show use of two forms of interfacing with the instruments. The all-states example uses the VISA COM API. The Mueller Matrix example uses pyvisa.

Script files will also be posted on the Keysight Photonic Discussion Forum.
https://community.keysight.com/community/discussion-forums/photonic

All-states example with VISA-COM

```
# -*- coding: utf-8 -*-
# Python 3.7
"""
Synchronized SOP sequence and power reading, with Max & Min Setting
Instrument control with Keysight VISA COM

"""
import sys
import time
import math
import matplotlib.pyplot as plt

from comtypes.client import GetModule
from comtypes.client import CreateObject

# Run GetModule once to generate comtypes.gen.VisaComLib.

if not hasattr(sys, "frozen"):
 #GetModule("C:\Program Files (x86)\IVI Foundation\VISA\VisaCom\GlobMgr.dll")
 GetModule("C:\Program Files\IVI Foundation\VISA\VisaCom64\GlobMgr.dll")

import comtypes.gen.VisaComLib as VisaComLib

# ============================================================
# Settings:
PMaddr = "TCPIP0::xxx.xxx.xx.xx::5025::SOCKET"
PolCaddr = "TCPIP0::xxx.xxx.xx.xxx::5025::SOCKET"

PMslots = ["1"]
# a list of ports to be measured, as strings
PMrange = "0"
PMavg = "100E-6"
points = "10000"
Wavelength = "1550"
maxrate = 0.001/(max(float(PMavg)+25E-6,float(PMavg)*2))
rate = str(maxrate)
holdoff = str(int(12800/maxrate)) #40% of trigger period
# ============================================================

rm = CreateObject("VISA.GlobalRM", interface=VisaComLib.IResourceManager)
FormattedIOpm = CreateObject("VISA.BasicFormattedIO", \
 interface=VisaComLib.IFormattedIO488)
FormattedIOpm.InstrumentBigEndian = False
FormattedIOpm.IO = rm.Open(PMaddr)
pm = FormattedIOpm.IO
pm.TerminationCharacterEnabled = True
pm.WriteString("*IDN?\n")
answer = pm.ReadString(50)
print (answer)

FormattedIOpc = CreateObject("VISA.BasicFormattedIO", \
 interface=VisaComLib.IFormattedIO488)
```

```
FormattedIOpc.InstrumentBigEndian = False
FormattedIOpc.IO = rm.Open(PolCaddr)
pc = FormattedIOpc.IO
pc.TerminationCharacterEnabled = True
pc.WriteString("*IDN?\n")
answer = pc.ReadString(50)
print (answer)

pc.Timeout = 10000 # 10 seconds
pm.Timeout = 10000 # 10 seconds.
#print ("Timeout set to 10000 milliseconds, for preset delay")
pc.WriteString("SYST:PRES\n")
pm.WriteString("SYST:PRES\n")

print ("What sequence should be used?")
print ("1. random scramble; 2. random walk; 3. last saved")
seqchoice = int(input('Enter corresponding number :'))

pm.WriteString("*OPC?\n")
complete = pm.ReadString(50)
pm.Timeout = 2000 # default timeout

for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":pow:rang:auto 0\n")
    pm.WriteString("sens"+PMslot+":pow:gain:auto 0\n") #use for high PDL
    pm.WriteString("sens"+PMslot+":pow:rang "+PMrange+"dBm\n")
    #pm.WriteString("sens"+PMslot+":pow:rang?\n") #use if ranging needs delay
    #answer = pm.ReadString(50)
    pm.WriteString("sens"+PMslot+":pow:unit 1\n")
    pm.WriteString("trig"+PMslot+":inp SME\n")
    pm.WriteString("sens"+PMslot+":pow:wav "+Wavelength+"nm\n")
    pm.WriteString("sens"+PMslot+":func:par:logg "+points+","+PMavg+"s\n")
#configures all ports before starting logging

pc.WriteString("*OPC?\n")
complete = pc.ReadString(50)
pc.Timeout = 2000 # default timeout

if seqchoice == 1:
    pc.WriteString("pcon:gen:scra? "+points+"\n")
    answer = pc.ReadString(50)
    pc.WriteString("pcon:save:arb?\n")
    answer = pc.ReadString(50)
elif seqchoice == 2:
    pc.WriteString("pcon:gen:rand? "+points+",1000\n")
    answer = pc.ReadString(50)
    pc.WriteString("pcon:save:arb?\n")
    answer = pc.ReadString(50)
elif seqchoice == 3:
    pc.WriteString("pcon:load:arb?\n")
    answer = pc.ReadString(50)
    pc.WriteString("pcon:seq:leng?\n")
    points = pc.ReadString(50)
    for PMslot in PMslots:
        pm.WriteString("sens"+PMslot+":func:par:logg "+points+","+PMavg+"s\n")
else:
    print ("Invalid entry")

pc.WriteString("pcon:seq:dcom 1\n")
pc.WriteString("pcon:rep 1\n")
pc.WriteString("pcon:seq:rrat "+rate+"\n")
pc.WriteString("pcon:seq:hold "+holdoff+"\n")# 40% trigger holdoff
pc.WriteString("pcon:seq:smod 2\n")
pc.WriteString("trig:conf 1\n")
pc.WriteString("pcon:star\n")
pc.WriteString("*OPC?\n")
opc = int(pc.ReadString(50))
firstopc = opc
while opc == 0:
    time.sleep(0.01)
    pc.WriteString("*OPC?\n")
```

```
    opc = int(pc.ReadString(50))

pc.WriteString("TRIG 1\n")
duration = float(points)/maxrate/1000 + 0.01
time.sleep(duration)

input('Connect fiber for reference and then press Enter to continue')

for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":func:stat logg,star\n")
    pm.WriteString("sens"+PMslot+":func:stat?\n")
    loggingStatus = pm.ReadString(50).strip("\n")

pc.WriteString("pcon:star\n")
pc.WriteString("*OPC?\n")
pc.WriteString("TRIG 1\n")
time.sleep(duration)
#triggers logging on all ports

loggingStatus = "PROGRESS"
first = PMslots[0]
while loggingStatus.endswith("PROGRESS"):
    pm.WriteString("sens"+first+":func:stat?\n")
    loggingStatus = pm.ReadString(50).strip("\n")
    time.sleep(0.1)

powerdata = []
for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":func:res?\n")
    powerdata.append(FormattedIOpm.ReadIEEEBlock(VisaComLib.BinaryType_R4,\
                                                 False,True))

refdata = powerdata

input('Connect DUT and then press Enter to continue')

for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":func:stat logg,star\n")
    pm.WriteString("sens"+PMslot+":func:stat?\n")
    loggingStatus = pm.ReadString(50).strip("\n")

pc.WriteString("pcon:star\n")
pc.WriteString("*OPC?\n")
pc.WriteString("TRIG 1\n")
time.sleep(duration)
#triggers logging on all ports
loggingStatus = "PROGRESS"
first = PMslots[0]
while loggingStatus.endswith("PROGRESS"):
    pm.WriteString("sens"+first+":func:stat?\n")
    loggingStatus = pm.ReadString(50).strip("\n")
    time.sleep(0.1)

powerdata = []
for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":func:res?\n")
    powerdata.append(FormattedIOpm.ReadIEEEBlock(VisaComLib.BinaryType_R4,\
                                                 False,True))
dutdata = powerdata

channels = len (powerdata)
arraylength = len(powerdata[0])
transmissiondata = []
for i in range(channels):
    channeldata = []
    for j in range(arraylength):
        channeldata.append(powerdata[i][j]/refdata[i][j])
    transmissiondata.append(channeldata)
PDLdata = []
ILdata = []
for i in range(channels):
```

```
    PDLdata.append(10*math.log10\
                   (abs(max(transmissiondata[i])/min(transmissiondata[i]))))
    ILdata.append(10*math.log10\
                  (abs(max(transmissiondata[i])+min(transmissiondata[i]))/2))

print("PDL (dB),", PDLdata)
print("IL (dB),", ILdata)

#optional additional section to set positions of maximum and minimum signal
pc.WriteString("pcon:stop\n")
pc.WriteString("*OPC?\n")
opc = int(pc.ReadString(50))
while opc == 0:
    time.sleep(0.01)
    pc.WriteString("*OPC?\n")
    opc = int(pc.ReadString(50))

maxindex = transmissiondata[0].index(max(transmissiondata[0]))
minindex = transmissiondata[0].index(min(transmissiondata[0]))
maxlength = str(maxindex + 1)
minlength = str(minindex + 1)

pc.WriteString("pcon:seq:leng "+maxlength+"\n")

pc.WriteString("pcon:star\n")
pc.WriteString("*OPC?\n")
opc = int(pc.ReadString(50))
while opc == 0:
    time.sleep(0.01)
    pc.WriteString("*OPC?\n")
    opc = int(pc.ReadString(50))

pc.WriteString("TRIG 1\n")
duration2 = float(maxlength)/maxrate/1000 + 0.1
time.sleep(duration2)

pm.WriteString("read"+first+":pow?\n")
maxpower = pm.ReadString(50).strip("\n")


pc.WriteString("pcon:stop\n")
pc.WriteString("*OPC?\n")
opc = int(pc.ReadString(50))
while opc == 0:
    time.sleep(0.01)
    pc.WriteString("*OPC?\n")
    opc = int(pc.ReadString(50))

pc.WriteString("pcon:seq:leng "+minlength+"\n")

pc.WriteString("pcon:star\n")
pc.WriteString("*OPC?\n")
opc = int(pc.ReadString(50))
while opc == 0:
    time.sleep(0.01)
    pc.WriteString("*OPC?\n")
    opc = int(pc.ReadString(50))

pc.WriteString("TRIG 1\n")
duration2 = float(minlength)/maxrate/1000 + 0.1
time.sleep(duration2)

pm.WriteString("read"+first+":pow?\n")
minpower = pm.ReadString(50).strip("\n")

print("Maximum power (W),", maxpower)
print("Minimum power(W)),", minpower)
#end of optional alignment section

#make plot
t = []
```

```
for i in range(arraylength):
    t.append(i)
for i in range(channels):
    plt.plot(t, refdata[i])
    plt.plot(t, dutdata[i])

plt.xlabel('Sample')
plt.ylabel('power (W)')
plt.title('DUT and Ref traces')
plt.grid(True)
plt.savefig("PDLmethod.png")
plt.show()
plt.close()

for PMslot in PMslots:
    pm.WriteString("sens"+PMslot+":func:stat logg,stop\n")
pc.WriteString("pcon:stop\n")

pm.Close()
pc.Close()
```

Mueller Matrix example with pyvisa

```
# -*- coding: utf-8 -*-
# Python 3.7

"""
Mueller Matrix PDL with N7786C, using power monitoring

"""
import time
import math
import pyvisa

# ============================================================
# Settings:
PMaddr = "TCPIP0::xxx.xxx.xx.xx::5025::SOCKET"
PolCaddr = "TCPIP0::xxx.xxx.xx.xxx::5025::SOCKET"

PMslots = ["1"]
first = PMslots[0]
# a list of ports to be measured, as strings
refPM = "1"
PMrange = "0"
PMavg = "200E-3"
avgtime = float(PMavg)

Wavelength = "1598.7"

StabDelay = 2
SopDelay = 0.3
SopList = ["1,0,0","-1,0,0","0,1,0","0,0,1"]

# ============================================================
rm = pyvisa.ResourceManager()

pc = rm.open_resource(PolCaddr)
pc.read_termination = '\n'
pc.write_termination = '\n'
answer = pc.query("*IDN?")
print (answer)

pm = rm.open_resource(PMaddr)
pm.read_termination = '\n'
pm.write_termination = '\n'
answer = pm.query("*IDN?")
print (answer)

pc.timeout = 10000 # 10 seconds
pm.timeout = 10000
#print ("Timeout set to 10000 milliseconds, for preset delay")

pc.write("SYST:PRES")
pm.write("SYST:PRES")

pm.write("*OPC?")
pmopc = int(pm.read())
pmfirstwopc = pmopc
while pmopc == 0:
    time.sleep(0.01)
    pm.write("*OPC?")
    pmopc = int(pm.read())

pm.timeout = 2000
#print ("Timeout reset to default 2000 milliseconds")

PortCount = pm.query("fetch:pow:all:csv?").count(',') + 1

for i in range(PortCount):
    PMslot = str(i+1)
```

```
    pm.write("sens"+PMslot+":pow:rang:auto 0")
    #pm.write("sens"+PMslot+":pow:gain:auto 0")
    pm.write("sens"+PMslot+":pow:rang "+PMrange+"dBm")
    #pm.write("sens"+PMslot+":pow:rang?")#can be used for ranging delay
    #answer = pm.read()
    #print (answer)
    pm.write("sens"+PMslot+":pow:unit 1")
    pm.write("sens"+PMslot+":pow:atim "+PMavg+"s")
    pm.write("init"+PMslot+":cont 0")
    pm.write("sens"+PMslot+":pow:wav "+Wavelength+"nm")
#configures all ports before starting logging

pc.write("*OPC?")
polopc = int(pc.read())
polfirstwopc = polopc #this usually starts with 0
while polopc == 0:
    time.sleep(0.01)
    pc.write("*OPC?")
    polopc = int(pc.read())

pc.timeout = 2000 # 2 seconds

pc.write("pol:gain 2") #lower signal level may need higher gain setting
pc.write("pol:wav "+Wavelength+"nm")
pc.write("stab:stab 1")
time.sleep(StabDelay)

input('Connect fiber for reference and then press Enter to continue')

Ref = []
index = 0
for sop in SopList:
    pc.write("stab:sop "+sop)
    time.sleep(SopDelay)
    pm.write("TRIG 1")
    t1 = time.perf_counter()
    t2 = time.perf_counter()
    PolPowSet = []
    while t2-t1 < avgtime + 0.01:
        pc.write("POL:POW?")
        PolPowSet.append(float(pc.read()))
        t2 = time.perf_counter()
    pm.write("fetc"+refPM+":pow?")
    avgPolPow = sum(PolPowSet)/len(PolPowSet)
    Ref.append(float(pm.read())/avgPolPow)
    index = index + 1

input('Connect DUT and then press Enter to continue')

Repeat = 1
while Repeat == 1:

    Pdut = []
    index = 0
    for sop in SopList:
        pc.write("stab:sop "+sop)
        time.sleep(SopDelay)
        pm.write("TRIG 1")
        t1 = time.perf_counter()
        t2 = time.perf_counter()
        PolPowSet = []
        while t2-t1 < avgtime + 0.01:
            pc.write("POL:POW?")
            PolPowSet.append(float(pc.read()))
            t2 = time.perf_counter()
        pm.write("fetc"+first+":pow?")
        avgPolPow = sum(PolPowSet)/len(PolPowSet)
        Pdut.append(float(pm.read())/avgPolPow)
        index = index + 1

    M1 = 0.5 * (Pdut[0]/Ref[0] + Pdut[1]/Ref[1])
```

```
    M2 = 0.5 * (Pdut[0]/Ref[0] - Pdut[1]/Ref[1])
    M3 = Pdut[2]/Ref[2] - M1
    M4 = Pdut[3]/Ref[3] - M1
    d = math.sqrt(M2*M2 + M3*M3 +M4*M4)
    Tmax = M1 + d
    Tmin = M1 - d

    PDL = 10*math.log10(abs(Tmax/Tmin))
    IL = -10*math.log10(M1)
    print("PDL from matrix calculation,", PDL)
    print("IL from matrix calculation,", IL)

    print ("1. repeat DUT measurement; 0. exit;")
    Repeat = int(input('Enter corresponding number :'))

pc.write("stab:stab 0")
time.sleep(SopDelay)

pm.close()
pc.close()
```

Conclusion

The synchronization of fast polarization sequences with the optical power meter sampling allows very fast all-states measurements, both for determining high polarization extinction levels and for measuring PDL values lower than the polarization dependence of the instrument. The real-time stabilization and monitoring of the N7786C provide a very accurate and reliable implementation of the Mueller Matrix method. These capabilities also provide the basis for functionality and algorithms to align the polarization with the axes of a device, as for probing of photonic integrated circuits (PIC). This is the subject of another application note.

https://www.keysight.com/us/en/assets/3120-1564

www.keysight.com/find/pol

Learn more at: www.keysight.com

For more information on Keysight Technologies' products, applications, or services, please contact your local Keysight office. The complete list is available at: www.keysight.com/find/contactus

KEYSIGHT TECHNOLOGIES