PROBLEM SET #3

TO: PROF. DAVID MILLER, PROF. JOHN KEESEE, AND MS. MARILYN GOOD

FROM: NAMES WITHHELD

SUBJECT: PROBLEM SET #3 (SPACE ENVIRONMENT AND POWER SUBSYSTEMS)

DATE: 12/11/2003

MOTIVATION

The Engineering Test Satellite VI (ETS-VI) was launched by the H-II Launch Vehicle Flight No. 2 in August 1994, but it was not placed into the prescribed geostationary orbit because of an apogee engine trouble. Later, the satellite was placed into a geostationary-transfer orbit across the Van Allen Belt made up of high energy protons and electrons trapped along field lines of the Earth. As a result, the ETS-VI had been exposed to a huge amount of space radiation and the solar cell decreased power output and the satellite ceased to operate in 1996.

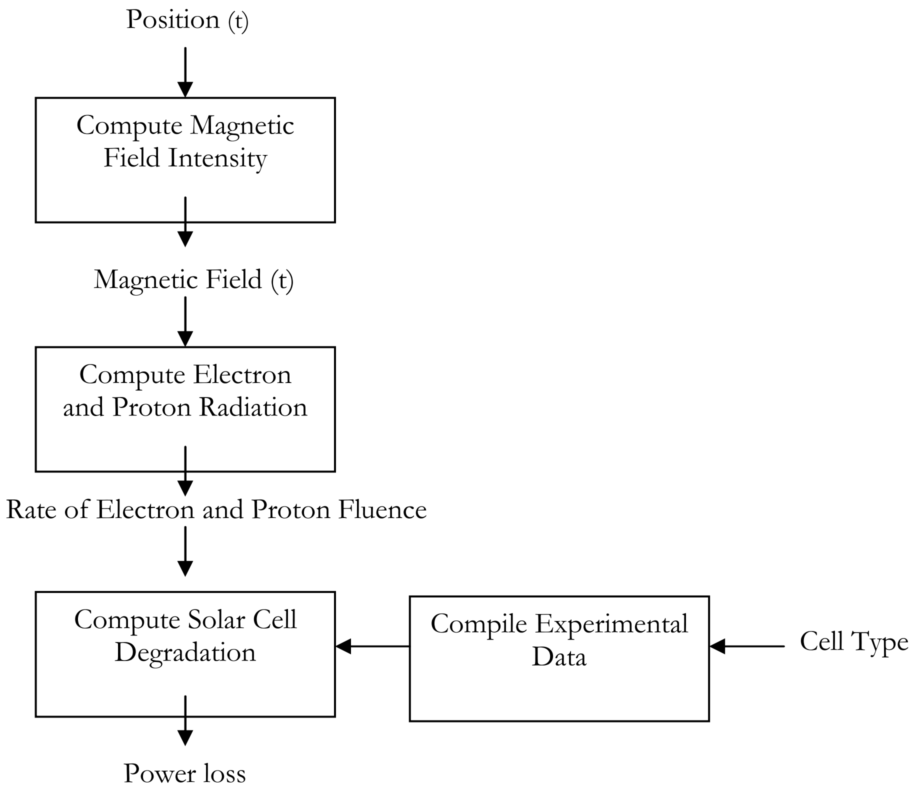
Degradation of solar array performance due to radiation within space environment is an important effect to consider when designing the power subsystem. Although not all spacecraft experience the fate of the ETS-VI in the Van Allen Belt, the solar arrays of a typical satellite in Low Earth Orbit loses 15% of its power producing capability in 5 years time. Furthermore, since many space missions are considering electric propulsion as its means for orbit transfer, the degradation effect on solar arrays will increase as more time is spent within the Van Allen radiation belts. As solar arrays are a critical complimentary subsystem to the power-intensive electric propulsion subsystem, it is important to provide a detailed characterization of the degradation effect on the solar arrays over the mission life.

PROBLEM STATEMENT

Characterize the degradation of a solar array given orbit position as a function of time. Conduct a literature search to better characterize the effect of radiation on solar arrays, beyond what has been presented in *Space Mission Analysis and Design* by Wertz and Larson. Develop a MATLAB module that characterizes the radiation dosage as a function of time. Also develop a method for computing degradation of the solar array based on the radiation dosage. To simply the problem, model the Van Allen Belts as a function of altitude and latitude only. In characterizing the radiation dose, model just the Van Allen Belts, but discuss how to incorporate other sources of radiation, such as solar particle events and galactic cosmic rays, into the module.

APPROACH

The following block diagram depicts the inputs and outputs of our model:



RADIATION ENVIRONMENT

Introduction

A spacecraft around Earth orbit is irradiated by many types of radiations, including trapped radiation, solar particle events, and Galactic cosmic rays. In particular, the trapped radiation of the Van Allen radiation belts is problematic to Earth orbiting satellites' solar arrays performance. An accurate model of the radiation from the Van Allen radiation belts can help in prediction of solar cell degradation during the design stage. In this section we describe a Matlab module that models the radiation due to trapped proton and electron particles around the Earth's magnetic field, i.e. Van Allen radiation belt.

Earth's Magnetic Field

Earth's magnetic field can be model using the International Association of Geomagnetism and Astronomy (IAGA) International Geomagnetic Reference Field (IGRF) model for given time (in UT) and location. More specifically, we use Tsyganenko's GEOPACK Library that has been written translated into a Matlab code from a Fortran code.

Using the GEOPACK Library, the magnetic field can be model for October 15, 2003, 18:00:00 UT at -71° longitude in Geocentric Geographic Coordinate system as shown in Figures 1.1 and 1.2.

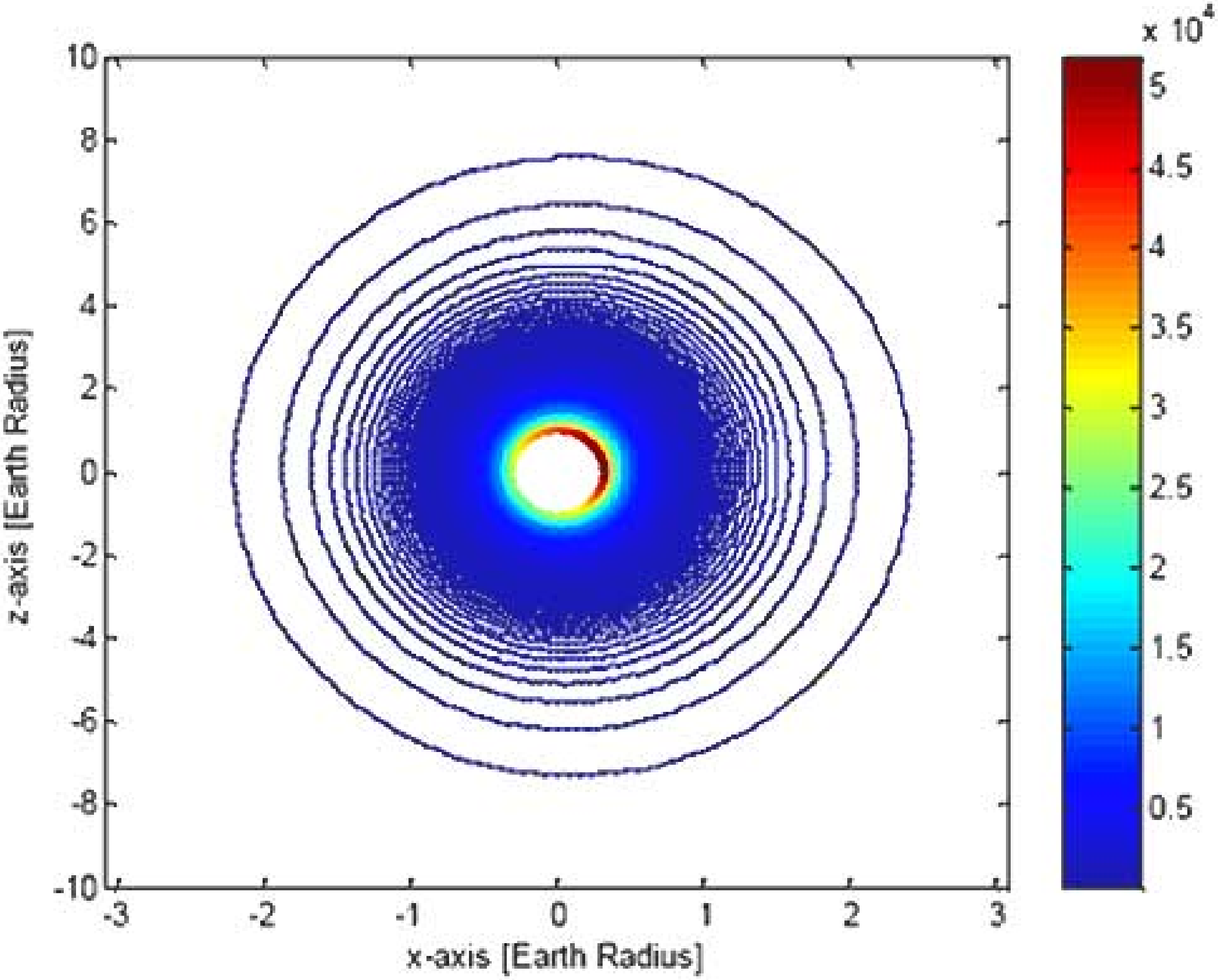


Figure 1.1 Magnetic field intensity on October 15, 2003, 18:00:00 UT at -71° longitude (Geocentric Geographic coordinate system)

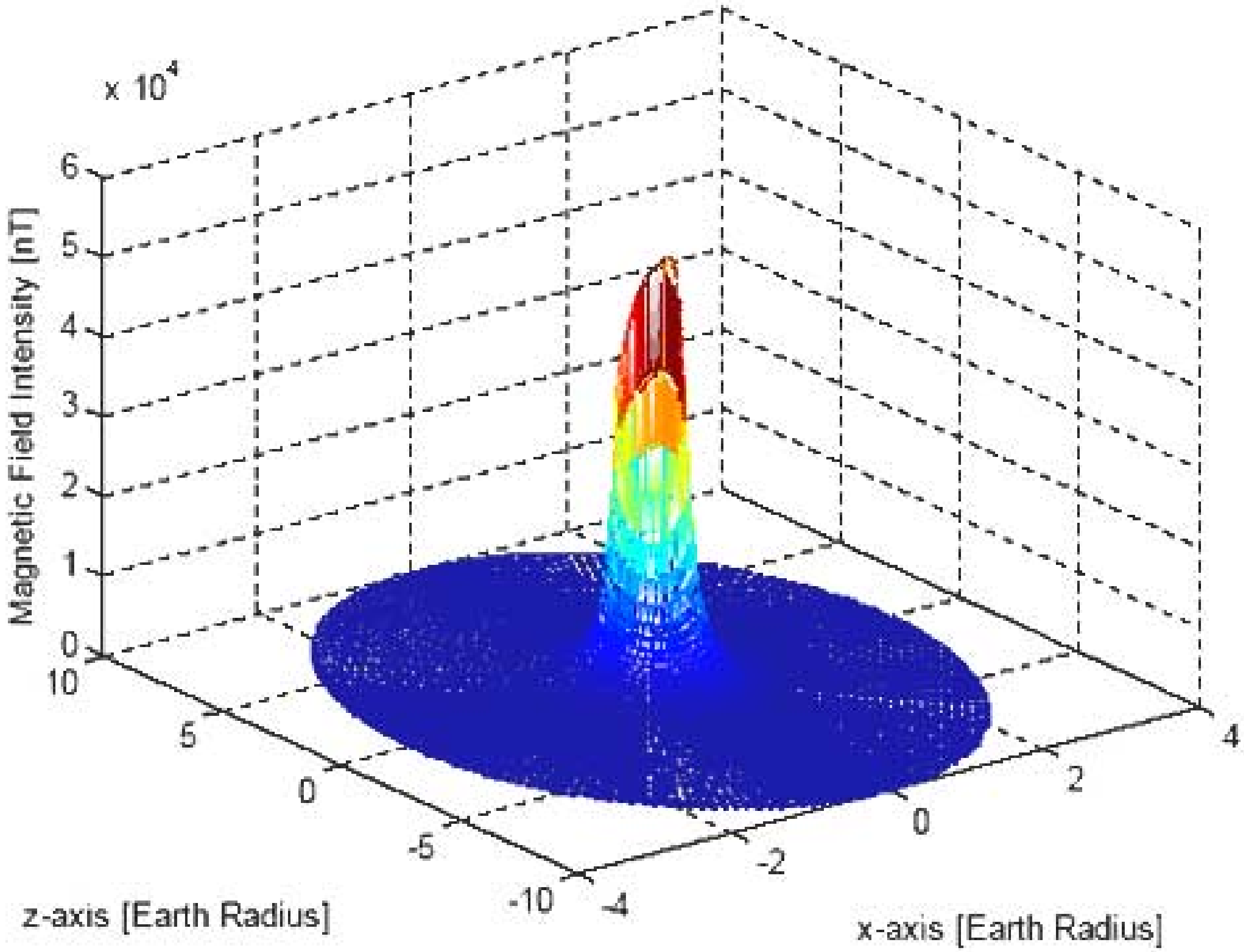


Figure 1.2 Magnetic field intensity on October 15, 2003, 18:00:00 UT at -71° longitude (Geocentric Geographic coordinate system)

Proton Electron Irradiation

Proton and electron irradiation is modeled using AP8MAX, AP8MIN, AE8MAX, and AE8MIN data. Bieter Bilitza originally wrote the model using Fortran for NASA Goddard Space Flight Center's Trapped Radiation Models Program, RADBELT. We translated this code in Matlab. As it's based on the experimental data from many satellites, it's only valid for a finite range of date, i.e. up to year 2005. Plots have been generated for the same location and time as the magnetic field intensity for both electron and proton, both at 1 MeV (see Figures 1.3 and 1.4).

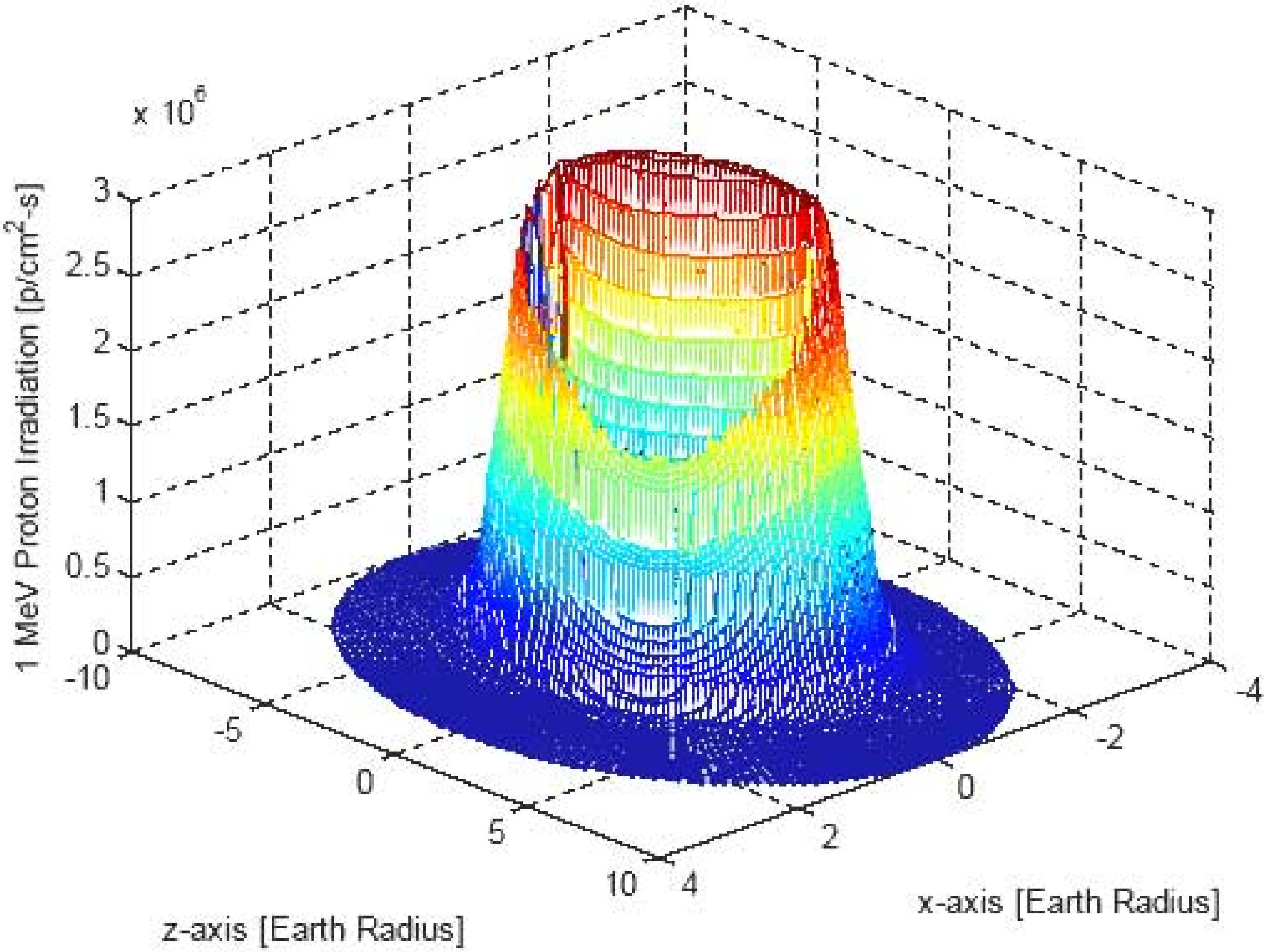


Figure 1.3 One MeV proton irradiation on October 15, 2003, 18:00:00 UT at -71° longitude (Geocentric Geographic coordinate system)

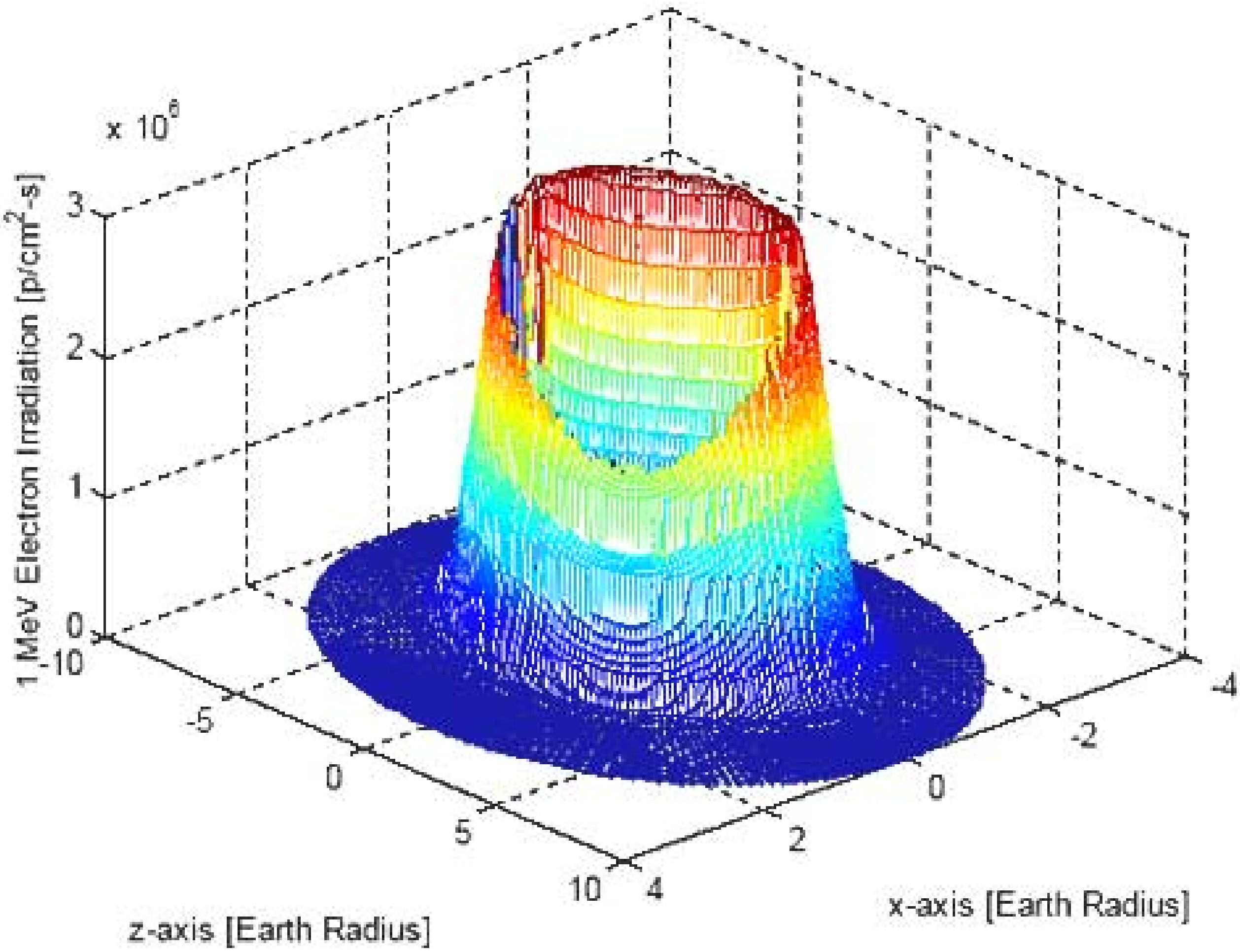


Figure 1.4 One MeV electron irradiation on October 15, 2003, 18:00:00 UT at -71° longitude (Geocentric Geographic coordinate system)

SOLAR CELL DEGRADATION

Introduction

There are two types of particle radiation that are harmful to solar cells: electrons and protons. Energized electrons and protons enter the cell and cause defects as they pass through and are slowed down (Sharps, 2003). This particle radiation affects the minority carrier diffusion length[1], which is an important parameter in determining solar cell performance.

---

[1] Solar cells contain regions of n-type and p-type material and p-n junctions where the two types join. An electrical charge generated by photons is carried by electrons in n-type material and by positively charged electron vacancies in p-type material. Vacancies, or holes in n-type material or electrons in p-type material are referred to as minority carriers. Extra electrons and holes are created on both sides of the p-n junctions when sunlight is absorbed. Minority carriers on each side then diffuse toward junctions, where they lower their energy by crossing over to the other side, generating an an electrical current. This electrical current and hence solar cell efficiency is reduced when minority carriers recombine with majority carriers before reaching a junction. The distance the carriers travel before recombination is called diffusion length.

The power output of solar cells is severely reduced by radiation damage. The solar arrays of a typical satellite in Low Earth Orbit loses 15% of its power producing capability in 5 years time. Predicting the degradation of the solar cells over the course of a mission can be crucial to determining the length of the mission. Power subsystems of spacecraft are designed to meet end-of-life requirements, and as such, it is necessary to predict power loss of solar cells over the mission life. There is limited existing experimental data of solar cell degradation as a function of electron and proton fluence. Therefore, a method for degradation predicition that relies on limited experimental data is desirable.

One approach to correlating power loss to radiation consists of multiplying the calculated nonionizing energy loss (NIEL)[2] and the fluence (Summers, 1994). The power losses associated with proton irradiation is a special case because of the linear dependance of the proton damage coefficient on the NIEL for all solar cells. Electron irradiation, however, does not vary linearly with NIEL and a more general approach is required. This method of characterizing the effects of radiation on solar cells in terms of absorbed displacement damage dose enables the prediction of power losses in the space environment with limited experimental data.

In the following sections, we demonstate the aforementioned method (Summers, 1994) for Gallium Arsenide solar cells. However, our modules can also be applied to Si (21eV), Si (12.9eV), and INP and solar cells for which experimental data of proton and electron fluence exists.

<u>Protons</u>

Figure 2.1 shows experimentally measured curves of the normalized power degradation for GaAs/Ge solar cells. Power degradation for the GaAs/Ge solar cell after exposure to protons for energy levels of 0.05, 0.1, 0.3, 1, 2.5, and 10 MeV are displayed.

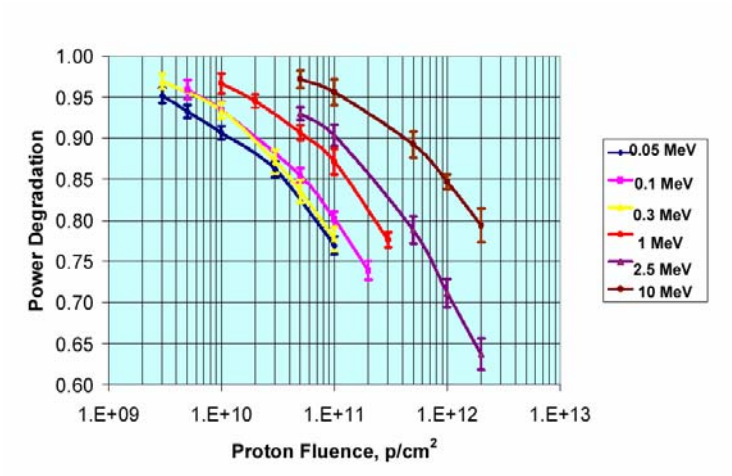


Figure 2.1 Power degradation for GaAs/Ge solar cell after exposure to protons, for different energies and fluences (Sharps, 2003).

As shown in Equation 2.1, a characteristic curve for proton damage (Figure 2.2) can then be obtained by multiplying the proton fluence, $\Phi(E)$, by the NIEL, $S(E)$, for a given energy level, E. (A table of NIEL values for protons and electrons in Si, GaAs and INP is included in the appendix).

[2] The NIEL is the rate at which an incident ion loses energy to displacement damage (Summers, 1991).

$$D = \Phi(E) \cdot S(E) \tag{2.1}$$

In this way, the data for all energy levels can be represented by a single characteric function.

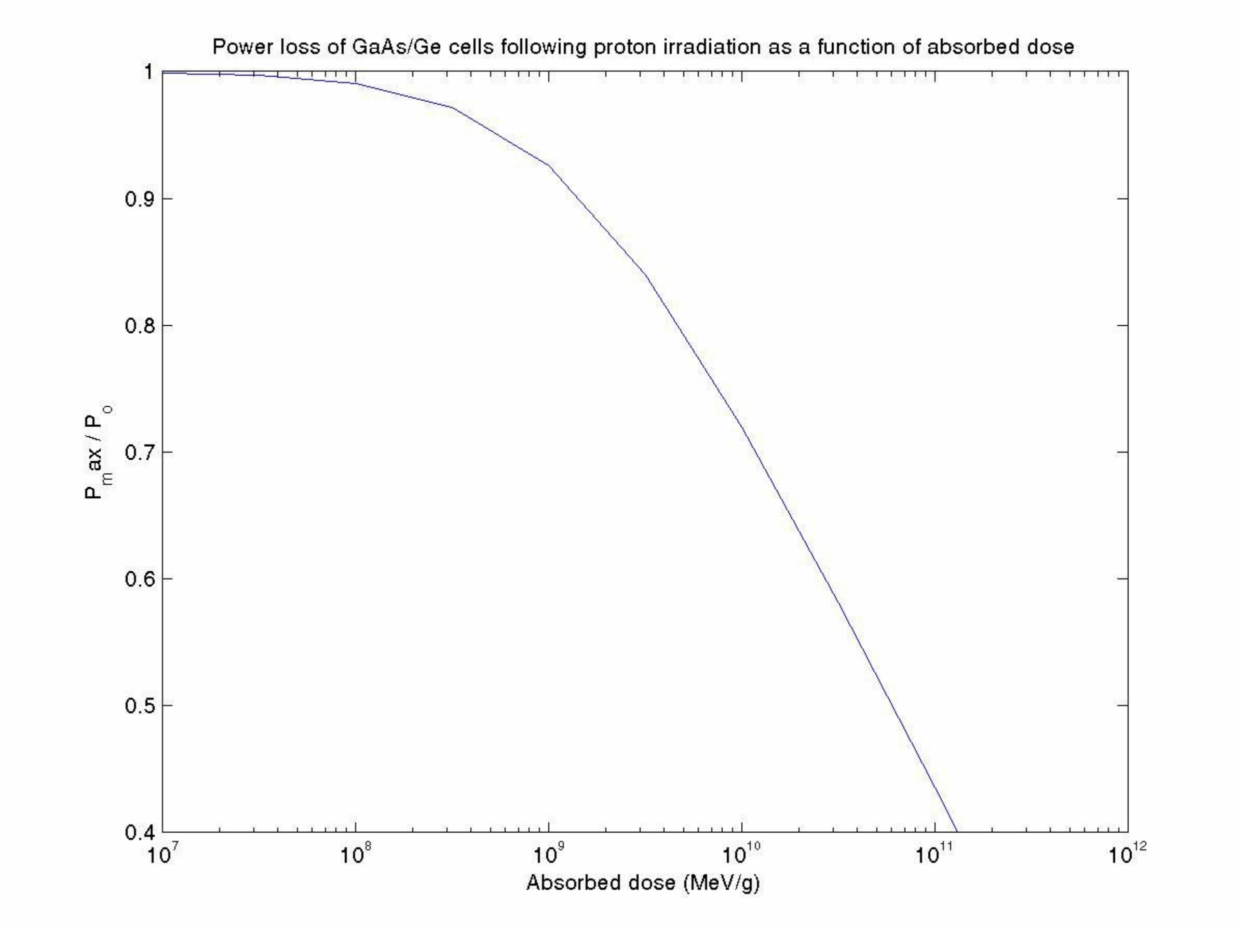


Figure 2.2 Characteristic curve for proton damage.

The characteristic curve is generated to model the power loss due to proton irradiation. The plot is fitted to experimental data (product of NIEL and fluence) and is defined by Equation 2.2. In Equation 2.2, P_{max}/P_o represents the normalized beginning-of-life power after exposure to a given dose of protons. D represents the dose in units of MeV/g. C and D_x are constants fitted to the experimental data, equal to .1295 and 1.295×10^9 respectively (Summers, 1994).

$$\frac{P_{\max}}{P_o} = 1 - C \ln\left(1 + \frac{D}{D_x}\right) \tag{2.2}$$

Equation 2.2 can be modified to compute the dose required to reduce the power to any given level:

$$D = D_x\left(\frac{e^{\frac{1-P_{max}}{P_o}}}{C} - 1\right) \qquad (2.3)$$

The primary advantage of this method is that the dose remains constant for any positive ion of energy. The plot provides the absorbed does for a given reduction of power. Dividing the dose by the NIEL for a energy value of our choice (not only energy values for which experimental data exists), we can output the fluence. This enables an entire group of damage coefficients to be obtained from a single characteristic curve.

Electrons

Figure 2.3 shows experimentally measured curves of the normalized power degradation for GaAs/Ge solar cells. Power degradation for the GaAs/Ge solar cell after exposure to electrons for energy levels of 0.6, 1, 2, and 12 MeV are displayed.

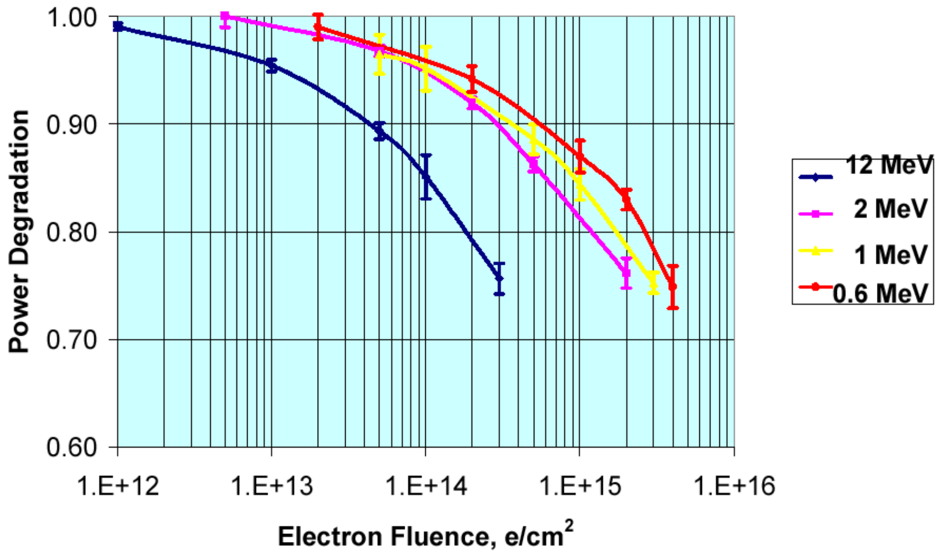


Figure 2.3 Power Degradation for GaAs/Ge solar cell after exposure to electrons, for different energies and fluences (Sharps, 2003).

As before, the damage energy deposited by electrons of a given energy is calculated using Equation 2.1. However, the electron case proves more complicated than the proton case because damage coefficients no longer vary linearly with the NIEL. Instead of generating a characteristic curve such as the one in Figure 2.2, a range of data is produced (Figure 2.4).

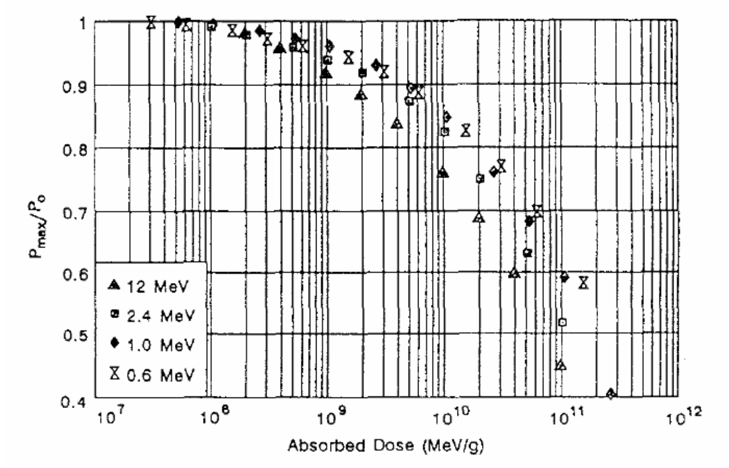


Figure 2.4 Power loss of GaAs/Ge cells from electron irradiation plotted against absorbed dose (Summers, 1994).

To compensate for the non-linear variation of damage coefficients and NIEL, it is necessary to define a effective dose of electron energy, D_{eff}. This value is typically chosen as 1.0 MeV (Ansbaugh, 1991). The value of n in Equation 2.4 must be experimentally determined.

$$D_{eff}(1.0) = D\left(\frac{S(E)}{S(1.0)}\right)^{(n-1)} \tag{2.4}$$

By defining this effective absorbed dose of 1.0 MeV of electron energy, a characteristic curve can be generated (Figure 2.5).

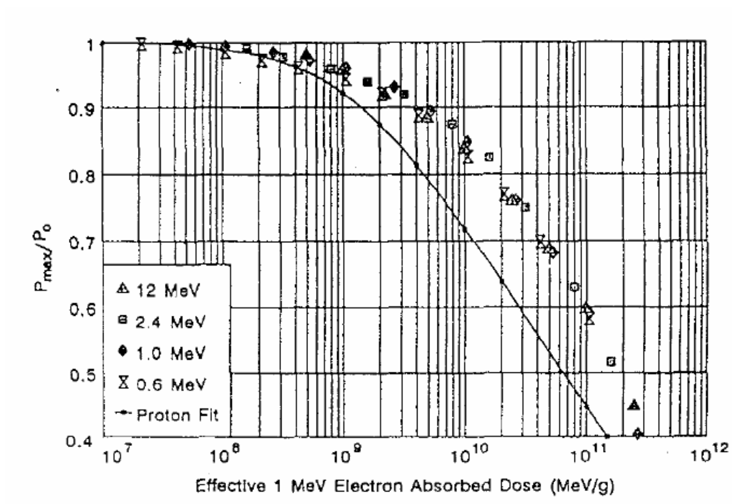


Figure 2.5 Power loss of GaAs/Ge cells from electron irradiation plotted against effective 1.0 MeV electron absorbed dose. (Summers, 1994).

Figure 2.5 is important not only for displaying the ability of the Equation 2.4 to return a characteristic curve, but also for displaying the relationship between degradation caused by electrons and protons. Manipulation of these plots will eventually enable correlation between proton and electron effects.

To demonstrate the applicability of this method of utilizing effective dose of electron energy, D_{eff}, for other solar cell types, the characteristic curve of Si solar cells is calculated. D_{eff} is again set to 1.0 MeV and n is found to be 2 for silicon from experimental data (Summers, 1994). Using Equations 2.1, 2.2, and 2.4, the electron absorbed dose is then plotted for various electron energies. As expected, the data coverges on a single line which can be defined by Equation 2.1 with C and D_x as constants fitted to plot, equal to 0.06593 and 1.841×10^8 respectively (Figure 2.6).

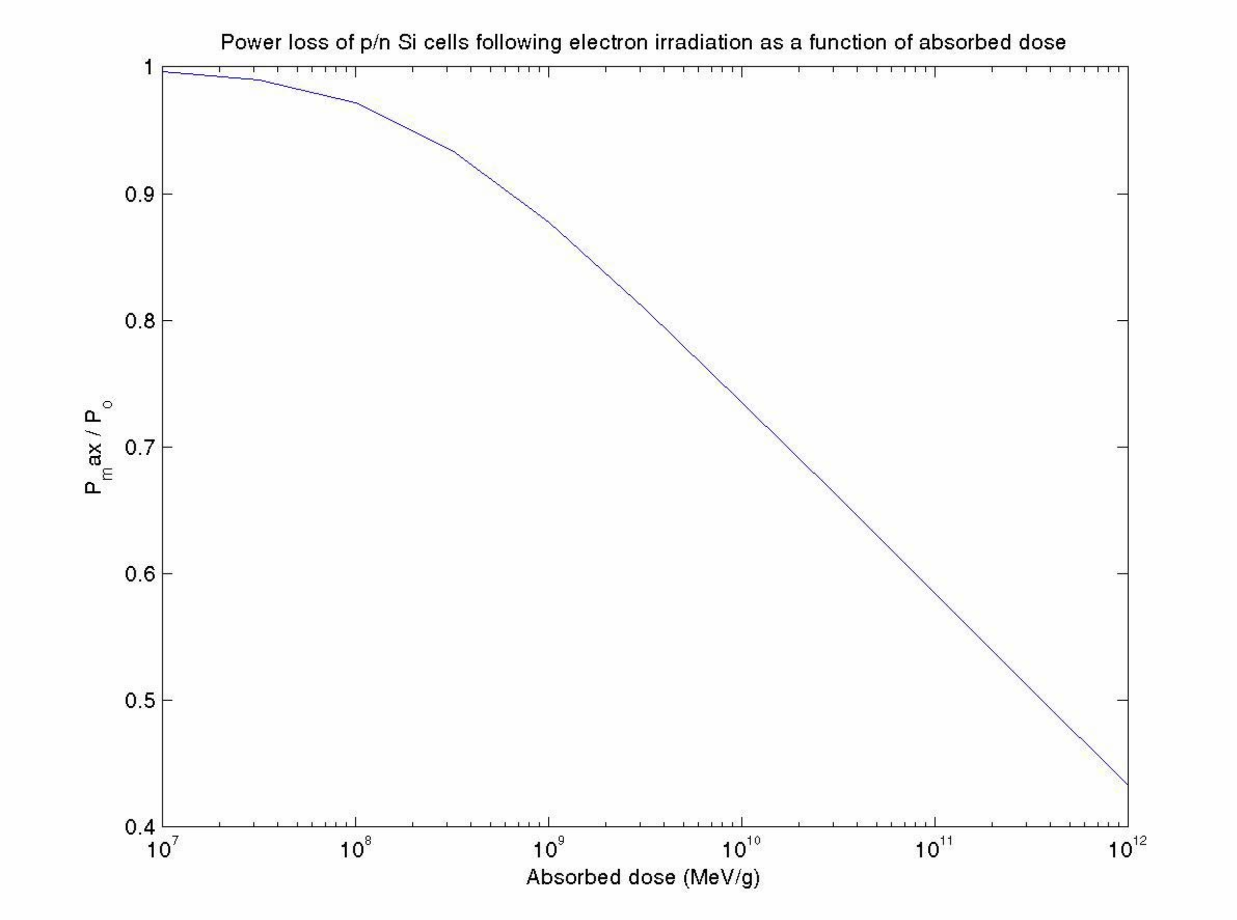


Figure 2.6 Characteristic curve for electron damage.

As in the proton case, the electron characteristic curve enables the fluence of electrons of any energy to be calculated for a given power loss. This can be useful in the solar cell selection process. For example, degradation of various solar cell types can be compared by calculation of the fluence of 1 MeV electrons.

Proton and Electron Effects Applied to Radiation Model

$$D = \iint \frac{d\phi(E,t)}{dE} S(E,t) dEdt \tag{2.5}$$

$$D_{eff}(1.0) = S(1.0)^{-(n-1)} \iint \frac{d\phi(E,t)}{dE} S(E,t)^n dEdt \tag{2.6}$$

SUGGESTIONS FOR FUTURE WORK

There are additional factors that should be included in future studies on solar cell performance in complex space environments.

In the solar cell degradation model, it is important to note that only cell degradation as a result of irradiation with a unidirectional beam of particles is considered. Solar cells in a real space environment will be subject to isotropic bombardments. Another simplifying assumption made in the solar cell degradation model was that low energy protons (those without enough energy to reach the active region of the cell) need not be considered. By choosing a suitbale lower limit used on the integral for the dose calculation, low energy protons are neglected. Although low energy protons can usually be neglected due to the logarithmic dependence of the maximum power on dose, it is something future work can consider.

CONCLUSIONS

Characterizing the effects of radiation on solar cells in terms of absorbed displacement damage dose as opposed to particle fluence reduces the experimental data needed to predict power losses in the space environment.

REFERENCES

N. A. Tsyganenko, A. V. Usmanov, Determination of the Magnetospheric Current System Parameters and Development of Experimental Geomagnetic Field Models Based on Data from IMP and HEOS Satellites, Planet. Space Sci. **30**, 985-998, 1982.

http://nssdc.gsfc.nasa.gov/space/model/

Allen, Douglas M. "A survey of next generation solar arrays (for spacecraft electric power)." *W. J. Schafer Associates*. AIAA, 1997.

Anspaugh, Bruce E. "Proton and Electron Damage Coefficients for GaAs/Ge Solar Cells. *Jet Propulsion Laboratory*. IEEE, 1991.

Gaddy, Edward M. "Relative cost effectiveness of multi-junction, gallium arsenide, and silicon solar cells." *NASA Goddard Space Flight Center*. AIAA, 1996.

Gardner, B. M. "A Tool for Assessing Photovoltaic Array End-of-Life Power Performance." *NASA Lewis Research Center.* AIAA, 1997.

Kerslake, Thomas W. "Solar Power System Options for the Radiation and Technology Demonstration Spacecraft." *NASA Glenn Research Center.* AIAA, 2000.

Landis, Geoffrey A. "Photovoltaic Power for Future NASA Missions." NASA *John Glenn Research Center.* AIAA, 2002.

Ralph, E. "Solar Cell Array System Trades – Present and Future." *TECSTAR/ASD.* AIAA, 1999.

"Research on Radiation Damage Mechanisms of Si Solar Cell." *NASDA Office of Research and Development.* No. 73, July 1998. http://www.nasda.go.jp/lib/nasda-news/1998/07/solar_e.html (5 October 2003).

Sharps, P. R. "High Efficiency Advanced Triple-Junction Space Solar Cell Production at Emcore Photovoltaics." *Emcore Photovoltaics.* AIAA, 2003.

"Solar Radiation Handbook." *Jet Propulsion Laboratory.* Third Edition, 1982.

Summers, Geoffrey P. "A General Method for Predicting Radiation Damage to Solar Cells in the Space Environment." *Naval Research Laboratory.* IEEE, 1991.

Summers, Geoffrey P. "A New Approach To Damage Prediction For Solar Cells Exposed To Different Radiations." *Naval Research Laboratory.* IEEE, 1994.

Visentine, J. "MIR Solar Array Return Experiment." *Boeing Space Station Program Office.* AIAA, 1999.

APPENDIX A: RADIATION ENVIRONMENT MODULE

```
function load_AP8AE8_data()
% LOAD_AP8AE8_DATA() loads AP8MAX, AP8MIN, AE8MAX, and AE8MIN data.
%       .type                   Model map type
%       .flux_log_increments_per_decade Increments per decadeof logarithmic flux
%
%       .epoch                  Epoch of model
%       .energy_scale_factor        Scale factor for energy = E(map)/(E/MeV)
%                                                      = 6400 (AE-8), 100 (AP-8)
%       .L_scale_factor           Scale factor for L
%                              = 2100 (AE-8), 2048 (AP-8)
%       .B_B0_scale_factor         scale factor for B/B0
%                              = 1024 (AE-8), 2048 (AP-8)
%       .flux_log_scale_factor      Scale factor for logarithm of fluxes
%                              = 1024 (AE,AP-8)
%       .number_of_elements        Number of elements in map
%                              = 13548 (AE8MAX), 13168 (AE8MIN),
%                                6509 (AP8MAX), 6688 (AP8MIN)
%       .map                    Data

global RADIATION_DB;
```

```
RADIATION_DB = struct('AP8MAX',[],'AP8MIN',[],'AE8MAX',[],'AE8MIN',[]);
file = {'ap8max.asc' 'ap8min.asc' 'ae8max.asc' 'ae8min.asc'};
data = {'AP8MAX' 'AP8MIN' 'AE8MAX' 'AE8MIN'};

for i = 1:length(data)
    x = textread(file{4},'%6n','delimiter',' ');
   RADIATION_DB = setfield(RADIATION_DB,data{i},struct('type',x(1),...
                                   'flux_log_increments_per_decade',x(2),...
                                   'epoch',x(3),...
                                   'energy_scale_factor',x(4),...
                                   'L_scale_factor',x(5),...
                                   'B_B0_scale_factor',x(6),...
                                   'flux_log_scale_factor',x(7),...
                                   'number_of_elements',x(8),...
                                   'map',x(9:length(x))));
end



function [proton_flux,electron_flux] = compute_irradiation_flux...
                                     (year,month,day,hour,minute,second,...
                                      radius,theta,phi,...
                                      proton_energy,electron_energy,solar_max)
% [PROTON_FLUX,ELECTRON_FLUX] =
%   COMPUTE_IRRADIATION_FLUX(YEAR,MONTH,DAY,HOUR,MINUTE,SECOND,
%                        RADIUS,THETA,PHI,PROTON_ENERGY,ELECTRON_ENERGY
%                        , ..
%                        SOLAR_MAX) computes theirradiation flux on an
%                        object.
%
% Input
%   year              UT time year, XXXX [A.D.]
%   month             UT time month, Jan, Feb, ... [1, 2, 3,...]
%   day               UT time [day]
%   hour              UT time [hr]
%   minute            UT time [min]
%   second            UT time [s]
%   radius            Geocentric Geographic sherical raidus [m]
%   theta             Geocentric Geographic sherical co-latitude [rad]
%   phi               Geocentric Geographic sherical azmuth [rad]
%   proton_energy     Proton energy of an interest [MeV]
%   electron_energy   Electron energy of an interest [MeV]
%   solar_max         Solar max? true or false [1,0]
%
% Output
%   proton_flux       Omnidirectional proton flux [p/cm^2/s]
%   electron_flux     Omnidirectional electon flux [p/cm^2/s]

R_Earth = 6378137;            % Earth radius [m]
```

```
% Decide which data to use.
if (solar_max)
   % If solar max, use AP8MAX and AE8MAX data.
   AP_data = 'AP8MAX';
   AE_data = 'AE8MAX';
else
   % Otherwise, use AP8MIN and AE8MIN data.
   AP_data = 'AP8MIN';
   AE_data = 'AE8MIN';
end

for i = 1:length(year)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This section uses the International Association of Geomagnetism and Aeronomy
% (IAGA) International Geomagnetic Refernce Field (IGRF) model to compute the
% magnetic field at a given time and location. The supporting functions are
% from:
%          Tsyganenko's GEOPACK Library
%          Translated from original FORTRAN April 10-11, 2003
%          By Paul O'Brien (original by N.A. Tsyganenko)
%          Paul.OBrien@aero.org (Nikolai.Tsyganenko@gsfc.nasa.gov)
%          Translation supported by NSF Grant ATM 0202107
   % Initialize for magnetic field intensity calculation.
   GEOPACK_RECALC(year,datenum(year(i),month(i),day(i))     -     datenum(year(i),1,1)     +
1,hour(i),min(i),sec(i));
   % Magnetic field vector at (r,theta,phi) in Geocentric Geographic
   % spherical coordinate [nT].
   [B_r,B_theta,B_phi] = GEOPACK_IGRF_GEO((radius(i)/R_Earth),theta,phi);
   % Magnetic field intensity [nT].
   BetaValueF = sqrt(B_r^2 + B_theta^2 + B_phi^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This section is from the radbelt_trmfun.c code written by:
%     August 10, 1998 Dan Leonard,
%       dleonard@cfa.harvard.edu,
%       (617) 496-7075
%       (617) 496-7049 fax
% This uses dipole approximation and can be improved. B0 is the magnetic
% field intensity at the magnetic equator. This value should be computed
% using the International Association of Geomagnetism and Aeronomy (IAGA)
% International Geomagnetic Refernce Field (IGRF) model. The L-value
% calculation method should be improved.
   % Compute L value
   bottomF = 4 - (BetaValueF^2*(radius(i)/R_Earth)^6/(3.06e4*3.06e4));
   L      = 3*(radius(i)/R_Earth)/bottomF;

   % From page 8 of the AP-8 Trapped Proton Documnent;
   % also easily derivable from dipole approximation
```

```
   B_B0 = BetaValueF*L^3/3.06e4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This section uses code translated from trmfun.for code written by:
%       Bieter Bilitza
%       TRMFUN.FOR
%       Trapped Radiation Models Program RADBELT
%       March 25, 1988
%       NASA Goddard Space Flight Center, code 633
   % Compute proton irradiation flux [p/cm^2/s]
   proton_flux(i,:) = 10.^trara1(L,B_B0,proton_energy,AP_data);
   % Compute electron irradiation flux [p/cm^2/s]
   electron_flux(i,:) = 10.^trara1(L,B_B0,electron_energy,AE_data);
end


function flux = trara1(L,B_B0,E,map_type)
% FLUX = TRARA1(L,B_B0,ENERGY,MAP_TYPE) computes proton and electron flux.
%
% Input
%   L          L-value of the magnetic shell currently in
%   B_B0       Magnetic field intensity normallized by the
%              magnetic field intensity at the magnetic
%              equitorial on the surface of Earth
%   E          Energy [MeV]
%   map_type   Trapped radiation map type [AP8MAX,AP8MIN,AE8MAX,AE8MIN]
%
% Output
%   flux       Decadic logarithm of integral fluxe [p/cm^2/s]
%
%
% This section uses code translated from trmfun.for code written by:
%       Bieter Bilitza
%       TRMFUN.FOR
%       Trapped Radiation Models Program RADBELT
%       March 25, 1988
%       NASA Goddard Space Flight Center, code 633

global RADIATION_DB;
if map_type == 'AP8MAX'
   map_description = RADIATION_DB.AP8MAX;
   MAP             = RADIATION_DB.AP8MAX.map;
elseif map_type == 'AP8MIN'
   map_description = RADIATION_DB.AP8MIN;
   MAP             = RADIATION_DB.AP8MIN.map;
elseif map_type == 'AE8MAX'
   map_description = RADIATION_DB.AE8MAX;
   MAP             = RADIATION_DB.AE8MAX.map;
elseif map_type == 'AE8MIN'
```

```
   map_description = RADIATION_DB.AE8MIN;
   MAP             = RADIATION_DB.AE8MIN.map;
end

F1 = 1.001;
F2 = 1.002;

FISTEP = map_description.flux_log_scale_factor/map_description.flux_log_increments_per_decade;
ESCALE = map_description.energy_scale_factor;
FSCALE = map_description.flux_log_scale_factor;
XNL = min(15.6,abs(L));
NL = XNL*map_description.L_scale_factor;
if B_B0 < 1
   B_B0 = 1;
end
NB = (B_B0 - 1)*map_description.B_B0_scale_factor;

% I2 IS THE NUMBER OF ELEMENTS IN THE FLUX MAP FOR THE FIRST ENERGY.
% I3 IS THE INDEX OF THE LAST ELEMENT OF THE SECOND ENERGY MAP.
% L3 IS THE LENGTH OF THE MAP FOR THE THIRD ENERGY.
% E1 IS THE ENERGY OF THE FIRST ENERGY MAP (UNSCALED)
% E2 IS THE ENERGY OF THE SECOND ENERGY MAP (UNSCALED)
I1 = 0;
I1 = 0;
I2 = MAP(1);
I3 = I2 + MAP(I2+1);
L3 = MAP(I3+1);
E1 = MAP(I1+2)/ESCALE;
E2 = MAP(I2+2)/ESCALE;

% S0, S1, S2 ARE LOGICAL VARIABLES WHICH INDICATE WHETHER THE FLUX FOR
% A PARTICULAR E, B, L POINT HAS ALREADY BEEN FOUND IN A PREVIOUS CALL
% TO FUNCTION TRARA2. IF NOT, S.. =.TRUE.
S1 = 1;
S2 = 1;

% ENERGY LOOP
for IE = 1:size(E)
   % FOR EACH ENERGY E(I) FIND THE SUCCESSIVE ENERGIES E0,E1,E2 IN
   % MODEL MAP, WHICH OBEY  E0 < E1 < E(I) < E2 .
   while (~((E(IE) <= E2) | (L3==0)))
      I0 = I1;
         I1 = I2;
      I2 = I3;
         I3 = I3 + L3;
      L3 = MAP(I3+1);
         E0 = E1;
         E1 = E2;
      E2 = MAP(I2+2)/ESCALE;
         S0 = S1;
```

```
        S1 = S2;
     S2 = 1;
        F0 = F1;
        F1 = F2;
   end

   % CALL TRARA2 TO INTERPOLATE THE FLUX-MAPS FOR E1,E2 IN L-B/B0-
   % SPACE TO FIND FLUXES F1,F2 [IF THEY HAVE NOT ALREADY BEEN
   % CALCULATED FOR A PREVIOUS E(I)].
   if (S1)
      F1 = trara2(NL,NB,map_type,I1+3-1)/FSCALE;
   end
   if (S2)
      F2 = trara2(NL,NB,map_type,I2+3-1)/FSCALE;
   end
   S1 = 0;
   S2 = 0;

   % FINALLY, INTERPOLATE IN ENERGY.
   F(IE) = F1 + (F2 - F1)*(E(IE) - E1)/(E2 - E1);
   if ~((F2 > 0) | (I1 == 0))
      % --------- SPECIAL INTERPOLATION ---------------------------------
      % IF THE FLUX FOR THE SECOND ENERGY CANNOT BE FOUND (I.E. F2=0.0),
      % AND THE ZEROTH ENERGY MAP HAS BEEN DEFINED (I.E. I1 NOT EQUAL 0),
      % THEN INTERPOLATE USING THE FLUX MAPS FOR THE ZEROTH AND FIRST
      % ENERGY AND CHOOSE THE MINIMUM OF THIS INTERPOLATIONS AND THE
      % INTERPOLATION THAT WAS DONE WITH F2=0.
      if (S0)
         F0 = trara2(NL,NB,map_type,I0+3-1)/FSCALE;
      end
      S0 = 0;
      F(IE) = min(F(IE),F0+(F1-F0)*(E(IE)-E0)/(E1-E0));
   end
   F(IE) = max(F(IE),0);
end
flux = F;

function f = trara2(IL,IB,map_type,offset)
%*****************************************************************
%***  TRARA2 INTERPOLATES LINEARLY IN L-B/B0-MAP TO OBTAIN     ***
%***  THE LOGARITHM OF INTEGRAL FLUX AT GIVEN L AND B/B0.      ***
%***    INPUT: MAP[] IS SUB-MAP (FOR SPECIFIC ENERGY) OF       ***
%***                TRAPPED RADIATION MODEL MAP                ***
%***          IL     SCALED L-VALUE                            ***
%***          IB     SCALED B/B0-1                             ***
%***   OUTPUT: TRARA2  SCALED LOGARITHM OF PARTICLE FLUX       ***
%*****************************************************************
%***  SEE MAIN PROGRAM 'MODEL' FOR EXPLANATION OF MAP FORMAT   ***
%***  SCALING FACTORS.                                         ***
%***  THE STEPSIZE FOR THE PARAMETERIZATION OF THE LOGARITHM   ***
```

```
%***  OF FLUX IS OBTAINED FROM 'COMMON/TRA2/'.                ***
%***************************************************************/
% /*     FUNCTION TRARA2(int MAP,IL,IB) */

global RADIATION_DB;
if map_type == 'AP8MAX'
   map_description = RADIATION_DB.AP8MAX;
   MAP             = RADIATION_DB.AP8MAX.map;
elseif map_type == 'AP8MIN'
   map_description = RADIATION_DB.AP8MIN;
   MAP             = RADIATION_DB.AP8MIN.map;
elseif map_type == 'AE8MAX'
   map_description = RADIATION_DB.AE8MAX;
   MAP             = RADIATION_DB.AE8MAX.map;
elseif map_type == 'AE8MIN'
   map_description = RADIATION_DB.AE8MIN;
   MAP             = RADIATION_DB.AE8MIN.map;
end
FISTEP = map_description.flux_log_scale_factor/map_description.flux_log_increments_per_decade;
I2     = 0;
ITIME  = 0;
FNB    = IB;
FNL    = IL;
% FIND CONSECUTIVE SUB-SUB-MAPS FOR SCALED L-VALUES LS1,LS2,
% WITH IL LESS OR EQUAL LS2.  L1,L2 ARE LENGTHS OF SUB-SUB-MAPS.
% I1,I2 ARE INDECES OF FIRST ELEMENTS MINUS 1.
L2 = MAP(offset+I2+1);
while (MAP(offset+I2+2) <= IL)
   I1 = I2;
   L1 = L2;
   I2 = I2 + L2;
   L2 = MAP(offset+I2+1);
end

% IF SUB-SUB-MAPS ARE EMPTY, I. E. LENGTH LESS 4, THAN TRARA2=0
if ((L1 < 4) & (L2 < 4))
   f = 0;
   return;
end

% IF FLOG2 LESS FLOG1, THAN LS2 FIRST MAP AND LS1 SECOND MAP
if (MAP(offset+I2+3) <= MAP(offset+I1+3))
   KT = I1;
   I1 = I2;
   I2 = KT;
   KT = L1;
   L1 = L2;
   L2 = KT;
end
```

```
goto_position = 0;

% DETERMINE INTERPOLATE IN SCALED L-VALUE
FLL1 = MAP(offset+I1+2);
FLL2 = MAP(offset+I2+2);
DFL = (FNL - FLL1)/(FLL2 - FLL1);
FLOG1 = MAP(offset+I1+3);
FLOG2 = MAP(offset+I2+3);
FKB1 = 0;
FKB2 = 0;
if (L1 < 4)
   goto_position = 32;
end

if (goto_position == 0)
   % B/B0 LOOP
   for J2 = 4:L2
      FINCR2 = MAP(offset+I2+J2);
      if ((FKB2 + FINCR2) > FNB)
         goto_position = 23;
         break;
      end
      FKB2 = FKB2 + FINCR2;
      FLOG2 = FLOG2 - FISTEP;
   end
end

if (goto_position == 0)
   ITIME = ITIME + 1;

   KT = I1;
   I1 = I2;
   I2 = KT;
   KT = L1;
   L1 = L2;
   L2 = KT;

   % DETERMINE INTERPOLATE IN SCALED L-VALUE
   FLL1 = MAP(offset+I1+2);
   FLL2 = MAP(offset+I2+2);
   DFL = (FNL - FLL1)/(FLL2 - FLL1);
   FLOG1 = MAP(offset+I1+3);
   FLOG2 = MAP(offset+I2+3);
   FKB1 = 0;
   FKB2 = 0;
   if (L1 < 4)
      goto_position = 32;
   end

   if (goto_position == 0)
```

```
        % B/B0 LOOP
        for J2 = 4:L2
            FINCR2 = MAP(I2+J2);
            if ((FKB2 + FINCR2) > FNB)
                goto_position = 23;
                break;
            end
            FKB2 = FKB2 + FINCR2;
            FLOG2 = FLOG2 - FISTEP;
        end
        if (goto_position == 0)
            ITIME = ITIME + 1;
            f = 0;
            return;
        end
    end
end

if (goto_position == 0 | goto_position == 23)
    goto_position = 0;
    if (ITIME == 1)
        goto_position = 30;
    elseif (J2 == 4)
        goto_position = 28;
    else
        SL2 = FLOG2/FKB2;
        for J1 = 4:L1
            FINCR1 = MAP(offset+I1+J1);
                 FKB1=FKB1+FINCR1;
            FLOG1=FLOG1-FISTEP;
            FKBJ1=((FLOG1/FISTEP)*FINCR1+FKB1)/((FINCR1/FISTEP)*SL2 + 1);
            if (FKBJ1 <= FKB1)
                goto_position = 31;
                break;
            end
        end
    end
end

if (goto_position == 0 & (FKBJ1 <= FKB2))
    f = 0;
    return;
end

if ((goto_position == 0 | goto_position == 31) & (FKBJ1 <= FKB2))
    goto_position = 29;
end;

if (goto_position == 0)
    FKB1 = 0;
```

```
end

if (goto_position == 0 | goto_position == 30)
   goto_position == 0;
   FKB2 = 0;
end

if (goto_position == 0 | goto_position == 32)
   goto_position == 0;
   J2 = 4;
   FINCR2 = MAP(I2+J2);
   FLOG2 = MAP(I2+3);
   FLOG1 = MAP(I1+3);
end

if (goto_position == 0 | goto_position == 28)
   goto_position = 0;
   FLOGM = FLOG1 + (FLOG2-FLOG1)*DFL;
   FKBM = 0;
   FKB2 = FKB2 + FINCR2;
   FLOG2 = FLOG2 - FISTEP;
   SL2 = FLOG2/FKB2;
   if (L1 < 4)
      goto_position = 35;
   end
   if (goto_position == 0)
      J1 = 4;
      FINCR1 = MAP(I1+J1);
      FKB1 = FKB1 + FINCR1;
      FLOG1 = FLOG1 - FISTEP;
      SL1 = FLOG1/FKB1;
      goto_position = 15;
   end
end

if (goto_position == 0 | goto_position == 29)
   goto_position = 0;
   FKBM = FKBJ1 + (FKB2 - FKBJ1)*DFL;
   FLOGM = FKBM*SL2;
   FLOG2 = FLOG2-FISTEP;
   FKB2 = FKB2+FINCR2;
   SL1 = FLOG1/FKB1;
   SL2 = FLOG2/FKB2;
end

if (goto_position == 35)
   goto_position = 0;
   FINCR1 = 0;
   SL1 = -900000;
   FKBJ1 = ((FLOG1/FISTEP)*FINCR1+FKB1)/((FINCR1/FISTEP)*SL2 + 1);
```

```
    FKB = FKBJ1 + (FKB2 - FKBJ1)*DFL;
    FLOG = FKB*SL2;
    if (FKB >= FNB)
        goto_position = 60;
        break;
    end
    FKBM = FKB;
    FLOGM = FLOG;
    if (J2 >= L2)
        f = 0;
        return;
    end
    J2 = J2 + 1;
    FINCR2 = MAP(I2+J2);
    FLOG2 = FLOG2 - FISTEP;
    FKB2 = FKB2 + FINCR2;
    SL2 = FLOG2/FKB2;
    goto_position = 15;
end

if (goto_position == 0 | goto_position == 15)
    goto_position = 0;
    while (1)
        if (SL1 >= SL2)
            FKBJ2 = ((FLOG2/FISTEP)*FINCR2 + FKB2)/((FINCR2/FISTEP)*SL1 + 1);
                FKB = FKB1 + (FKBJ2 - FKB1)*DFL;
                FLOG = FKB*SL1;
            if (FKB >= FNB)
                goto_position = 60;
                break;
            end
                FKBM = FKB;
             FLOGM = FLOG;
            if (J1 >= L1)
                f = 0;
                return;
            end
             J1 = J1 + 1;
            FINCR1 = MAP(I1+J1);
             FLOG1 = FLOG1 - FISTEP;
             FKB1=FKB1+FINCR1;
             SL1 = FLOG1/FKB1;
        else
            FKBJ1 = ((FLOG1/FISTEP)*FINCR1+FKB1)/((FINCR1/FISTEP)*SL2 + 1);
            FKB = FKBJ1 + (FKB2 - FKBJ1)*DFL;
            FLOG = FKB*SL2;
            if (FKB >= FNB)
                goto_position = 60;
                break;
            end
```

```
            FKBM = FKB;
            FLOGM = FLOG;
            if (J2 >= L2)
                f = 0;
                return;
            end
            J2 = J2 + 1;
            FINCR2 = MAP(I2+J2);
            FLOG2 = FLOG2 - FISTEP;
            FKB2 = FKB2 + FINCR2;
            SL2 = FLOG2/FKB2;
        end
    end
end

if (goto_position == 60)
    goto_position = 0;
    if (FKB < (FKBM + 1e-10))
        f = 0;
        return;
    end
    f = FLOGM + (FLOG - FLOGM)*((FNB - FKBM)/(FKB - FKBM));
    f = max(f,0);
    return;
end

f = 0;
```

APPENDIX B: SOLAR CELL DEGRADATION MODULE

```
% Predicting Solar Cell Degradation as a Function of Radiation
% Seung Chung & Matthew Richards
% 16.851 Problem Set #3

% This module performs two tasks
% 1. Model power loss of GaAs/Ge cells following proton irradiation
% 2. Model power loss of n/p Si cells following electron irradiation

% Define Constants
P_o = 1
C = .1295
D_x = 1.295*10^9

% Input Dose of Protons
D = [10^7 10^7.5 10^8 10^8.5 10^9 10^9.5 10^10 10^10.5 10^11 10^11.5 10^12]  % MeV/g

% Calculate Characteristic Curve For Proton Damage of GaAs/Ge cells
P_max = (1-C*log(1+D./D_x)).*P_o
```

```
% Graph P_max / P_o, the normalized maximum power of GaAs/Ge cells as a function of absorbed proton dose
figure
semilogx(D, P_max, '-');
title('Power loss of GaAs/Ge cells following proton irradiation as a function of absorbed dose')
xlabel('Absorbed dose (MeV/g)')
ylabel('P_max / P_o')
AXIS([10^7 10^12 .4 1])
hold

% Electron induced damage
% Overall equation: D_eff = D * (S_e(E) / S_e(1.0))^(n-1)

% Defining variables
D_eff = 1.0; % Measured in MeV

% Experimentally determined exponents
% n=1 for n-type silicon
% n=2 for p-type silicon, GaAs, and InP

% Calculate Characteristic Curve For Electron Damage of n/p Si cells
P_o = 1
C = 0.06593
D_x = 1.841*10^8
P_max = (1-C*log(1+D./D_x)).*P_o

% Input Dose of Electrons
D = [10^7 10^7.5 10^8 10^8.5 10^9 10^9.5 10^10 10^10.5 10^11 10^11.5 10^12]
% MeV/g

% Graph P_max / P_o, the normalized maximum power of n/p Si cells as a function of absorbed electron dose
figure
semilogx(D, P_max, '-');
title('Power loss of n/p Si cells following electron irradiation as a function of absorbed dose')
xlabel('Absorbed dose (MeV/g)')
ylabel('P_max / P_o')
AXIS([10^7 10^12 .4 1])
Hold
```

Table of NIEL Values for Protons and Electrons in Si, GaAs and INP

		PROTON ($MeV.cm^2/g$)				ELECTRON ($MeV.cm^2/g$)		
ENERGY(MeV)	Si(21eV)	Si(12.9eV)	GaAs	InP	Si(21eV)	Si(12.9eV)	GaAs	InP
1×10^{-4}	-	2.581	-	5.778	-	-	-	-
1×10^{-3}	1.546×10^{1}	1.964×10^{1}	1.150×10^{1}	1.288×10^{1}	-	-	-	-
2	1.053×10^{1}	1.262×10^{1}	8.071	8.757	-	-	-	-
3	8.098	9.492	6.275	6.731	-	-	-	-
5	5.659	6.496	4.434	4.706	-	-	-	-
7	4.413	5.010	3.479	3.673	-	-	-	-
1×10^{-2}	3.360	3.779	2.664	2.800	-	-	-	-
2	1.938	2.147	1.551	1.619	-	-	-	-
3	1.390	1.529	1.118	1.163	-	-	-	-
5	9.058×10^{-1}	9.895×10^{-1}	7.338×10^{-1}	7.607×10^{-1}	-	-	-	-
7	6.799×10^{-1}	7.396×10^{-1}	5.536×10^{-1}	5.729×10^{-1}	-	-	-	-
1×10^{-1}	5.003×10^{-1}	5.421×10^{-1}	4.093×10^{-1}	4.228×10^{-1}	-	-	-	-
2	2.722×10^{-1}	2.931×10^{-1}	2.251×10^{-1}	2.318×10^{-1}	-	9.741×10^{-6}	-	3.883×10^{-6}
3	1.895×10^{-1}	2.034×10^{-1}	1.578×10^{-1}	1.622×10^{-1}	6.481×10^{-6}	1.850×10^{-5}	2.350×10^{-6}	6.713×10^{-5}
5	1.194×10^{-1}	1.278×10^{-1}	1.005×10^{-1}	1.031×10^{-1}	1.629×10^{-5}	2.973×10^{-5}	1.177×10^{-5}	1.719×10^{-5}
7	8.773×10^{-2}	9.371×10^{-2}	7.441×10^{-2}	7.624×10^{-2}	2.320×10^{-5}	3.453×10^{-5}	1.861×10^{-5}	2.482×10^{-5}
1×10^{0}	6.381×10^{-2}	6.727×10^{-2}	5.402×10^{-2}	5.527×10^{-2}	3.142×10^{-5}	4.253×10^{-5}	2.656×10^{-5}	3.348×10^{-5}
2	3.296×10^{-2}	3.506×10^{-2}	2.886×10^{-2}	2.944×10^{-2}	5.069×10^{-5}	6.137×10^{-5}	4.486×10^{-5}	5.223×10^{-5}
3	2.243×10^{-2}	2.383×10^{-2}	1.993×10^{-2}	2.031×10^{-2}	6.366×10^{-5}	7.417×10^{-5}	5.655×10^{-5}	6.370×10^{-5}
5	1.376×10^{-2}	1.460×10^{-2}	1.248×10^{-2}	1.269×10^{-2}	8.106×10^{-5}	9.147x10-5	7.182×10^{-5}	7.870×10^{-5}
7	1.047×10^{-2}	1.107×10^{-2}	9.155×10^{-3}	9.374×10^{-3}	9.282×10^{-5}	1.031×10^{-4}	8.208×10^{-5}	8.879×10^{-5}
1×10^{1}	7.885×10^{-3}	7.885×10^{-3}	6.588×10^{-3}	7.241×10^{-3}	1.051×10^{-4}	1.153×19^{-4}	9.301×10^{-5}	9.948×10^{-5}
2	5.360×10^{-3}	5.360×10^{-3}	4.693×10^{-3}	4.682×10^{-3}	1.271×10^{-4}	1.373×10^{-4}	1.140×10^{-4}	1.199×10^{-4}
3	4.778×10^{-3}	4.778×10^{-3}	4.028×10^{-3}	3.958×10^{-3}	1.381×10^{-4}	1.482×10^{-4}	1.257×10^{-4}	1.313×10^{-4}
5	3.884×10^{-3}	3.884×10^{-3}	3.749×10^{-3}	3.356×10^{-3}	1.496×10^{-4}	1.597×10^{-4}	1.395×10^{-4}	1.448×10^{-4}
7	3.161×10^{-3}	3.161×10^{-3}	3.640×10^{-3}	3.107×10^{-3}	1.555×10^{-4}	1.657×10^{-4}	1.476×10^{-4}	1.529×10^{-4}
1×10^{2}	2.598×10^{-3}	2.598×10^{-3}	3.490×19^{-3}	3.005×10^{-3}	1.603×10^{-4}	1.704×10^{-4}	1.550×10^{-4}	1.606×10^{-4}
2	1.940×10^{-3}	1.940×10^{-3}	3.930×10^{-3}	3.537×10^{-3}	1.651×10^{-4}	1.751×10^{-4}	1.660×10^{-4}	1.721×10^{-4}