solarPy: A Python Library and GUI for Solar Cell Radiation Degradation Modeling

Don Walker, Misha Dowd, and Simon H. Liu

The Aerospace Corporation, El Segundo, CA, 90601, USA

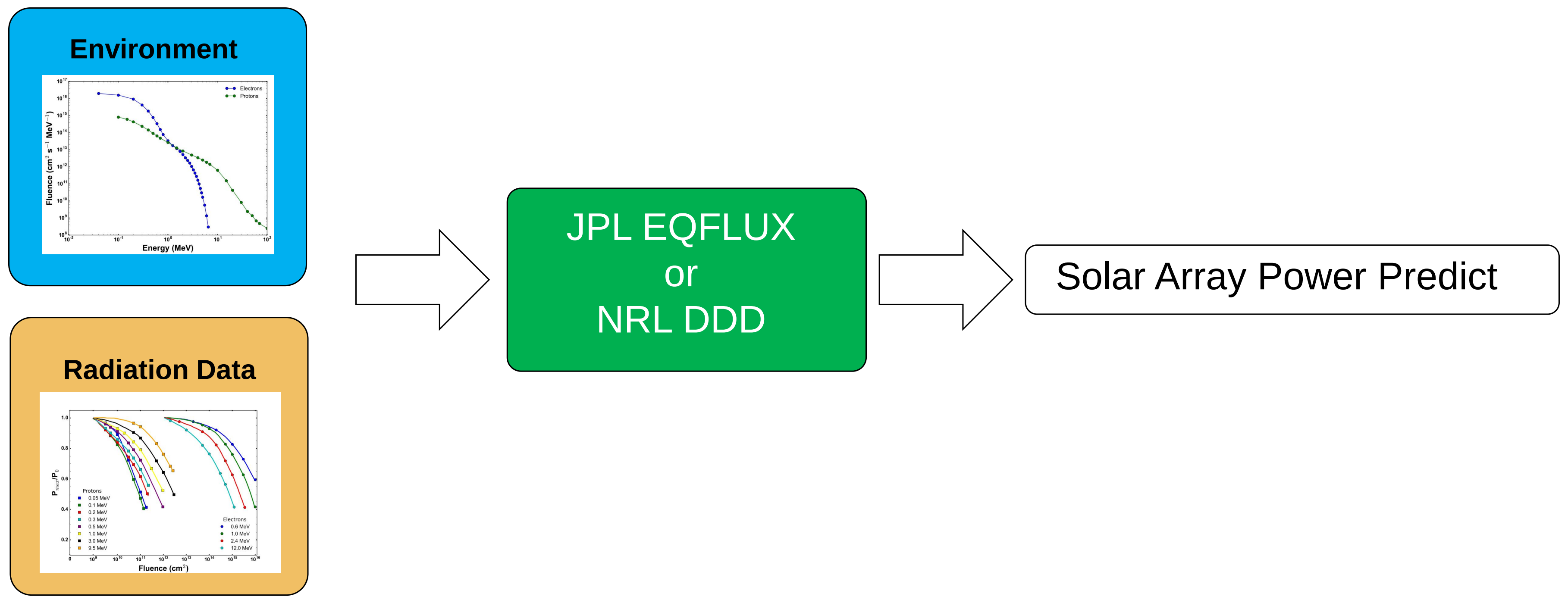
Introduction

Solar cell degradation from radiation in the space environment is one of the largest contributors to loss of solar array performance. Currently, there are two methods to perform solar cell degradation predictions:

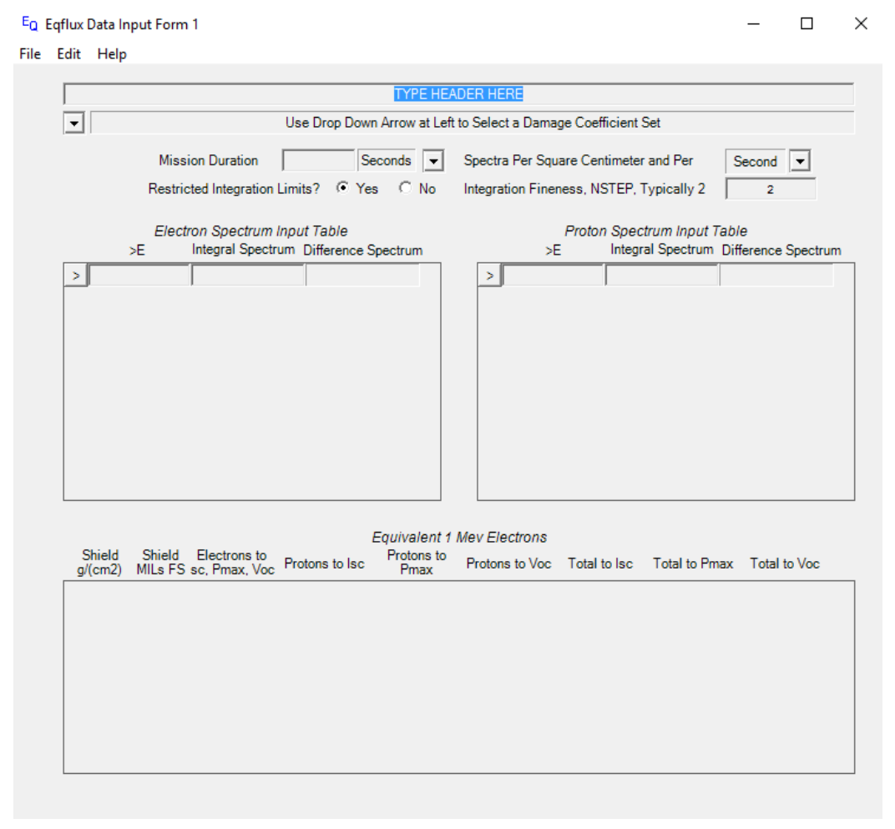
- The empirical NASA JPL Equivalent Flux (EQFLUX) method
- Semi-empirical NRL Displacement Damage Dose (NRL DDD) method

Goal: Compare the similarities and differences between the two methods from the ground up, and identify areas for improvement for the two methods that could bring the two methods closer in agreement.

- Impact
  - *Potential to improve radiation modeling methods*
  - *Various radiation degradation methods can be compared side-by-side*
  - *New methods and models can easily be incorporated*
  - *Pros for various methods can be combined*
  - *Everyone can perform the same calculation*



EQFLUX



TYPE HEADER HERE

Use Drop Down Arrow at Left to Select a Damage Coefficient Set

Mission Duration: Seconds

Spectra Per Square Centimeter and Per: Second

Restricted Integration Limits? ● Yes ○ No

Integration Fineness, NSTEP, Typically 2: 2

Electron Spectrum Input Table

>E	Integral Spectrum	Difference Spectrum

Proton Spectrum Input Table

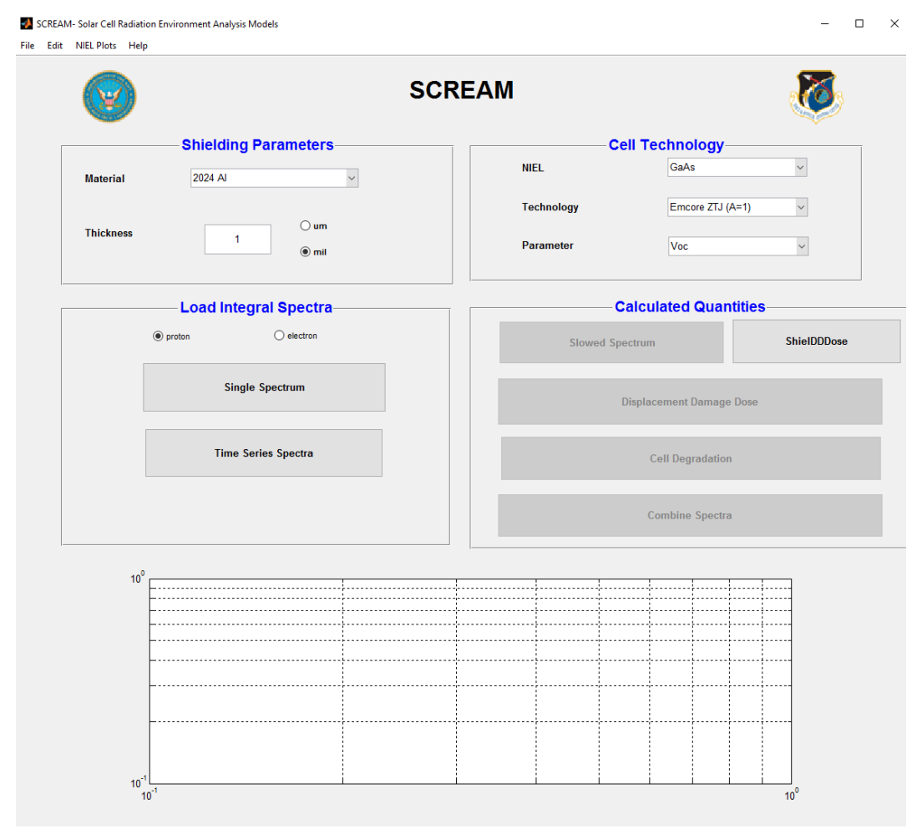
>E	Integral Spectrum	Difference Spectrum

Equivalent 1 Mev Electrons

Shield g/(cm2)	Shield MILs FS	Electrons to sc, Pmax, Voc	Protons to Isc	Protons to Pmax	Protons to Voc	Total to Isc	Total to Pmax	Total to Voc

JPL EqFlux program available on JPL's website

NRL SCREAM



Shielding Parameters

Material: 2024 Al

Thickness: 1 ○ um ● mil

Cell Technology

NIEL: GaAs

Technology: Emcore ZTJ (A=1)

Parameter: Voc

Load Integral Spectra

● proton ○ electron

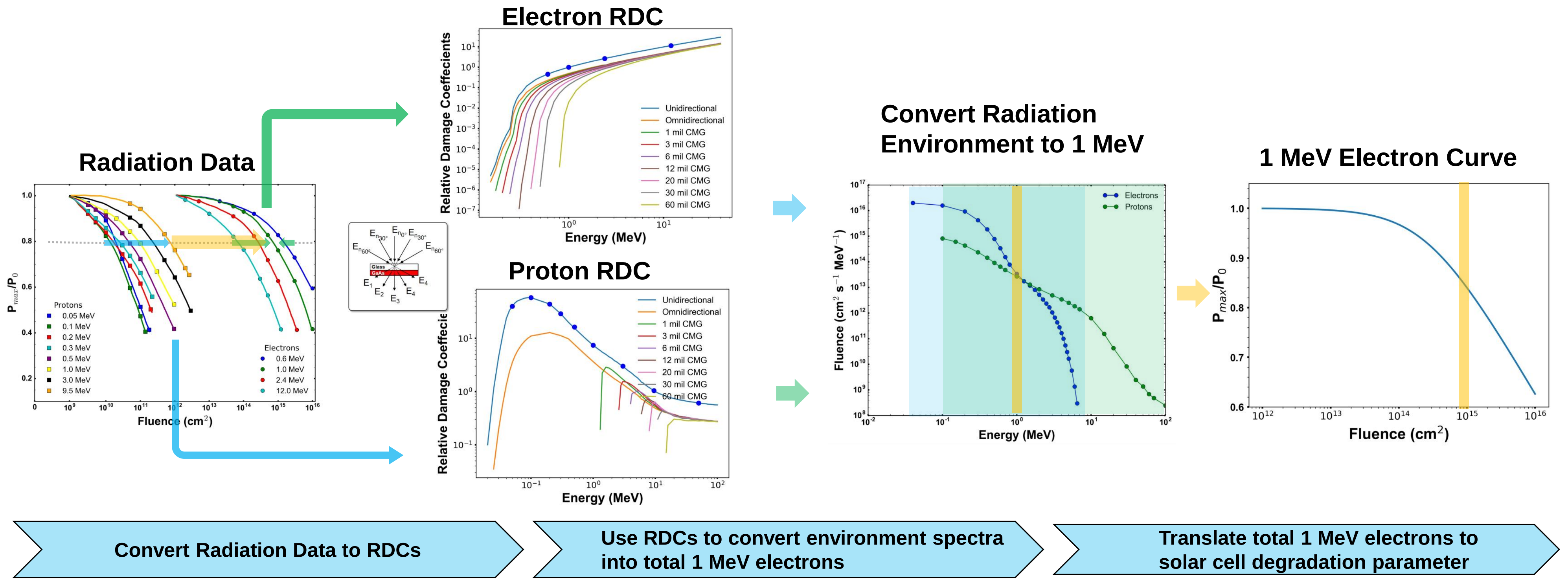
Calculated Quantities

Older version of NRL Scream available from NRL

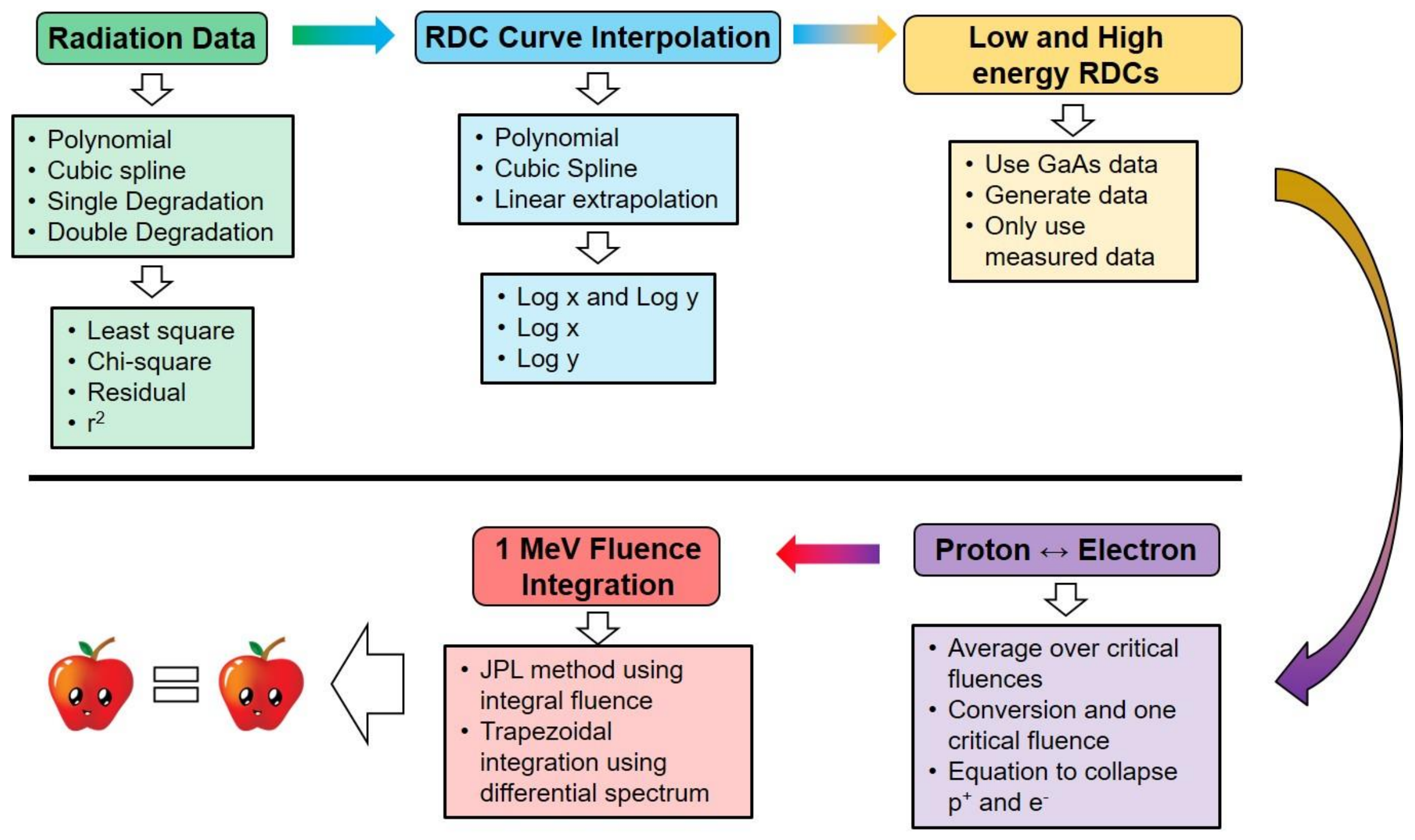
JPL EQFLUX

Relates all fluences of all particle energies to a single1 MeV electron fluence equivalent

1. *Relate damage of protons and electrons to one common particle energy and fluence*
   a. Calculate Relative Damage Coefficients (RDC)
   - *Find the relative fluence needed at each particle energy and relates that fluence to the fluence at one common particle energy that is at the same level of degradation*
   - *Calculate RDCs for both protons and electrons*
   - *Find one common factor/coefficient that relates the protons to electrons*
2. *Convert a proton particle spectrum to one proton energy and one fluence using the proton RDC curves*
3. *Repeat step 2 for electrons*
4. *Add the fluence calculated from 2 and 3 together to get the total 1 MeV Electron Fluence*
5. *Go to the 1 MeV Electron Remaining Factor vs Fluence and then determine the remaining factor at the fluence calculated*



Convert Radiation Data to RDCs → Use RDCs to convert environment spectra into total 1 MeV electrons → Translate total 1 MeV electrons to solar cell degradation parameter



Radiation Data → **RDC Curve Interpolation** → **Low and High energy RDCs**

Radiation Data:
- Polynomial
- Cubic spline
- Single Degradation
- Double Degradation

then:
- Least square
- Chi-square
- Residual
- r^2

RDC Curve Interpolation:
- Polynomial
- Cubic Spline
- Linear extrapolation

then:
- Log x and Log y
- Log x
- Log y

Low and High energy RDCs:
- Use GaAs data
- Generate data
- Only use measured data

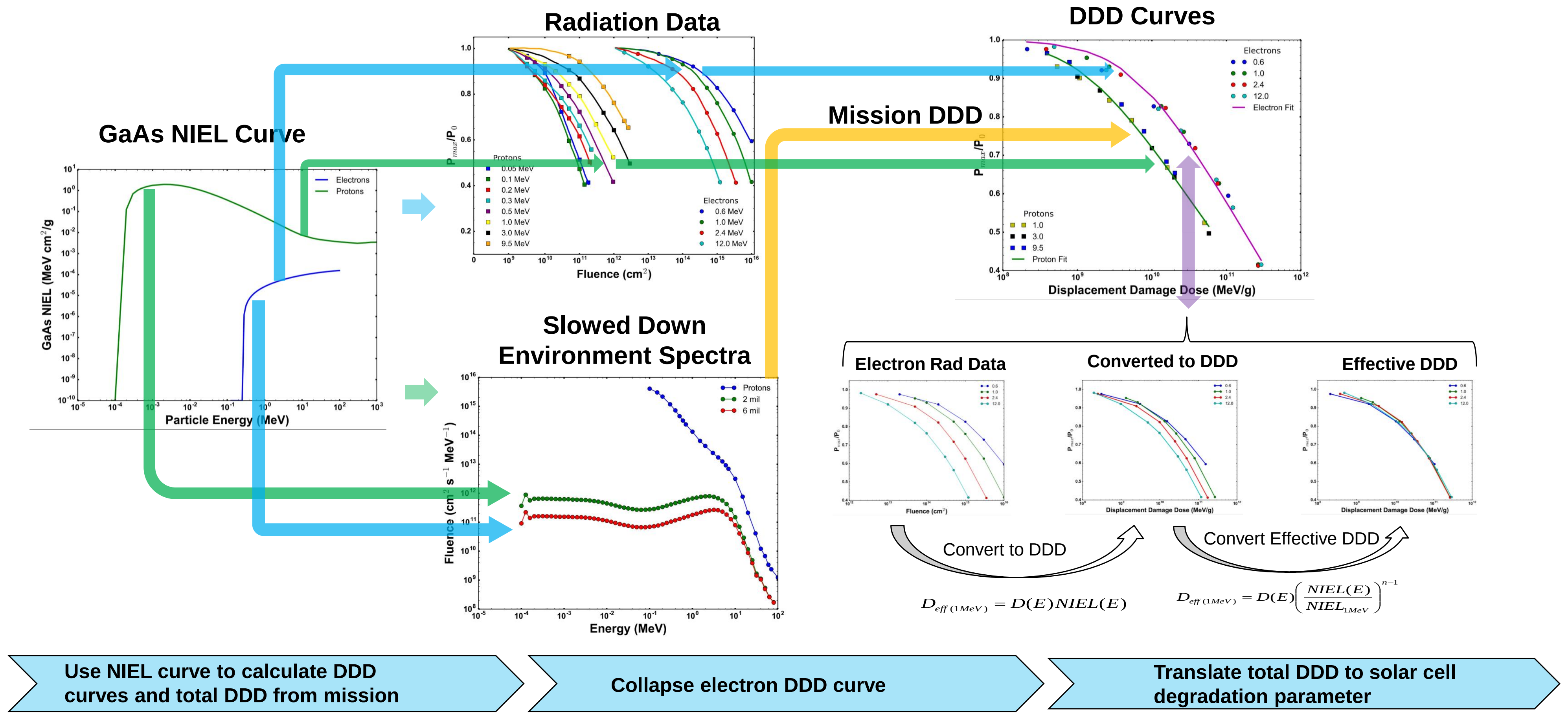
Proton ↔ Electron:
- Average over critical fluences
- Conversion and one critical fluence
- Equation to collapse p^+ and e^-

1 MeV Fluence Integration:
- JPL method using integral fluence
- Trapezoidal integration using differential spectrum

NRL DDD

Relates all fluences of all particle energies to a total Displacement Damage Dose (DDD). By utilizing theoretical data to calculate the Non-Ionizing Energy Loss (NIEL) for a solar cell material, a total displacement damage dose can be calculated.

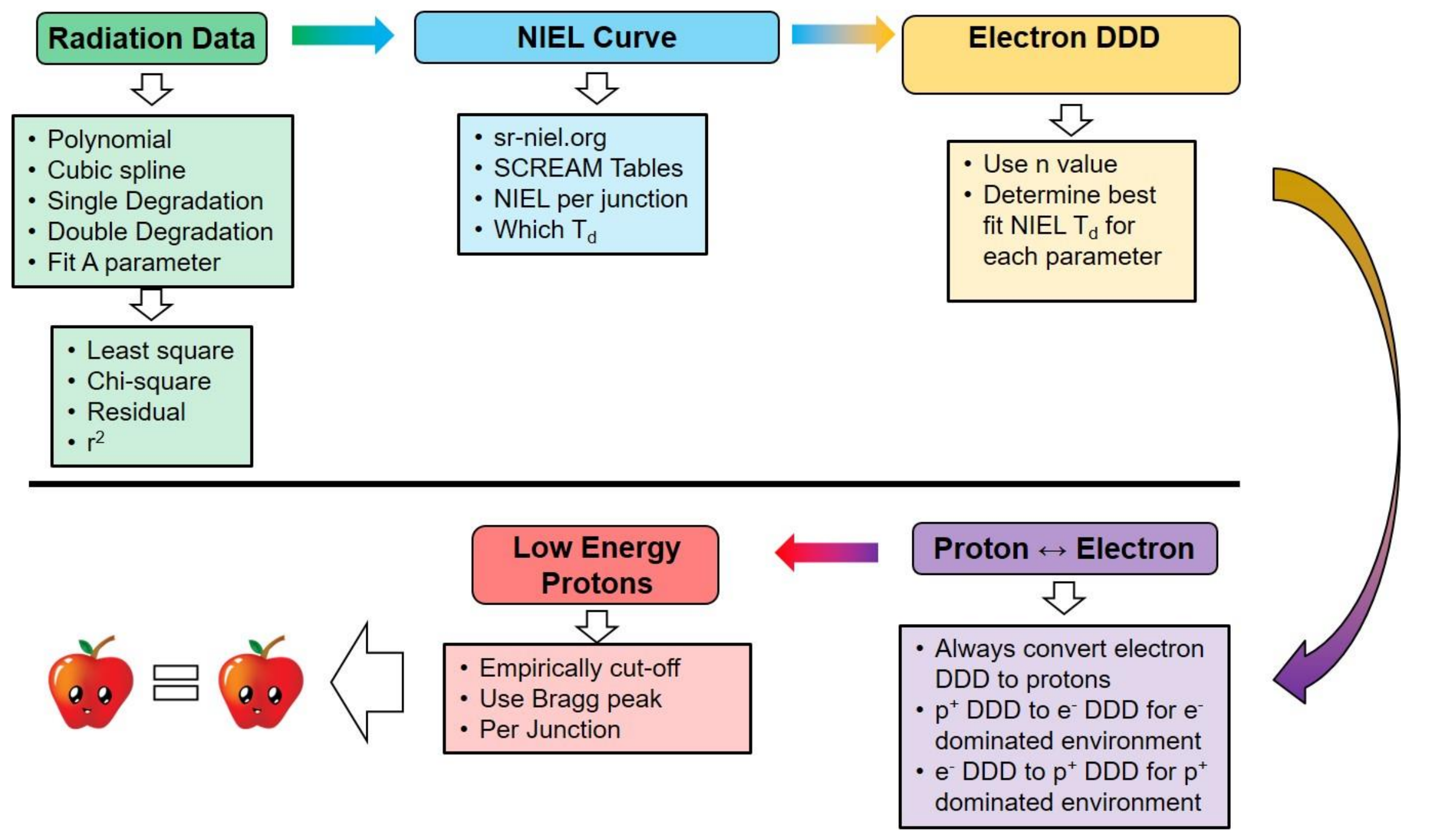
1. *Convert the fluences at all proton energies to DDD, which generates a single characteristic curve using the NIEL curve*
   - Only necessary to use one proton energy to scale the curve
2. *Convert all fluences at all electron energies to an "effective" DDD, which generates a single characteristic curve using the NIEL curve*
   - Using the NIEL curve with electron data does not fully collapsed curve
   - An empirically derived correction factor is needed to generate a single characteristic curve for all energies; therefore, at least 2 electron energies are needed
3. *Convert the single characteristic ELECTRON curve to a single characteristic PROTON curve using another coefficient.*
4. *Generate Slowed Down Spectrum*
   - Using transport models, the space particle radiation environment spectrum is determined after it passes through cover glass
5. *Multiply the proton NIEL curve by the proton slowed down spectrum to calculate total displacement damage dose for the proton environment*
6. *Repeat step 5 for electrons*
7. *Convert electron DDD to proton DDD and sum the two*
8. *Go to the single characteristic PROTON curve (Remaining Factor vs Dose) and then determine the remaining factor at the dose calculated*



$$D_{eff\,(1MeV)} = D(E)NIEL(E)$$

$$D_{eff\,(1MeV)} = D(E)\left(\frac{NIEL(E)}{NIEL_{1MeV}}\right)^{n-1}$$

Use NIEL curve to calculate DDD curves and total DDD from mission → Collapse electron DDD curve → Translate total DDD to solar cell degradation parameter



Radiation Data → **NIEL Curve** → **Electron DDD**

Radiation Data:
- Polynomial
- Cubic spline
- Single Degradation
- Double Degradation
- Fit A parameter

then:
- Least square
- Chi-square
- Residual
- r^2

NIEL Curve:
- sr-niel.org
- SCREAM Tables
- NIEL per junction
- Which T_d

Electron DDD:
- Use n value
- Determine best fit NIEL T_d for each parameter

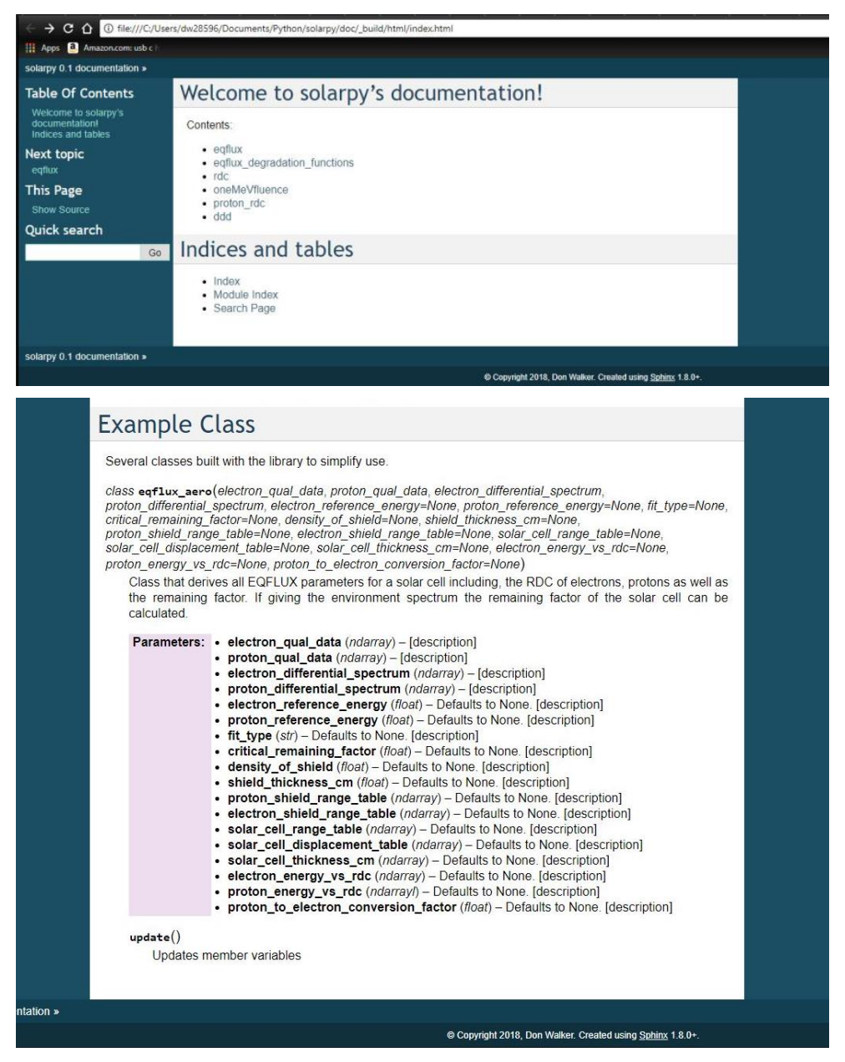
Proton ↔ Electron:
- Always convert electron DDD to protons
- p^+ DDD to e^- DDD for e^- dominated environment
- e^- DDD to p^+ DDD for p^+ dominated environment

Low Energy Protons:
- Empirically cut-off
- Use Bragg peak
- Per Junction

solarPy

- Community supported solar cell radiation degradation analysis code built from the ground up
- Built using Python, which is free, easy to read, and learn
- Incorporate all methods of radiation degradation modeling such as EQFLUX and NRL DDD in one common language and source code
- Provide apples-to-apples comparison for predicting solar cell and array performance for various missions at all levels of the calculation
- Enables comprehensive comparisons between NRL Displacement Damage Dose method and the JPL Equivalent Flux Method
- GUI developed using web application technologies which enables it to be deployed to any modern OS.
- Python, windows, and unix command line interface programs are included
- Python library is fully documented with examples

Documentation



file:///C:/Users/dw28596/Documents/Python/solarpy/doc/_build/html/index.html

solarpy 0.1 documentation »

Table Of Contents: Welcome to solarpy's documentation! · Indices and tables

Next topic: eqflux

This Page: Show Source

Quick search [Go]

Welcome to solarpy's documentation!

Contents:
- eqflux
- eqflux_degradation_functions
- rdc
- oneMeVfluence
- proton_rdc
- ddd

Indices and tables
- Index
- Module Index
- Search Page

© Copyright 2018, Don Walker. Created using Sphinx 1.8.0+.

Example Class

Several classes built with the library to simplify use.

class **eqflux_aero**(*electron_qual_data, proton_qual_data, electron_differential_spectrum, proton_differential_spectrum, electron_reference_energy=None, proton_reference_energy=None, fit_type=None, critical_remaining_factor=None, density_of_shield=None, shield_thickness_cm=None, proton_shield_range_table=None, electron_shield_range_table=None, solar_cell_range_table=None, solar_cell_displacement_table=None, solar_cell_thickness_cm=None, electron_energy_vs_rdc=None, proton_energy_vs_rdc=None, proton_to_electron_conversion_factor=None*)

Class that derives all EQFLUX parameters for a solar cell including, the RDC of electrons, protons as well as the remaining factor. If giving the environment spectrum the remaining factor of the solar cell can be calculated.

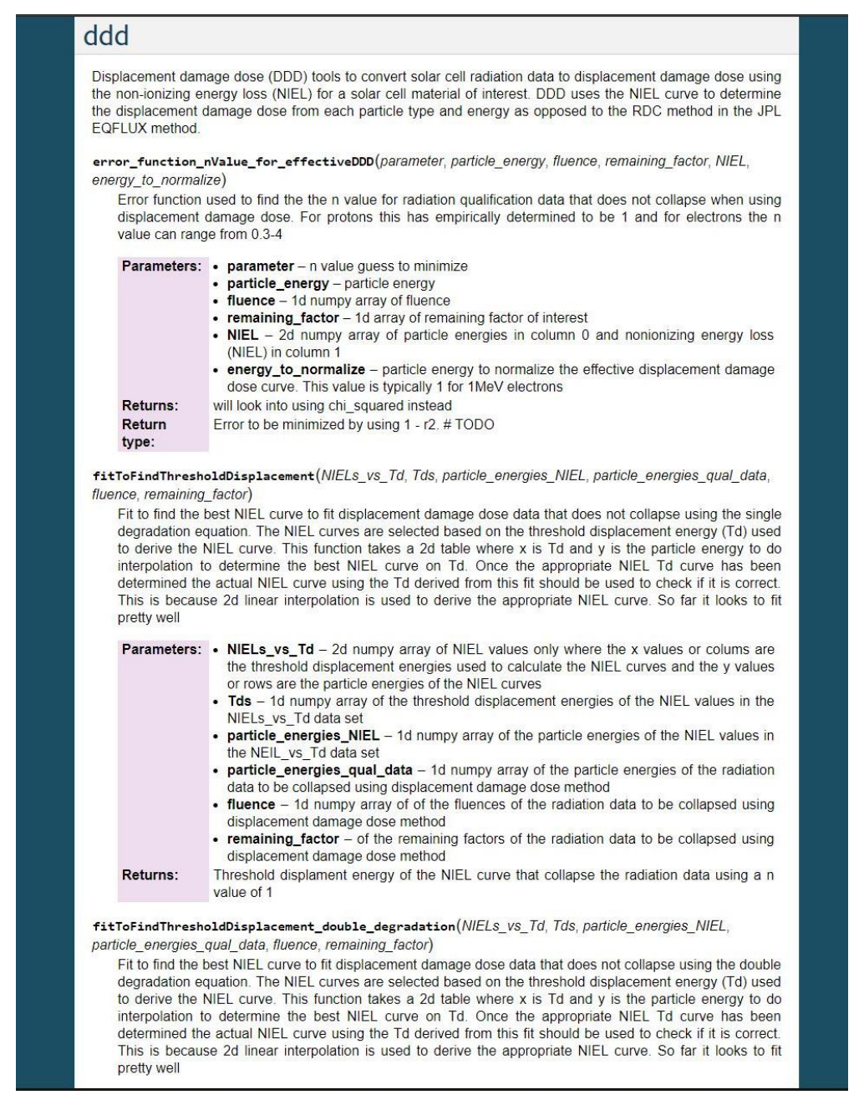
Parameters:
- **electron_qual_data** (*ndarray*) – [description]
- **proton_qual_data** (*ndarray*) – [description]
- **electron_differential_spectrum** (*ndarray*) – [description]
- **proton_differential_spectrum** (*ndarray*) – [description]
- **electron_reference_energy** (*float*) – Defaults to None. [description]
- **proton_reference_energy** (*float*) – Defaults to None. [description]
- **fit_type** (*str*) – Defaults to None. [description]
- **critical_remaining_factor** (*float*) – Defaults to None. [description]
- **density_of_shield** (*float*) – Defaults to None. [description]
- **shield_thickness_cm** (*float*) – Defaults to None. [description]
- **proton_shield_range_table** (*ndarray*) – Defaults to None. [description]
- **electron_shield_range_table** (*ndarray*) – Defaults to None. [description]
- **solar_cell_range_table** (*ndarray*) – Defaults to None. [description]
- **solar_cell_displacement_table** (*ndarray*) – Defaults to None. [description]
- **solar_cell_thickness_cm** (*ndarray*) – Defaults to None. [description]
- **electron_energy_vs_rdc** (*ndarray*) – Defaults to None. [description]
- **proton_energy_vs_rdc** (*ndarrayl*) – Defaults to None. [description]
- **proton_to_electron_conversion_factor** (*float*) – Defaults to None. [description]

update()

Updates member variables

© Copyright 2018, Don Walker. Created using Sphinx 1.8.0+.

DDD



ddd

Displacement damage dose (DDD) tools to convert solar cell radiation data to displacement damage dose using the non-ionizing energy loss (NIEL) for a solar cell material of interest. DDD uses the NIEL curve to determine the displacement damage dose from each particle type and energy as opposed to the RDC method in the JPL EQFLUX method.

error_function_nValue_for_effectiveDDD(*parameter, particle_energy, fluence, remaining_factor, NIEL, energy_to_normalize*)

Error function used to find the the n value for radiation qualification data that does not collapse when using displacement damage dose. For protons this has empirically determined to be 1 and for electrons the n value can range from 0.3-4

Parameters:
- **parameter** – n value guess to minimize
- **particle_energy** – particle energy
- **fluence** – 1d numpy array of fluence
- **remaining_factor** – 1d array of remaining factor of interest
- **NIEL** – 2d numpy array of particle energies in column 0 and nonionizing energy loss (NIEL) in column 1
- **energy_to_normalize** – particle energy to normalize the effective displacement damage dose curve. This value is typically 1 for 1MeV electrons

Returns: will look into using chi_squared instead

Return type: Error to be minimized by using 1 - r2. # TODO

fitToFindThresholdDisplacement(*NIELs_vs_Td, Tds, particle_energies_NIEL, particle_energies_qual_data, fluence, remaining_factor*)

Fit to find the best NIEL curve to fit displacement damage dose data that does not collapse using the single degradation equation. The NIEL curves are selected based on the threshold displacement energy (Td) used to derive the NIEL curve. This function takes a 2d table where x is Td and y is the particle energy to do interpolation to determine the best NIEL curve on Td. Once the appropriate NIEL Td curve has been determined the actual NIEL curve using the Td derived from this fit should be used to check if it is correct. This is because 2d linear interpolation is used to derive the appropriate NIEL curve. So far it looks to fit pretty well

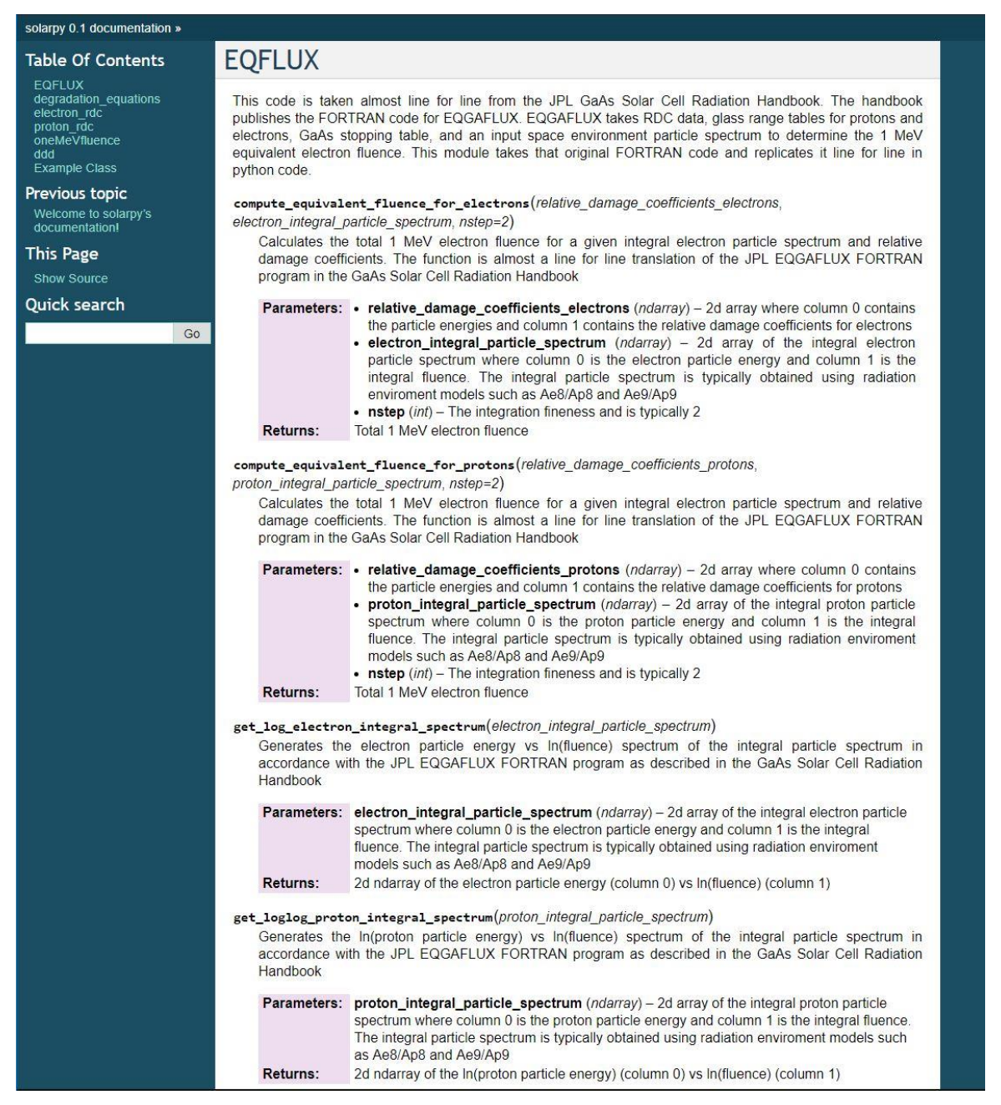
Parameters:
- **NIELs_vs_Td** – 2d numpy array of NIEL values only where the x values or colums are the threshold displacement energies used to calculate the NIEL curves and the y values or rows are the particle energies of the NIEL curves
- **Tds** – 1d numpy array of the threshold displacement energies of the NIEL values in the NIELs_vs_Td data set
- **particle_energies_NIEL** – 1d numpy array of the particle energies of the NIEL values in the NEIL_vs_Td data set
- **particle_energies_qual_data** – 1d numpy array of the particle energies of the radiation data to be collapsed using displacement damage dose method
- **fluence** – 1d numpy array of of the fluences of the radiation data to be collapsed using displacement damage dose method
- **remaining_factor** – of the remaining factors of the radiation data to be collapsed using displacement damage dose method

Returns: Threshold displament energy of the NIEL curve that collapse the radiation data using a n value of 1

fitToFindThresholdDisplacement_double_degradation(*NIELs_vs_Td, Tds, particle_energies_NIEL, particle_energies_qual_data, fluence, remaining_factor*)

Fit to find the best NIEL curve to fit displacement damage dose data that does not collapse using the double degradation equation. The NIEL curves are selected based on the threshold displacement energy (Td) used to derive the NIEL curve. This function takes a 2d table where x is Td and y is the particle energy to do interpolation to determine the best NIEL curve on Td. Once the appropriate NIEL Td curve has been determined the actual NIEL curve using the Td derived from this fit should be used to check if it is correct. This is because 2d linear interpolation is used to derive the appropriate NIEL curve. So far it looks to fit pretty well

EQFLUX



solarpy 0.1 documentation »

Table Of Contents: EQFLUX · degradation_equations · electron_rdc · proton_rdc · oneMeVfluence · ddd · Example Class

Previous topic: Welcome to solarpy's documentation!

This Page: Show Source

Quick search [Go]

EQFLUX

This code is taken almost line for line from the JPL GaAs Solar Cell Radiation Handbook. The handbook publishes the FORTRAN code for EQGAFLUX. EQGAFLUX takes RDC data, glass range tables for protons and electrons, GaAs stopping table, and an input space environment particle spectrum to determine the 1 MeV equivalent electron fluence. This module takes that original FORTRAN code and replicates it line for line in python code.

compute_equivalent_fluence_for_electrons(*relative_damage_coefficients_electrons, electron_integral_particle_spectrum, nstep=2*)

Calculates the total 1 MeV electron fluence for a given integral electron particle spectrum and relative damage coefficients. The function is almost a line for line translation of the JPL EQGAFLUX FORTRAN program in the GaAs Solar Cell Radiation Handbook

Parameters:
- **relative_damage_coefficients_electrons** (*ndarray*) – 2d array where column 0 contains the particle energies and column 1 contains the relative damage coefficients for electrons
- **electron_integral_particle_spectrum** (*ndarray*) – 2d array of the integral electron particle spectrum where column 0 is the electron particle energy and column 1 is the integral fluence. The integral particle spectrum is typically obtained using radiation enviroment models such as Ae8/Ap8 and Ae9/Ap9
- **nstep** (*int*) – The integration fineness and is typically 2

Returns: Total 1 MeV electron fluence

compute_equivalent_fluence_for_protons(*relative_damage_coefficients_protons, proton_integral_particle_spectrum, nstep=2*)

Calculates the total 1 MeV electron fluence for a given integral electron particle spectrum and relative damage coefficients. The function is almost a line for line translation of the JPL EQGAFLUX FORTRAN program in the GaAs Solar Cell Radiation Handbook

Parameters:
- **relative_damage_coefficients_protons** (*ndarray*) – 2d array where column 0 contains the particle energies and column 1 contains the relative damage coefficients for protons
- **proton_integral_particle_spectrum** (*ndarray*) – 2d array of the integral proton particle spectrum where column 0 is the proton particle energy and column 1 is the integral fluence. The integral particle spectrum is typically obtained using radiation enviroment models such as Ae8/Ap8 and Ae9/Ap9
- **nstep** (*int*) – The integration fineness and is typically 2

Returns: Total 1 MeV electron fluence

get_log_electron_integral_spectrum(*electron_integral_particle_spectrum*)

Generates the electron particle energy vs ln(fluence) spectrum of the integral particle spectrum in accordance with the JPL EQGAFLUX FORTRAN program as described in the GaAs Solar Cell Radiation Handbook

Parameters: **electron_integral_particle_spectrum** (*ndarray*) – 2d array of the integral electron particle spectrum where column 0 is the electron particle energy and column 1 is the integral fluence. The integral particle spectrum is typically obtained using radiation enviroment models such as Ae8/Ap8 and Ae9/Ap9

Returns: 2d ndarray of the electron particle energy (column 0) vs ln(fluence) (column 1)

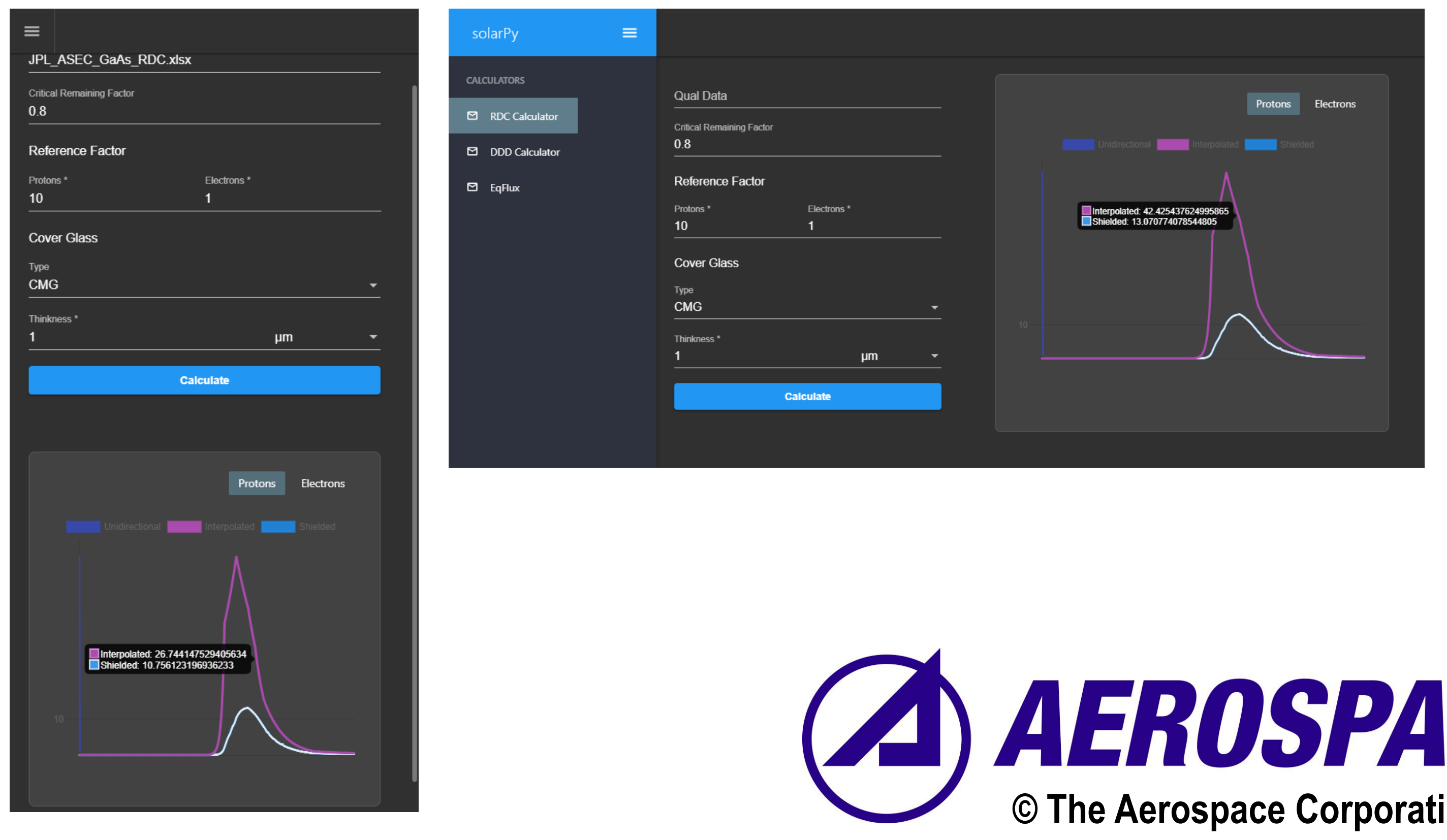
get_loglog_proton_integral_spectrum(*proton_integral_particle_spectrum*)

Generates the ln(proton particle energy) vs ln(fluence) spectrum of the integral particle spectrum in accordance with the JPL EQGAFLUX FORTRAN program as described in the GaAs Solar Cell Radiation Handbook

Parameters: **proton_integral_particle_spectrum** (*ndarray*) – 2d array of the integral proton particle spectrum where column 0 is the proton particle energy and column 1 is the integral fluence. The integral particle spectrum is typically obtained using radiation enviroment models such as Ae8/Ap8 and Ae9/Ap9

Returns: 2d ndarray of the ln(proton particle energy) (column 0) vs ln(fluence) (column 1)

solarPy GUI



JPL_ASEC_GaAs_RDC.xlsx

Qual Data

Critical Remaining Factor: 0.8

Reference Factor

Protons *: 10

Electrons *: 1

Cover Glass

Type: CMG

Thinkness *: 1 µm

Interpolated: 26.744147529405634

Shielded: 10.756123196936233

Interpolated: 42.425437624995865

Shielded: 13.070774078544805

Acknowledgements:

We would like to acknowledge and thank The Aerospace Corporation's Independent Research & Development Funding