



James Ramdeo

Analysis of Space-Flight Radiation Damage to ExoMars Trace Gas Orbiter NOMAD-UVIS Spectrometer

School of Aerospace, Transport and Manufacturing
Astronautics and Space Engineering

MSc
Academic Year: 2024 – 2025

Supervisor: Dr. Jennifer Kingston
Associate Supervisors: Dr. Ben Dryer and
Dr. Jon Mason
August 2025





School of Aerospace, Transport and Manufacturing

Astronautics and Space Engineering

MSc

Academic Year: 2024 – 2025

James Ramdeo

Analysis of Space-Flight Radiation Damage to ExoMars Trace Gas Orbiter NOMAD-UVIS Spectrometer

Supervisor: Dr. Jennifer Kingston

Associate Supervisors: Dr. Ben Dryer and

Dr. Jon Mason

August 2025

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

© Cranfield University 2025. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

ExoMars Trace Gas Orbiter (TGO) mission, launched in 2016 as part of the European Space Agency (ESA) ExoMars programme, carries the Nadir and Occultation for MArs Discovery (NOMAD) instrument suite which consists of the Ultraviolet and Visible Spectrometer (UVIS). Operating in Martian orbit, NOMAD-UVIS is subject to accumulating radiation-induced degradation that can impact detector performance and long-term data quality. This study explores radiation-induced damage to NOMAD-UVIS through an analysis of dark current evolution, integrating both pre-flight ground testing data and in-flight mission data. The results establish a comparison between ground test predictions of radiation-induced dark current via the use of estimated radiation doses, and the actual in-flight performance of the detector with consideration of the Martian radiation environment.

Keywords:
CCD, Dark Current, Annealing, Dark Frames, Spectrometer, Bright Pixels.

Acknowledgements

The author would like to thank his parents for their unwavering support and encouragement throughout the course of this Master's degree. The author would also like to thank his friends, with whom he has shared endless laughs with, lifelong memories and have helped strengthen the drive and passion to pursue this MSc to completion. A very special thanks to Dr. Jennifer Kingston, Dr. Ben Dryer and Dr. Jon Mason for the relentless and exceptional academic supervision and support throughout this thesis period.

Contents

Academic Integrity Declaration i

Abstract ii

Acknowledgements iii

Contents iv

List of Figures vii

List of Tables ix

Listings x

List of Abbreviations xii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Thesis Aim and Objectives . . . 1
  - 1.2.1 Thesis Aim . . . 2
  - 1.2.2 Thesis Objectives . . . 2
- 1.3 Thesis Composition . . . 3

2 ExoMars and the Space Radiation Environment 4
- 2.1 ExoMars TGO . . . 4
  - 2.1.1 The ExoMars Mission . . . 4
  - 2.1.2 Characterisation of the NOMAD-UVIS Flight CCDs . . . 5
  - 2.1.3 CCD Data Readout and Calibration . . . 6
  - 2.1.4 Dark Frames . . . 8
  - 2.1.5 Dark Current . . . 8
  - 2.1.6 Bright Pixels . . . 8
- 2.2 Space Radiation Environment . . . 9
  - 2.2.1 Galactic Cosmic Rays (GCR) . . . 9
  - 2.2.2 Van Allen Belts . . . 9
  - 2.2.3 Solar Particle Events (SPEs) . . . 10
  - 2.2.4 Solar Wind . . . 11
- 2.3 Degradation Modes of CCDs . . . 11
  - 2.3.1 Ionising Damage . . . 11

2.3.2 Displacement Damage . . . 12
2.3.3 Single Event Effects (SEEs) . . . 13
2.4 Ground-Based Testing . . . 13
2.4.1 Standards for Testing . . . 14
2.4.2 Ground Testing for ExoMars TGO NOMAD-UVIS . . . 15
2.5 Ground Testing Evaluation . . . 17

3 Data Interpretation and Tool Utilisation 19
3.1 Computational Environment . . . 19
3.2 Dark Frame Dataset . . . 19
3.2.1 In-Flight Data . . . 19
3.2.2 Ground Data . . . 25

4 Image Processing and Analysis Methods 29
4.1 Bright Pixel Approach . . . 29
4.1.1 Frame Extraction . . . 29
4.1.2 Bright Pixel Characterisation . . . 33
4.1.3 Pivot to Dark Current . . . 37
4.2 Ground Test Data Approach . . . 38
4.2.1 Plotting Dark Current against Temperature . . . 38
4.2.2 Plotting Dark Current against Proton Fluence . . . 47
4.3 In-Flight Mission Data Approach . . . 53
4.3.1 Plotting Dark Current against Temperature over the Mission Duration . . . 57
4.3.2 Plotting Dark Current against Mission Duration . . . 58
4.4 Data Comparison Approach . . . 61

5 Data Characterisation and Findings 66
5.1 Ground Test Dark Current . . . 66
5.2 In-Flight Dark Current . . . 70
5.3 Direct comparison of Ground Vs In-flight . . . 72

6 Conclusions and Future Recommendations 78
6.1 Concluding Remarks . . . 78
6.2 Future Recommendations . . . 79

A Ethical Approval Letter 84

B Code for Frame Extraction 86

C Code for Bright Pixel Characterisation 89

D Plots for Bright Pixel Analysis 94

E Full Code For Ground Test Data 97

F Full Code section for In-Flight Data 106

G Incremental Time Analysis Fitted Figures **109**

H Code for Comparison Plot **119**

List of Figures

2.1 Teledyne e2v CCD30-11 used on NOMAD-UVIS. 6
2.2 Illustration of a CCD Array arrangement (Teledyne, 2020). 7
2.3 Diagram showing the ground testing process (ECSS, 2008). 15
2.4 UVIS Flight Model for Testing (Gow et al., 2017). 16
2.5 Diagram illustrating the Irradiation regions for the different devices (Gow et al., 2017). 17

3.1 Raw Dark Frame Plot. 22
3.2 Processed Dark Frame Plot. 22
3.3 Plot comparing a normal pixel to 3 bright pixels. 24
3.4 Half EOL Dark Frame Plot. 27
3.5 EOL Dark Frame Plot. 27
3.6 Double EOL Dark Frame Plot. 28

4.1 A figure to illustrate the PTC Plot for Half End of Life dose. 40
4.2 A figure to illustrate the PTC Plot for End of Life dose. 40
4.3 A figure to illustrate the PTC Plot for Double End of Life dose. 41

5.1 Ground Testing Arrhenius Dark Current Vs Temperature Plot with Linear Fits 67
5.2 Ground Testing Arrhenius Dark Current Vs Temperature Plot. 68
5.3 Ground Testing Dark Current Vs Proton Fluence 69
5.4 In-Flight Mission Dark Current Vs Temperature Plot 70
5.5 In-Flight Dark Current Vs Mission Duration Plot. 71
5.6 Combined Scatter Plot for In-Flight and Ground Data. 72
5.7 Enlarged ROI section of Combined Scatter Plot for In-Flight and Ground Data. 73
5.8 Cold Dark Frame for CSL Data 75
5.9 Hot Dark Frame for CSL Data 76
5.10 Electron variation due to noise fluctuation. 77

A.1 CURES Ethical Approval Letter 85

D.1 Mean Dark Current vs Temp plot with exponential fit. 95
D.2 Mean Dark Current over singular frame time histogram. 95
D.3 Pixel Noise over singular frame time histogram. 96
D.4 Pixel Noise Distribution over over a singular frame with the 3-Sigma Threshold applied 96

G.1 Dark current vs temperature plot from 0-0.5 years (fitted). 110
G.2 Dark current vs temperature plot from 1-1.5 years (fitted). 110
G.3 Dark current vs temperature plot from 1.5-3 years (fitted). 111
G.4 Dark current vs temperature plot from 2-2.5 years (fitted). 111
G.5 Dark current vs temperature plot from 2.5-3 years (fitted). 112
G.6 Dark current vs temperature plot from 3-3.5 years (fitted). 112
G.7 Dark current vs temperature plot from 3.5-4 years (fitted). 113
G.8 Dark current vs temperature plot from 4-4.5 years (fitted). 113
G.9 Dark current vs temperature plot from 4.5-5 years (fitted). 114
G.10 Dark current vs temperature plot from 5-5.5 years (fitted). 114
G.11 Dark current vs temperature plot from 5.5-6 years (fitted). 115
G.12 Dark current vs temperature plot from 6-6.5 years (fitted). 115
G.13 Dark current vs temperature plot from 6.5-7 years (fitted). 116
G.14 Dark current vs temperature plot from 7-7.5 years (fitted). 116
G.15 Dark current vs temperature plot from 7.5-8 years (fitted). 117
G.16 Dark current vs temperature plot from 8-8.5 years (fitted). 117
G.17 Dark current vs temperature plot from 8.5-9 years (fitted). 118

List of Tables

4.1 A table to show the calibration gains at their respective radiation doses. . 43

5.1 Results table for dark current at each radiation dose for ground data (at 0 °C). 74

5.2 Results table for dark current at each equivalent ground testing timestamps for in-flight data (at 0 °C). 74

Listings

3.1 Section of code for reshaping In-flight data with initial variables. 20
3.2 Section of code for plotting dark frames with and without bright pixels. . 21
3.3 Section of code for plotting 3 bright pixels and 1 normal pixels to show the difference in behavour. 23
3.4 Section of code for ground data structure. 25
3.5 Section of code for adjusting ROI. 28
4.1 Code for plotting the Temperature Vs Time. 29
4.2 Code for identifying frames close to the threshold temperature. 31
4.3 Code for extracting frames close to the threshold temperature and plotting the mean temperature variation. 32
4.4 Code for computing the mean dark current as well plotting mean dark current against temperature and a histogram for dark current against time. 33
4.5 Code for plotting two histograms; Pixel Noise against Time and Pixel Count against Pixel Noise. 35
4.6 Code for creating a binary mask and applying to data to produce dark frames. 36
4.7 Code for computation gain via Photon Transfer Curve (PTC). 42
4.8 Code for computation of Dark Current. 43
4.9 Code for computaion of Dark Current as a function of temperature for each radiation dose. 45
4.10 Code for Arrhenius Plot of Dark Current Vs Temperature. 46
4.11 Code for gain values as well as establishing the target temperatures and the desired dark current limit. 47
4.12 Code for conversion of proton fluences 48
4.13 Code for computing dark current and sorting temperatures. 48
4.14 Code for collecting dark current values at the target temperatures. 49
4.15 Code for producing dark current Vs proton fluence plot. 51
4.16 Code for looping through the desired temperature range. 53
4.17 Code for extracting frames at the desired temperature range. 54
4.18 Code for Plotting the In-Flight Dark current against temperature. 57
4.19 Code for filtering dark current and performing time increment analysis. . 59
4.20 Code for filtering dark current against mission duration. 61
4.21 Code for loading in both ground and in-flight data. 61
4.22 Code for pre-characterisation of ground data. 63
4.23 Code for producing combined scatter plot of both datasets. 64
B.1 Code for Frame Extraction and Storage 86
C.1 Code for applying 3-Sigma Threshold 89

E.1 Full Code for Ground Data. 97
F.1 Code for In-Flight Mission Dark Current Vs Temperature Plot 106
H.1 Code for producing the comparison plot of both datasets 119

List of Abbreviations

CCD	Charge-Coupled Device
ESA	European Space Agency
LEO	Low Earth Orbit
GCR	Galactic Cosmic Rays
SPE	Solar Particle Event
NASA	National Aeronautics and Space Administration
CME	Coronal Mass Ejection
CTI	Charge Transfer Inefficiency
NIEL	Non-Ionising Energy Loss
PKA	Primary Knock-On Atom
SEU	Single Event Upset
SET	Single Event Transient
SEL	Single Event Latch-up
SEGR	Single Event Gate Rupture
ECSS	European Cooperation for Space Standardization
EOL	End of Life
SPENVIS	SPace ENVironment Information System
ROI	Region of Interest
NLR	Non-Light Region
LR	Light Region
SNR	Signal-to-Noise Ratio
PTC	Photon Transfer Curve
DN	Digital Number

Chapter 1

Introduction

This chapter provides a preface to the thesis topic, capturing the motivations of the project and describing the main deliverables to be presented.

1.1 Background

Launched on the 14th of March 2016, the ExoMars Trace Gas Orbiter (TGO) was the first of two ExoMars missions. The second mission is yet to launch and is set to carry the Rosalind Franklin rover (scheduled for 2028)(ESA, 2025). Together, these two missions combine forces to answer the question of whether life on Mars has ever existed.

The ExoMars TGO hosts a suite of spectrometers onboard, which are otherwise known as the Nadir and Occultation for MArs Discovery - Ultraviolet and VISible light (NOMAD-UVIS). Through the utilisation of these NOMAD-UVIS spectrometers, the Martian atmosphere can be observed via the visible light and ultraviolet range, operating between the wavelength range of 200 and 650nm (ESA, 2025). More specifically, the NOMAD-UVIS instrument employs six Charge-Coupled Devices (CCDs)- a sensor used to convert photons of light into electrical signals, thus producing an image.

CCDs are particularly susceptible to radiation damage, hence sustained damage from the space environment leads to degradation of the captured image. The damage to the CCD is the primary focus of this paper.

1.2 Thesis Aim and Objectives

This section outlines the main aim of the thesis, thus highlighting the purpose of the thesis and its relevance. The objectives are then stated and expanded to portray how the criterion set out by the aim is achieved.

1.2.1 Thesis Aim

The overall aim of this thesis is to evaluate the extent of radiation-induced damage to the NOMAD-UVIS spectrometer during spaceflight via both mission data and ground testing data, to assess whether the methods of irradiation on ground are representative of flight performance.

1.2.2 Thesis Objectives

To fulfil the aim of this thesis, four main objectives were defined, following a logical and systematic progression

1. Extended study to deepen understanding of the space radiation environment.
    (a) Assess what space radiation is and how it affects the ExoMars NOMAD-UVIS spectrometer.
    (b) Research into ground-based irradiation testing, exploring multiple facets.
    (c) Assess what the generated data is and how it is typically interpreted.
2. Contextualise and analyse the dark image data.
    (a) Understand the presented format of data output and how it is visually represented.
    (b) Develop MATLAB code to interpret the dark images and generate plots for comparison.
    (c) Analyse plots and data for any explicit features.
3. Establish the effectiveness of ground-based dark images in comparison to real mission-based dark images.
    (a) Identify features of interest within the analysed data.
    (b) Utilise these features to develop figures of merit that enable the dose of ground testing irradiation to be related to mission radiation exposure.
    (c) Assess whether said relationship is under or overestimating the mission radiation dose.
4. Draw conclusions on the effectiveness of ground-based irradiation testing in comparison to mission irradiation damage.
    (a) Identify possible patterns or trends related to the bright pixels/dark current.

(b) Perform a behavioural analysis, looking at how the damage occurs over the period of irradiation.

(c) Identify any relations between bright pixels/dark current and figures of merit.

1.3 Thesis Composition

This thesis comprises of six chapters, with Chapter One being this introductory chapter.

Chapter Two provides the scientific and technical background necessary for this study. It introduces the ExoMars TGO mission, explains the fundamentals of Charge Coupled Devices (CCDs) and their operating principles. The main format of data under investigation is also briefly described (dark frames). It then examines dark current and bright pixels, the space radiation environment, and CCD degradation modes. The chapter concludes with a discussion of ground-based irradiation testing and its expected outcomes.

Chapter Three provides an overview of the analysis tools employed throughout this study, and present the initial datasets. The raw data is then characterised with its relevance explained, setting the context for the methodology and results in later chapters.

Chapter Four details the methodology, outlining the processes, algorithms, and data-handling techniques that were used to develop the figures and plots necessary for data comparison.

Chapter Five presents the results of the study. Firstly, it analyses the ground-test data and in-flight data individually, before providing a direct comparison between the two datasets, highlighting similarities, differences, and key trends exhibited throughout the analysis.

Chapter Six combines all of the factors analysed and the derived results to provide a summary of the findings within this study. It discusses the limitations encountered, outlines issues that arose during the study analysis, and provides recommendations for future work.

Chapter 2

ExoMars and the Space Radiation Environment

This chapter presents a comprehensive overview of the ExoMars mission, outlining its primary objectives alongside the methods of data capture and calibration. Moreover, it details the issue space radiation damage, addressing the underlying causes, associated effects and implications. In addition to this, it further investigates the process of ground-based irradiation testing and the expected outcomes in comparison to in-orbit exposure.

2.1 ExoMars TGO

Building upon Section 1.1, the ExoMars (Exobiology on Mars) mission seeks to investigate the signs of life on Mars, looking at what it may have been like previously and what it is at present. The mission itself is a collaboration between the European Space Agency (ESA) and the Russian Space Agency (ROSCOSMOS) (Vandaele et al., 2015). The ExoMars TGO spacecraft houses four scientific payloads: NOMAD - Nadir and Occultation for MArs Discovery, ACS - Atmospheric Chemistry Suite, CaSSIS - Colour and Stereo Surface Imaging System and FREND - Fine Resolution Epithermal Neutron Detector (ESA, 2016). The focus of this paper will be on the NOMAD-UVIS spectrometer suite as this the main imaging device used onboard.

2.1.1 The ExoMars Mission

ExoMars TGO was launched in March 2016 and arrived at Mars in October of the same year. A period of aerobraking was then pursued in order to achieve the final orbit and speed, which was reached in April 2018. With a final orbit altitude of 400 km above the surface of Mars and an inclination, ExoMars TGO is able to complete an orbit in 2 hours

(120 minutes). Each orbit begins with a sunrise occultation measurement, then a 1-hour dayside nadir observation. This is then succeeded by a sunrise occultation measurement after which the final hour is spent on the night side of Mars, capturing dark frames before the orbit cycle begins again. It was only from this point onwards that the science mission commenced, thus leaving a two-year delay before data could be collected by the NOMAD-UVIS spectrometer suite (Willame et al., 2022). The nominal mission lifetime of the ExoMars TGO mission is until 2022 (four years from start of data collection) (ESA, 2016), which is the equivalent of two years in orbit around Mars. However, ExoMars is still in orbit as of 2025. The NOMAD-UVIS spectrometer suite set out to achieve 3 main objectives (Vandaele et al., 2015):

1. Analyse the chemical composition of the Martian atmosphere via trace gases and isotopes.

2. Characterisation of seasonal cycles and variability in climate in and around Mars.

3. Investigation of the production and loss of trace gases within the Martian atmosphere.

As per the abbreviation of NOMAD-UVIS, UVIS consists of two telescopes, one telescope is positioned to point towards and observe the surface of Mars (nadir), whilst the other faces and observes the Sun (solar occultation). The nadir telescope is also capable of performing observations from a perspective tangential to the atmosphere (limb measurements), which allow the measurement of light emission and scattering at various altitudes (Willame et al., 2022).

2.1.2 Characterisation of the NOMAD-UVIS Flight CCDs

As mentioned in Section 1.1, NOMAD-UVIS is comprised of six CCDs. Each of these CCDs are made up of pixels that are highly sensitive to incident photons. When a photon of light comes into contact with the CCDs effective area, it will be converted into one (or more) electrons (photoelectric effect - as first discussed in Einstein's 'Annus Mirabilis' papers) (Arons & Peppard, 1965). The number of electrons is directly proportional to the intensity of the incoming light, thus enabling the CCD to produce quantitative values that represent the intensity of light across the array of pixels on the surface (Teledyne, 2020). The CCD model used onboard NOMAD-UVIS can be seen in 2.1.

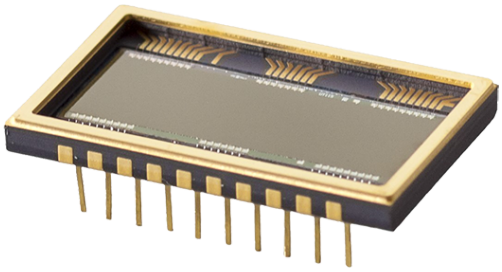


Figure 2.1: Teledyne e2v CCD30-11 used on NOMAD-UVIS.

2.1.3 CCD Data Readout and Calibration

A CCD layout is divided into multiple gates (or electrodes). CCD arrays can be either one-dimensional (a singular row) or two-dimensional (multiple rows and columns). The NOMAD-UVIS suite uses six two-dimensional array CCDs. The array size and the number of gates on the CCD share a directly proportional relationship; hence, the larger the CCD array, the more gates present on the surface of the CCD. Underneath these gates, the charge is collected (groups of electrons) in a buried channel region (sometimes called a "well"). Charge is only collected in the presence of the potential well, which is generated by a potential present on the gate. The gates run in a horizontal orientation across the device and then are broken up into columns by a channel stop. The layout is as such to allow for the charge to be transferred downwards one row at a time (column-wise shifting) during the process of clocking (movement of electrical charge). One row is clocked at a time (column-wise shifting) until the charge has reached the readout register (bottom row) where it is now shifted horizontally along the row (row-wise shifting) towards the output node. At the output node, the charge is then converted to a voltage. The clocking process is then repeated for the next row, and the next until all rows have been shifted to the bottom and read out (Teledyne, 2020). Upon completion of all rows being read out, a raw CCD frame is produced. An illustration of the CCD array arrangement can be seen in Figure 2.2.

In order to assess the extent of the damage done to the CCD, calibration must be performed to the raw frames, whereby the damaged sections of the frame can be masked to produce processed frames. There are two measurements used for calibration of frames; offset bias and dark current. Both measurements are taken with zero exposure (no light reaching the spectrometer or CCD) to allow for detection of damaged pixels. Offset bias measures the random fluctuations of noise coming from the components of the CCD itself during clocking (read noise). Dark current measures the accumulation of back-

ground charge present on the CCD, in the absence of light (Willame et al., 2022). These two measurements are absolutely essential for calibration of the raw frames; however, to achieve complete calibration, multiple other processes must be carried out before the frame can be used for scientific analysis.

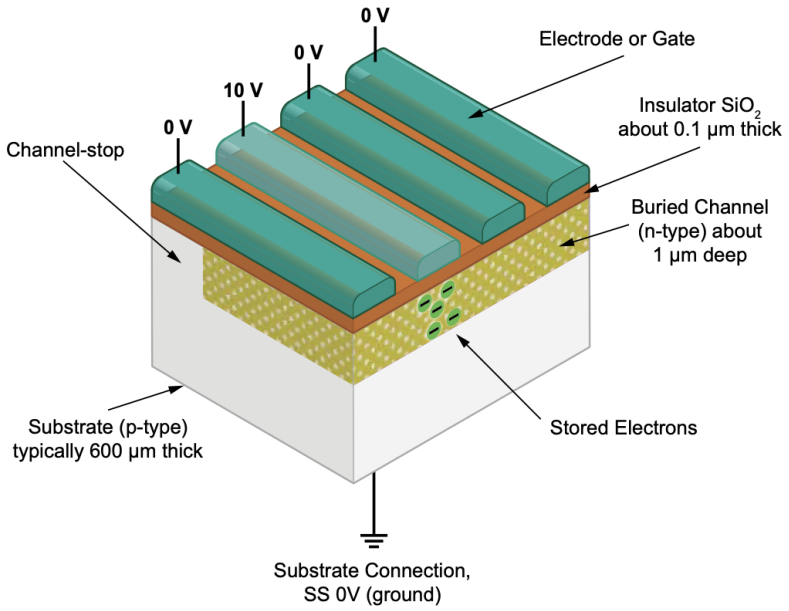


Figure 2.2: Illustration of a CCD Array arrangement (Teledyne, 2020).

In order to attain complete calibration of the frames produced by NOMAD-UVIS, there are a few steps that must be followed (Willame et al., 2022):

1. Correct for non-linearity
2. Offset Bias removal
3. Dark Current Removal
4. Bad Pixel Identification
5. Removal of smearing
6. Assigning specific Pixels to wavelengths
7. Stray light removal
8. Vertical Binning
9. Radiance Conversion

Although CCD data calibration follows the process outlined in the steps above, this thesis concentrates more on the characterisation and reduction of dark current interference using the dark frames. Other procedures, such as the removal of smearing and stray light removal, are beyond the scope of this work.

2.1.4 Dark Frames

A dark frame is an image captured by the CCD with no exposure to light; a visualisation of the dark current measurements as introduced in subsection 2.1.3. In context of the ExoMars TGO mission, the dark frames were initially captured via the use of the nadir channel, but due to concerns about the lifetime of the selector mechanism, this was stopped (Willame et al., 2022). Aside from this selector mechanism, there is no physical mechanism alternative that can stop the light from reaching the CCD. As a result of this, the only alternative dark images are taken during the night cycle of the Martian orbit (1-hour duration), whereby the camera is facing the dark side of Mars. The number of frames taken during this 1-hour duration is solely dependent on the integration time. Integration time varies based on type of measurement being taken. Dark frames from Nomad-UVIS use a specific integration time of 15 s; hence, at this integration time there will be 240 dark frames taken during the night cycle.

2.1.5 Dark Current

Despite the lack of light exposure, electrons are still thermally excited across the silicon surface bandgap whereby they are collected in the 'potential wells' during the integration periods. This is known as dark current, and is visually represented on a dark frame as noise. It is an inherent characteristic of dark frame capture and hence is subtracted to provide the fully calibrated frame. Essentially, each pixel of the CCD will have the photoelectrons from daytime exposure and then a separate set of electrons gathered during no exposure. The amount of dark current generated is dependant on factors such as the environmental temperature and the integration(exposure) time. A higher temperature results in a rise in dark current (Widenhorn et al., 2002). The same effect can be observed due to an extended integration time.

2.1.6 Bright Pixels

A dark frame is made up of pixels sorted into rows and columns (as per the format of the CCD). Usually, there is little to no variation in the pixel value during a dark frame; however, if a pixel happens to consistently read a higher value, then it is referred to as a bright

pixel (or hot pixel). As of current, there is no specific value or threshold to identify what qualifies as a bright pixel. This lack of definition is due to factors such as the variation in temperature and change in exposure time. Another contributing factor is pixel annealing. Pixel annealing is the process by which damaged pixels are able to partially or fully return to normal behaviour (heal), due to a change in temperature (thermal cycling) (Polidan et al., 2004). Over the life of the mission, ExoMars TGO is exposed to more accumulated radiation, hence the number of bright pixel occurrences will increase, representing the damage done to the CCD inline with the increased radiation dose.

2.2 Space Radiation Environment

Upon departure from Low Earth Orbit (LEO) (above 2000 km), spacecraft are exposed to heightened levels of ionising and non-ionising radiation; inherent to deep space environment. The cause of this intense radiation in deep space can be attributed to 3 main sources, Galactic Cosmic rays (GCR), Van Allen Belts (otherwise known as Earth's Radiation Belts), and Solar Particle Events (Benton & Benton, 2001).

2.2.1 Galactic Cosmic Rays (GCR)

Galactic Cosmic Rays (GCR) are high-energy charged particles (usually protons) that originate from beyond our solar system (Benton & Benton, 2001) and possess energy levels of 1000 MeV for protons and a range of 500 to 600 MeV for heavier ions (Sihver & Mortazavi, 2019). Within the constraints of LEO, the strongest dose of GCR is present at the Earth's poles; however, the dose is at a minimum upon close proximity to the equator.

Due to its close proximity to Earth, LEO has been extensively studied, providing a large number of findings. However, as outlined in Section 1.1, the primary focus of the ExoMars TGO mission is to assess the Martian atmosphere. Findings from the 2001 Odyssey Martian radiation environment experiment (MARIE) is indicative of the Martian atmosphere displaying radiation doses approximately 2.5 times greater than that of LEO, with GCR accounting for 70-90% of said dose - on average over mission duration (NASA, 2003).Consequently, due to being such a large contributor to space radiation around the Martian atmosphere, it is a vital aspect of this study.

2.2.2 Van Allen Belts

Due to the Earth's magnetic field, certain particles (protons and electrons) are trapped within specific regions surrounding the Earth. Both protons and electrons have their own

belts, but the collective region is referred to as the Van Allen Belts, and is often characterised as "Doughnut-Shaped" due to the structure formed by the inner and outer radiation belts surrounding the Earth (Farley, 1964). Proton belts consist of protons with energies in excess of 7 MeV and are limited to 700 MeV, whilst electron belts consist of electrons that are approximately 0.5 MeV or greater, with rare occurrences of electrons >1 MeV (Reeves et al., 2020). By comparison of energy levels, the magnitude of the Van Allen Belts are far inferior to that of GCR.

Many other planets also have their own Van Allen Belts; however, it is to be noted that Mars is not one of these planets due to the lack of a magnetic field. Thus, for the CCDs onboard the NOMAD UVIS Spectrometers, the Van Allen Belts are not a contributing factor to the radiation damage whilst in the Martian orbit. However, they likely contributed during the Earth-Mars transfer period, as the ExoMars TGO had to pass through the Van Allen Belts upon escaping Earth.

2.2.3 Solar Particle Events (SPEs)

Solar Particle Events (SPE) take place when the sun emits a large amount of protons which either are accelerated close to the sun's radius (solar flare), or in some cases interplanetary. Interplanetary emission is caused by significantly large ejections of plasma (consisting of protons, electrons and alpha particles), otherwise known as coronal mass ejections (CMEs). The energy levels of protons emitted during such large solar events are upwards of 30 MeV (Hu, 2017), with the energy of alpha particles being even higher. To understand the extent of radiation damage caused by these events, it is essential to further examine solar activity through the analysis of the solar cycle.

The sun's magnetic field behaves in a cyclical nature over the period of 11 years. This is better known as the solar cycle. The beginning of the cycle is known as the solar minimum; this is the period of time where solar activity is at its lowest. Solar activity gradually rises over the duration of a few years and approaches a solar maximum, whereby the solar activity is at its peak. Activity then decreases over the remaining cycle duration, until solar minimum is reached and the cycle starts again. During the solar minimum period, there tends to be a CME less than once a day (~ 0.5), however, when very active (solar maximum period), the frequency considerably increases to upwards of six CMEs per day (Fleck et al., 2006). Through this direct correlation between solar activity and CMEs, it is evident that the radiation dose will vary based on the frequency of CME occurrences.

2.2.4 Solar Wind

Travelling at speeds in excess of 3 million km/h, protons and electrons flow from the sun into the surrounding space (Hu, 2017). This stream of particles is known as solar wind. The radiation dose received from solar wind damage varies with time due to it being dependant on the 11-year solar cycle (Ahmed, 2022; Hathaway, 2015). The protons from the solar wind collide with the surface of the CCD and produces electron traps (Prod'homme et al., 2012); a displacement damage mechanism mentioned in Section 2.3. The resulting factor from this damage is an increase in charge transfer inefficiency (CTI), which leads to a noisier image. Currently, the Martian atmosphere provides a significant level of protection for ExoMars TGO as it is able to shield from the solar wind. Nonetheless, according to the National Aeronautics and Space Administration (NASA) this will not always be the case for missions within the Martian atmosphere, as the solar wind constantly erodes the atmosphere, leading to approximately a quarter of a pound (~ 0.11 kg) of atmosphere lost per second (NASA, 2024).

2.3 Degradation Modes of CCDs

Upon exposure to the space radiation environment, high-energy protons, electrons, and heavy ions interact with the silicon lattice of the CCD, thus leading to damage, resulting in a deterioration in performance. The types of damage caused can be attributed to two degradation modes: ionising damage and displacement damage.

2.3.1 Ionising Damage

Ionising damage results from ionising radiation interacting with the CCDs onboard the NOMAD UVIS module. There are two main mechanisms via which the damage is caused, as the incident particle travels through the CCD, a trail of electron-hole pairs are created which lead to long-term hole trapping and the creation of Si-SiO_2 surface states (Dryer, 2013).

When photons collides with the CCD, electron hole pairs are formed. The electrons are collected in the potential well, whilst the hole usually migrates in the opposite direction (usually to the Si-SiO_2 interface layer) or recombines quickly with the electron (Dryer, 2013). Upon return to the substrate, the holes may encounter a trap defect and become lodged there. These holes accumulate within the trap defects over time and begin to act as stable charge (permanent positive charge), which alters how the CCD behaves; the threshold voltage changes, thus reducing the amount of charge that can be stored (flat-band voltage shift). The overall resulting factors are an increase in dark current and a

reduction in charge transfer efficiency (CTI) produced. This mechanism is referred to as long-term hole trapping.

The other mechanism aforementioned is the creation of Si-SiO_2 surface states. At the Si-SiO_2 interface, there are manufacturing defects present, some atoms have incomplete bonds(less than four bonds). Within these incomplete bonds are unpaired electrons called dangling bonds. Dangling bonds serve as an electrically active trap; hence, charge is captured, much in the same manner as long-term hole trapping. This leads to a further increase in dark current.

2.3.2 Displacement Damage

Displacement damage stems from high-energy protons and neutrons interacting with the silicon atoms within the onboard CCDs. Upon interaction, the silicon atoms are displaced from their crystal lattice position. The consequential effects of such interactions are not as straightforward as ionising damage. Instead, the effect of displacement damage will depend on various factors such as the particle type, energy of the particle and CCD properties (such as impurity levels) (Dryer, 2013). The energy lost by particle collision is the particle's Non-Ionising Energy Loss (NIEL). By measuring the energy lost, the amount of damage caused by a specific particle can be quantified.

As the high-energy particles interact with the silicon atoms, an amount of energy is transferred to the atoms by means of an elastic or inelastic collision. As previously mentioned, the collision results in the silicon atoms being displaced and released from their lattice position, which gives rise to a vacancy (gap in the lattice). Once released, these atoms can then go on to further displace other atoms in the silicon, leaving the CCD silicon surface with more defects. Due to this effect, the released atoms are referred to as the primary knock-on atom (PKA). In order to be released, a threshold energy level must be met ($\sim$ 20 eV for silicon) (Kerbiriou et al., 2007). The PKA comes to rest in the space between the regular lattice sites. This is known as the 'interstitial'. The combination of a a vacancy and an interstitial are better known as a Frenkel pair. In most cases, these Frenkel pairs are unstable, however, in some cases they interact with other nearby vacancies or impurities to form bulk traps. Depending on the bulk trap, they can also become trap defects which have an adverse effect on the CCD, as the Charge Transfer Inefficiency (CTI) and dark current increases.

CTI is a metric of performance used to measure the amount of charge lost from transfer of charge between pixels. CTI has various effects on the CCD readout, such as an increase in noise levels, image smearing (causes trails on dark frames), loss of signal from affected pixels. Ultimately, CTI is the limiting factor of performance for the CCD, hence CTI and performance of the CCD share an inversely proportional relationship.

2.3.3 Single Event Effects (SEEs)

Unlike ionising and displacement damage, Single Event Effects (SEEs) are instantaneous and do not accumulate damage over time. SEEs occur due to a singular high-energy particles interacting with the surface of the CCD. In the process of interaction, they deposit enough energy to overload sensitive electronics components and structures (Dryer, 2013). There are three main types of SEEs: Single Event Upset (SEU), Single Event Transient (SET) and Single Event Latch-up (SEL). Single Event Gate Rupture (SEGR) is also observed, however, considering that ExoMars TGO is orbiting the Martian atmosphere, it is a much rarer phenomenon.

SEUs occur due to particles generating enough charge to trigger a 'bit flip'; a change of logic from a 0 to 1 or vice versa, which switches the voltage parameter from low to high or high to low. Depending on the data stored within that pixel, the extent of the damage can vary, but they are typically seen as the least damaging SEE. SETs arise from anomalous events not intended to be measured, such as cosmic rays colliding with the CCD or noise from onboard instrumentation interfering with the pixel values. This results in false pixel values upon readout, which can affect the accuracy of the data received. SELs deposit a significant amount of charge to the CCD, to a degree that the charge disrupts the entire CCD and creates a low resistance path between the ground and the power supply (Sexton, 2003). This leads to the device drawing very large amounts of current continuously (even after the particle/trigger has disappeared). This can lead to high localised temperatures or even component burnout which in turn means permanent loss of CCD functionality. Hence, for this ExoMars mission, and any space imaging mission, SELs tend to be of most concern.

2.4 Ground-Based Testing

Ground-Based irradiation testing is the process of exposing the CCD to a radiation environment that replicates the conditions it will encounter in space. By carrying out tests of this nature, the radiation's impact on CCD performance and thus the science performance of the overall instrument can be assessed (Gow et al., 2017). Testing is usually conducted over a much shorter period of time than the expected mission duration; however, the doses that the CCD exposed to are modelled to be as close to, if not the same as the dose the CCD and instrumentation will receive in flight.

2.4.1 Standards for Testing

All ground testing is done in conformance with ECSS-Q-ST-70-06C: 'Particle and UV radiation testing for space materials'(ECSS, 2008), which is a document put into place by the European Cooperation for Space Standardization (ECSS) that outlines requirements that must be met during the testing process. This document stipulates requirements for developing a methodology that allows for simulation of the space radiation under controlled lab conditions, ensuring the testing process and results are reproducible, consistent and traceable, whilst confirming to the acceptance criteria. Some of the key requirements specified in ECSS-Q-ST-70-06C are as follows:

- 'When ionizing radiations are used, ISO 15856:2003 clause 6.3.2 shall apply for methodology "in case on dose dependant properties".' (Section 5.1.3.1).
    1. 'The methodology for ionizing radiations on dose dependant properties replaces the simulation of space environment charged particles or photons by the simulation of the absorbed dose profile through a material thickness that depends on the investigated physical property.'
    2. 'When possible, simulate each part of the dose profile using the type of prevailing ionizing radiation to which the material would be exposed in the natural space environment (e.g. protons at the surface of the materials in GEO).'
- 'Samples shall be protected from exposure to light and kept at ambient temperature of (22 ± 3) °C, and with a relative humidity of (55 ± 10)% .' (Section 5.2.2)
- 'The ambient conditions for the process and work areas shall be (22 ± 3) °C with a relative humidity of (55 ± 10)% .' (Section 5.2.3)

These are just a few of many requirements outlined in ECSS-Q-ST-70-06C for conducting an effective ground test. The full process can be seen in Figure 2.3. In conjunction with this document, there are other ECSS documents that delve into further detail about specific sub-sections of requirements. Examples of this being ECSS-E-ST-10-04C Rev.1 (Space Environment), ECSS-Q-ST-20C (Quality Assurance) and ISO 15856:2010 (Simulation guidelines for radiation exposure of non-metallic materials) (ECSS, 2018, 2020; ISO, 2010). ECSS-E-ST-10-04C Rev.1 delves into the specifics of the space environment with a specific section on 'Energetic Particle Radiation'. Within this section, the requirements for modelling of solar particle events and cosmic rays are defined. ECSS-Q-ST-20C outlines requirements with regard to the test facilities used, test equipment used and the criteria for documentation prior to and post testing. ISO 15856:2010 dictates requirements on matters such as how the space radiation environment should be recreated

and sources of radiation that can be used for simulation (such as low/high energy protons and and UV). The presence of standards for ground testing proves advantageous as they produce consistent, reproducible results that can be verified by meeting the criteria thus leading to the devices qualifying for flight use.

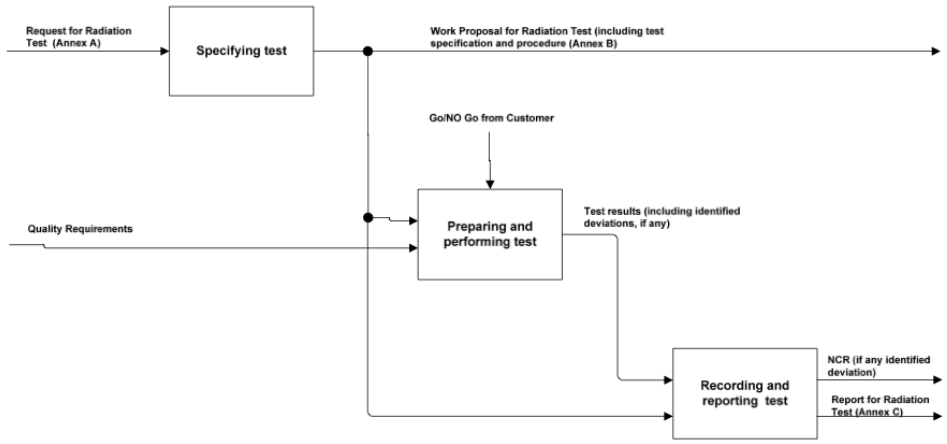


Figure 2.3: Diagram showing the ground testing process (ECSS, 2008).

2.4.2 Ground Testing for ExoMars TGO NOMAD-UVIS

To compliment the ground testing conducted for the CCDs in NOMAD-UVIS, the study 'Proton Radiation Damage Assessment of a CCD for use in a Ultraviolet and Visible Spectrometer' was provided (Gow et al., 2017), which was a study run by the Open University. This paper details the experimental setup, CCD parameters, radiation doses, figures of merit, results, and conclusions. The performance metrics under analysis within this study are dark current and CTI.

The CCD under testing here is the CCD30-11 manufactured by Teledyne e2V. Due to NOMAD-UVIS using a single spectrometer and two telescopes, a single CCD30-11 is used for detection. The CCD itself uses a back illumination and has a Ultra Violet (UV) coating to enhance the sensitivity to UV upon detection. In combination with a grating, UVIS is able to achieve a spectral range of 200 - 650 nm with a resolution ranging between 1.2 nm (at shortest wavelengths) and 1.7 nm (at longest wavelengths) (Willame et al., 2022). Moreover, CCD30-11 boasts an image area of 26.6 x 6.67 mm. Within this image area CCD30-11 has 1024 (horizontal) and 256 (vertical) active pixels; each of these pixels being 26 x 26 μm, with both the under-scan and over-scan values being equal to 8 pixels (Gow et al., 2017; Teledyne, 2024). For the test to truly replicate exposure within the space radiation environment, a model of the UVIS flight model had to be made for the CCD to be mounted to during the irradiation. This can be seen in Figure 2.4.

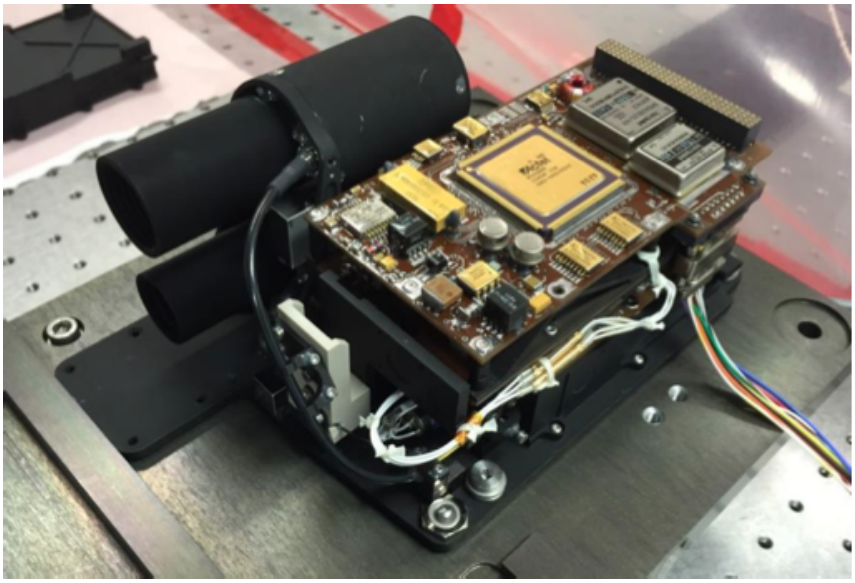


Figure 2.4: UVIS Flight Model for Testing (Gow et al., 2017).

To estimate the End of Life (EOL) radiation exposure, a European Space Agency tool call SPace ENVironment Information System (SPENVIS) was used. SPENVIS is a software used to model and analyse the space radiation environment, in addition to its effects on spacecraft. From this, the EOL proton fluence and the ionising dose were estimated. Additionally, as prescribed by ECSS-E-ST-10-04C (ECSS, 2020), the analysis was done in accordance with the Emission of Solar Protons (ESP) model (Xapsos et al., 1999), whereby the spacecraft being held at 1 AU during solar maximum must be taken into account. This overestimates the proton fluence and assumes the worst-case scenario thus allowing for a safety margin upon flight. Applying these models, the EOL dose was estimated to be 1.9 krad with a 10 MeV equivalent proton fluence of 4.7 x 10^9 protons.cm^{-2}. In the actual experiment, 4.8 x 10^9 protons.cm^{-2} was used instead. Based upon this, it was decided to run the tests at 3 separate radiation doses: half the end-of-life dose (3.6 x 10^9 protons.cm^{-2}), end-of-life (4.8 x 10^9 protons.cm^{-2}) and double the end-of-life dose (9.4 x 10^9 protons.cm^{-2}).

Four CCDs were used for this study, with three being the subject of proton irradiation, whilst a singular CCD was used as a control to ensure a baseline reference for all values. On each device, only 50% of the image area was irradiated. Devices 1 and 3 were irradiated only on the right-hand side of image area whilst the left-hand side of the image area was covered with a radiation shield. Device 2 is the exact opposite, with the left-hand side of the image area being exposed whilst the right-hand side of the image area was covered with a radiation shield. The non-irradiated section of the CCD could be used to calibrate the radiation data from the same device due to being exposed to the same temperature for the same amount of time. This setup is illustrated in Figure 2.5.

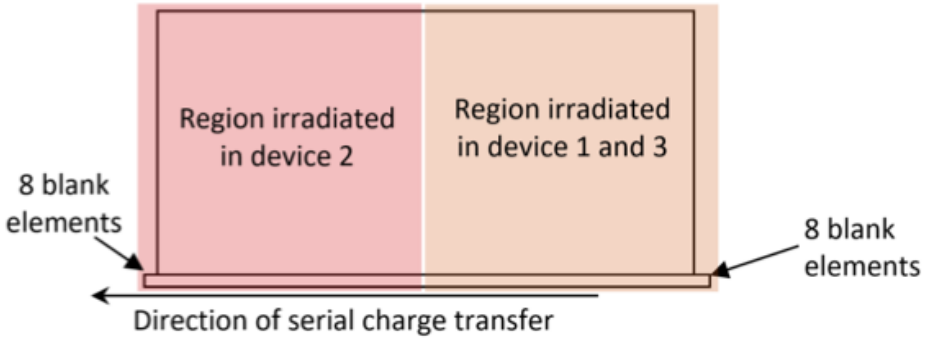


Figure 2.5: Diagram illustrating the Irradiation regions for the different devices (Gow et al., 2017).

In line with predictions concerning the Martian climate, temperatures were chosen to be within the range -30°C to +20°C (Gow et al., 2017), with the exact temperatures being -30°C, -20°C, -10°C, 0°C, +10°C and +20°C. Additionally, integration times were varied depending on the measurement being taken. 30 s to 160 s were used for nominal measurements, whilst substrate voltage (V*ss*) measurements used 10 s for 3 V - 7.5 V and 5 s for V*ss* below 3.0 V.

2.5 Ground Testing Evaluation

Ground Testing radiation damage campaigns are common to space missions, with damage assessment on CCD only being relevant for the space imaging missions. Missions such as Gaia, Euclid and SMILE are just a handful of other missions that have conducted ground testing pre-flight.

Gaia, a mission for tracking the motions and positions of stars, was launched in 2013 an ended operation in 2025. Its primary objective was to delve into the origin of our galaxy via observing comets, asteroids and even other galaxies from the L2 Lagrange point (Seabroke et al., 2008). Pre-flight, the CCDs underwent five radiation campaigns to test the device performance. The set-up used for Gaia is different from ExoMars TGO, as instead of one CCD being used for in-flight measurements, multiple were used in an array thus forming a focal plane. The dose of protons used for experimentation was a predicted value of 3.1 × 109 p/cm ^{2} using 10 MeV equivalent protons. This estimated value came from a simulation during 2006, a time of solar maximum. This testing neglected consideration towards galactic cosmic rays, but did do testing at an overestimated dose of 4 × 109 p/cm ^{2}. Upon comparison of ground testing with Gaia's mission data, it was seen that the overall accumulated damage was less than predicted (Ahmed et al., 2022).

Euclid's main aim was to study and map dark energy and dark matter within the uni-

verse using visible and near-infrared imaging. It was launched in 2023 and had a nominal mission duration of six years. The Engineering Models (EM) are tested both warm and cold. It was concluded that upon testing, the devices performed well; however, during cold testing, non-uniformities within the silicon and other manufacturing defects were identified which altered the performance metrics of the device (Szafraniec et al., 2016). Additionally, a new operational mode was implemented for this mission called trap pumping, whereby the charge is released from the trap defects (Skottfelt et al., 2024). This in turn mitigates the effects of CTI within the CCD (Gow & Murray, 2017). Another operational mode developed from this mission was charge injection, whereby rows of charge are injected into the CCD and fill up the traps before exposure. This further alleviates the effect of CTI. In addition to pre-flight ground testing for Euclid, simulations were used to estimate densities and energy levels to reproduce the experimental data to great accuracy (Skottfelt et al., 2016).

As of this study, the Solar Wind Magnetosphere Ionosphere Link Explorer (SMILE) mission has not yet been launched. Planning to launch in 2026, the aim of the mission is to develop understanding of how the solar weather interacts with the Earth's magnetosphere-ionosphere via four different instruments (Carter et al., 2025). As the mission has not yet launched, ground testing is still commencing; however, some has been carried out to date. SMILE CCDs ground testing was conducted in cryogenic conditions with temperatures ranging from -85 °C to -135 °C. Upon an incremental increase in temperature from 135 °C to -85 °C, no detectable impact was observed in the results. Additionally, via the mechanism of optical loading, the CCD trap defects can be pre-filled and were seen to reduce the parallel CTI by 10 - 15% at temperatures below -110°C. When using charge injection at the same temperature range (the mechanism first used on Euclid), the CTI decreased by a further 10 - 20% (Parsons et al., 2020).

Drawing on these missions, it is evident that ground testing has limitations in areas such as manufacturing defects and estimation of radiation doses. Similar issues are present within the ground testing process of ExoMars TGO mission. The full assessment of ground testing for ExoMars will be detailed in Chapter 5.

Chapter 3

Data Interpretation and Tool Utilisation

This chapter will provide an outline of the dataset provided and how it is stored, in addition to what this data looks like within the MATLAB environment, and how it can be manipulated to produce results.

3.1 Computational Environment

In order to perform any analysis of dark frames, the frames had to be loaded into separate software for analysis. The choice of software for this study was MATLAB, chosen for its ability to handle arrays, ease of plotting tools, and environment, as well as the ability to produce higher quality figures and plots. Furthermore, MATLAB is designed and optimised specifically to work with arrays; hence, its usage within the code is more accessible for use with the datasets provided for this study.

3.2 Dark Frame Dataset

Both the ground irradiation testing data and the in-flight mission datasets were provided by the Open University. The flight data and ground data were stored in separate HDF5 files due to their large file sizes. Through the use of MATLAB, the files can be read in and manipulated to extract the desired results.

3.2.1 In-Flight Data

The in-flight data contained a culmination of the raw frames from the ExoMars TGO mission over the past seven years. Corresponding metadata was also provided: frame time, frame number, exposure time, frame temperature (in °C), processed frames, and a bright pixel map for each orbit. Each of these variables has an array of values corresponding to

frame number, for example, 'Frame 1' will have an associated frame time, temperature, and exposure time.

Once loaded into MATLAB, the data was not in the correct order and needed to be reshaped and sorted. The best approach to this was to first correct the orientation of the rows and columns so that the frames show up correctly, and then, secondly, sort the frames by time (in ascending order), thus providing a chronological overview of the mission via frames. The section of code in Listing 3.1 outlines the code used to perform these corrections. This subset of code was provided by the Open University (Mason, 2025).

```
fnm = 'UVIS_data_subset_forJames.h5';  %% Load the UVIS data from
    the HDF5 file
FrameNumber = h5read(fnm,'/FrameNumber');  % Frame number
FrameTime = h5read(fnm,'/FrameTime');    % Frame time in seconds
FrameTemp = h5read(fnm,'/FrameTemp');  % Frame temperature in
    degrees C
Integration = h5read(fnm,'/Integration'); % Exposure time
w_RawFrames  = h5read(fnm,'/RawFrames');  % Raw' frames across all
    orbits (contains bright pixels)
w_BrightPixelMap  = h5read(fnm,'/BrightPixelMap');  % The
    brightpixel map for each orbit
w_ProcessedFrames  = h5read(fnm,'/ProcessedFrames'); % Processed
    frames across all orbits (bright pixels removed)

% Going between Python and Matlab transposes the row and column
    dimensions - this function corrects for this.
[RawFrames,ProcessedFrames,BrightPixelMap] = reshape_frame_data(
    w_RawFrames,w_ProcessedFrames,w_BrightPixelMap);
[Time,isort] = sort(FrameTime,'ascend');
FrameNumber = (1:size(RawFrames,3));

function [rdat1,rdat2,rdat3] = reshape_frame_data(dat1,dat2,dat3)
    [nx,ny,nz] = size(dat1);
    rdat1 = zeros(ny,nx,nz);
    rdat2 = zeros(ny,nx,nz);
    rdat3 = zeros(ny,nx,size(dat3,3));
    for i = 1:nz
        rdat1(:,:,i) = dat1(:,:,i)'; % transpose the array
        rdat2(:,:,i) = dat2(:,:,i)'; % transpose the array
        if i <= size(dat3,3)
            rdat3(:,:,i) = dat3(:,:,i)';
        end
    end
end
```

Listing 3.1: Section of code for reshaping In-flight data with initial variables.

Using a condensed version of the full in-flight dataset, an exemplar raw frame and processed frame were plotted using the '**imagesc**' function within MATLAB, which allows matrix data to be displayed as a scaled image, and each of the values in the matrix is mapped to specific colours depending on their value. This proves advantageous from a scientific analysis perspective, as the damaged pixels can be identified much more easily, in contrast to searching the metadata to find a specific row and column of a pixel that may be damaged. Additionally, using a visual representation of the damage allows for easier identification of bright pixels or even anomalies, such as cosmic rays, could be present on the dark frame. The output of '**imagesc**' for both the raw and processed frames can be seen in Figures 3.1 and 3.2, respectively, with the code used to produce these dark frames in Listing 3.2 (Mason, 2025).

```
%% plot the dark frame with Bright pixels
iframe = 1064 ; % i.e plot the nth frame
fig = figure();
imagesc(RawFrames(:,:,iframe))
clim([0 40000])
xlabel('columns')
ylabel('rows')
colorbar()

%% plot the dark frame without Bright pixels
iframe = 1064 ; % i.e plot the nth frame
fig = figure();
imagesc(ProcessedFrames(:,:,iframe))
clim([0 10])
xlabel('columns')
ylabel('rows')
colorbar()
```

Listing 3.2: Section of code for plotting dark frames with and without bright pixels.

Listing 3.2 uses the last frame as the reference frame for these example plots. This last frame (1064th frame) was stored in the variable called '**iframe**'. A figure was then produced using the figure function and the aforementioned '**imagesc**' function with the raw frames data. The '**RawFrames**' variable is presented with the brackets '**(:, :, iframe)**', which was in the format of (row, column, frame number). Essentially, this snippet of code is retrieving and plotting onto a figure the desired frame, the last frame in the dataset. The code for plotting the dark frame without bright pixels is exactly the same, besides the type of frame used being changed from '**RawFrames**' to '**ProcessedFrames**', as the processed frames have already had the noise subtracted from them.

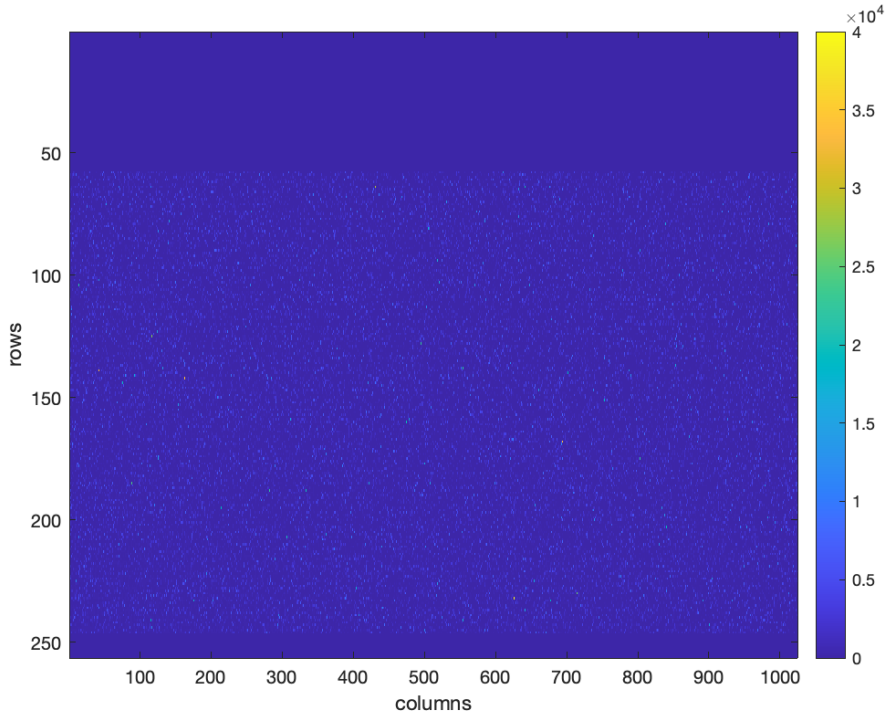


Figure 3.1: Raw Dark Frame Plot.

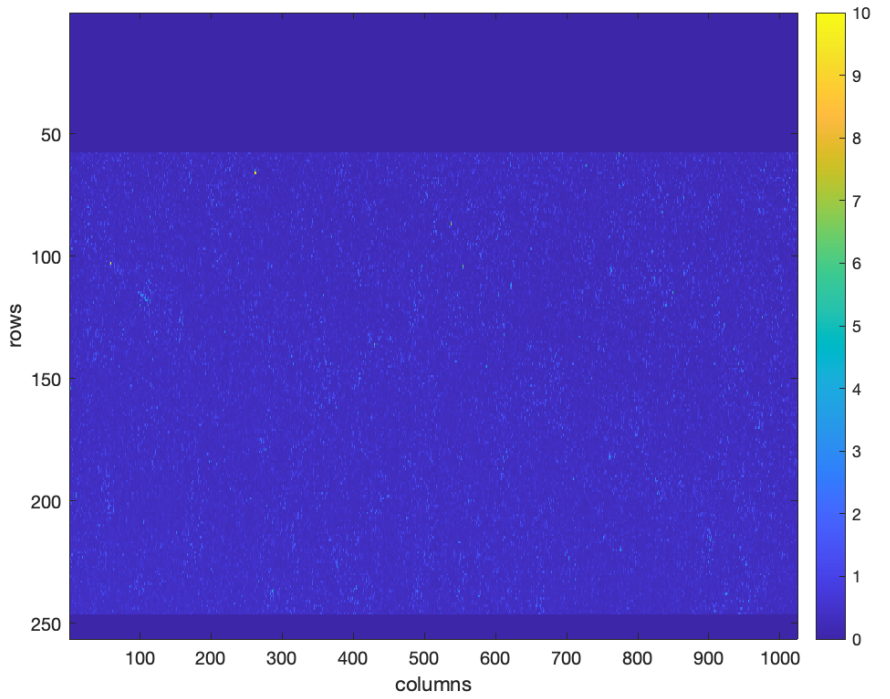


Figure 3.2: Processed Dark Frame Plot.

As demonstrated by Figures 3.1 and 3.2, the dark frames have missing rows at the bottom and the top. This is known as vertical truncation. Vertical truncation is performed by UVIS before it is transmitted back to Earth, resulting in a smaller amount of data being sent. The rows necessary for accurate data measurement are kept, the rest are discarded as they are not vital (Willame et al., 2022). The remaining CCD frame that is transmitted is the region of interest (ROI), which is made up of the 'non-light' region (NLR) and the 'light' region (LR). The NLR is the region seen at the top and bottom of the dark frame, whilst the LR is the illuminated section situated between the NLR.

In addition to vertical truncation, there are three other methods to obtain an ROI: horizontal truncation, vertical binning, and horizontal binning. Horizontal truncation performs the same process as vertical truncation, just in the other direction, discarding columns that are not needed. Binning aims to also reduce the rows and columns on a dark frame, but goes about doing so in an entirely separate approach. Rather than discarding the rows and columns, the process of binning combines multiple rows or columns. In vertical binning, the charge is shifted between adjacent rows, thus adding the charges together into a singular row. The same can be said for horizontal binning, just in the opposite direction. The process of binning greatly increases the signal strength and signal-to-noise ratio (SNR), but sacrifices the resolution of the image transmitted.

As the processed frame shown in Figure 3.2 has had all of the noise and dark current subtracted, it is much easier to identify anomalies and outliers within the data. The plot highlights a few areas with bright pixels, illustrating regions of significant damage due to radiation. Figure 3.3 portrays what a normal pixel should look like in terms of brightness (Digital Number) and what three examples of a bright pixel look like for comparison. When plotted over time, as in Figure 3.3, the significant difference in brightness becomes far more apparent. Additionally, the code to produce Figure 3.3 is presented in Listing 3.3 (Mason, 2025).

```
%% Plot 3 bright pixels and 1 normal pixels to show the difference
    in behaviour.
[Time,isort] = sort(FrameTime,'ascend');
fig = figure;
plot(Time,squeeze(RawFrames(139,42,isort)),'-b', 'DisplayName', '
    bright pixel')
hold on
plot(Time,squeeze(RawFrames(168,694,isort)),'-r', 'DisplayName', '
    bright pixel')
plot(Time,squeeze(RawFrames(73,995,isort)),'-g', 'DisplayName', '
    bright pixel')
plot(Time,squeeze(RawFrames(169,486,isort)),'-k', 'DisplayName', '
    normal pixel')
```

```
xlabel('time / s')
ylabel('DN')
legend()
```

Listing 3.3: Section of code for plotting 3 bright pixels and 1 normal pixels to show the difference in behavour.

Within Listing 3.3, the time of frame capture in seconds ('**FrameTime**') was sorted into ascending order to allow for a cleaner and more readable plot. Additionally, each of the three bright pixels and one normal pixel are plotted in the format '(**Time, squeeze (** **RawFrames (x , y, isort)**)'. The x and y refer to the row and column location of the pixels in question within the relevant frame. Where the frame number is usually stated, the variable 'isort' takes its place and rearranges the third dimension within this array (frames) so that it matches the sorted ascending order of frame times.

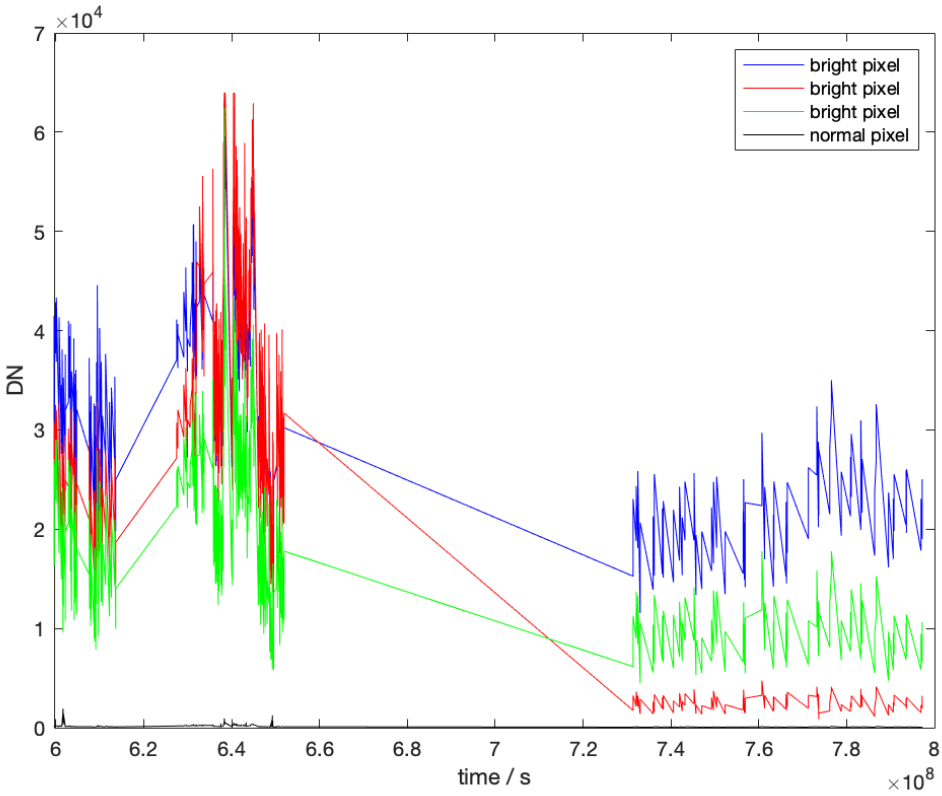


Figure 3.3: Plot comparing a normal pixel to 3 bright pixels.

The solid diagonal lines present in the middle of Figure 3.3 are due to using a condensed version of the full in-flight dataset as previously mentioned. Hence, there is no data for those timestamps due to the usage of a smaller file size. As a result, MATLAB has tried to create its own link between the two data points.

3.2.2 Ground Data

The ground data was stored in a '.mat' format as it was converted to run with MATLAB, it uses a slightly different process to store the metadata. The data was separated into two sections: the Engineering Model (EM) detectors and the Flight Model (FM) detectors. The EM detectors were used for pre-radiation data and the radiation data itself, whilst the FM detectors were devices tested and ready for flight; they were simply used as a means of reference. The EM detectors used a lower grade of silicon and hence were much cheaper to use and manufacture, whilst the FM detectors cost a lot more (due to using higher grade silicon), hence any damage to them would have been catastrophic at the pre-flight stage. As a result, the EM detectors were the choice of device used for the irradiation testing. The metadata for pre-radiation and the flight model was stored within 3 variables: '**pre**', '**rad**', and '**fm**', with each variable having sub level fields for each of the 4 detectors (det1, det2, det3, and det4). Within each detector's sub-level field, there are also additional sub-level arrays which store the integration time (exposure time), temperature, and the dark frame data. For the radiation variable specifically, the way the variables are accessed was slightly different. The sub-levels are arranged according to the level of radiation exposure (half EOL, EOL, and double EOL). Within the doses are sub-levels for dose (value of dose), dark frames, temperature, and integration. Listing 3.4 demonstrates this structure. The code structure and layout was provided by the Open University (Mason, 2025).

```
load('uvis_ground_data.mat') % LOAD IN GROUND DATA FILE

% The data is separated into Engineering Model (EM) detectors and
   Flight
% Model (FM) detectors

% EM data
%%%%%% pre-radiation data %%%%%%%%%%%%%%%%%
pre = gnd.em.pre;
% top level of pre is the 4 detectors
%pre.det1
%pre.det2
%pre.det3
%pre.det4

% The structure for each detector is as follows;
%pre.det1.integration
%pre.det1.temp
%pre.det1.frames

```

```
20
21 %%%%%%% radiation data %%%%%%%%%%%%%%%%%%%%
22 rad = gnd.em.irradiation;
23 % top level of rad
24 %rad.halfeol
25 %rad.eol
26 %rad.eolx2
27 % The structure in rad for each dose is as follows;
28 % rad.halfeol.dose => radiation dose
29 % rad.halfeol.dark.frames => dark frames
30 % rad.halfeol.dark.temp => temperature
31 % rad.halfeol.dark.integration => exposure time
32
33
34 %%%%%%%%% FM data %%%%%%%%%%%%%%%%%%%%
35 fm = gnd.fm;
36 % top level of fm is the 4 detectors
37 %fm.det1
38 %fm.det2
39 %fm.det3
40 %fm.det4
41
42 % The structure for each detector is as follows;
43 %fm.det1.integration
44 %fm.det1.temp
45 %fm.det1.frames
```

Listing 3.4: Section of code for ground data structure.

Through the utilisation of this ground dataset, the raw frames were created for all three of the radiation doses. This once again utilised the 'images' function within MATLAB. These can be seen in Figures 3.4 , 3.5 and 3.6.

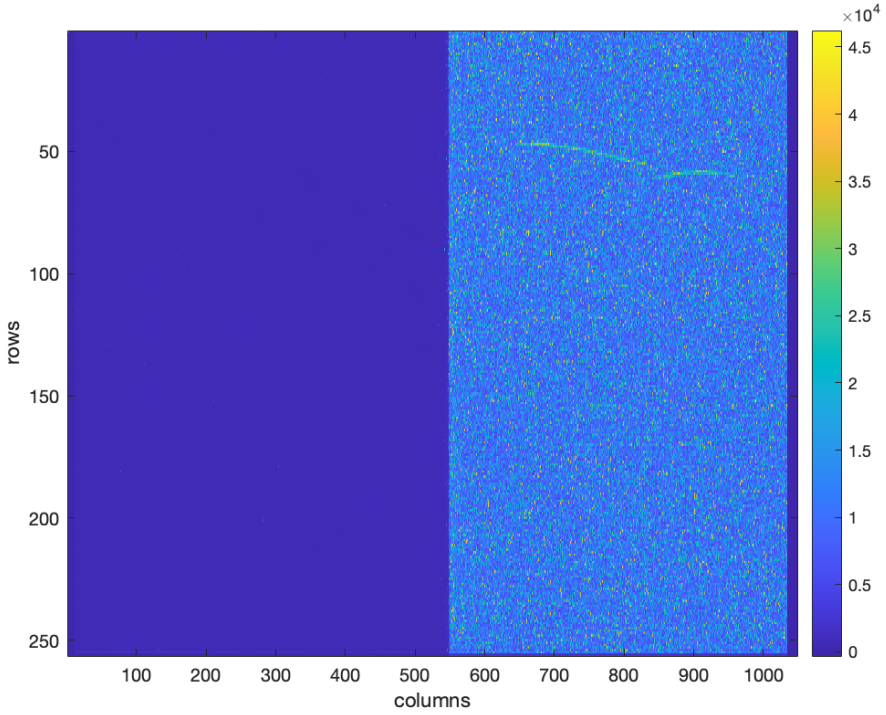


Figure 3.4: Half EOL Dark Frame Plot.

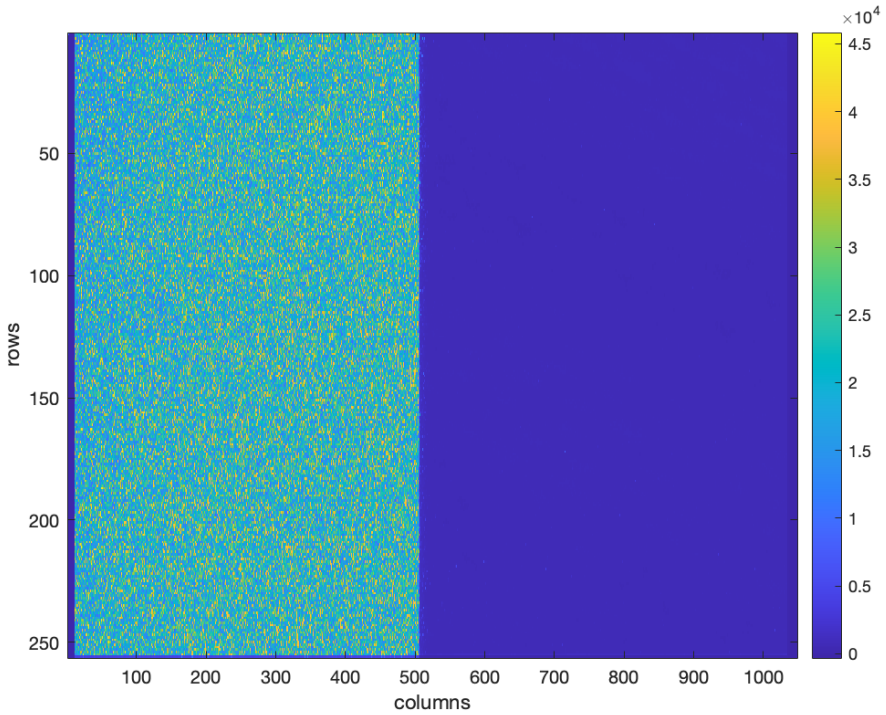


Figure 3.5: EOL Dark Frame Plot.

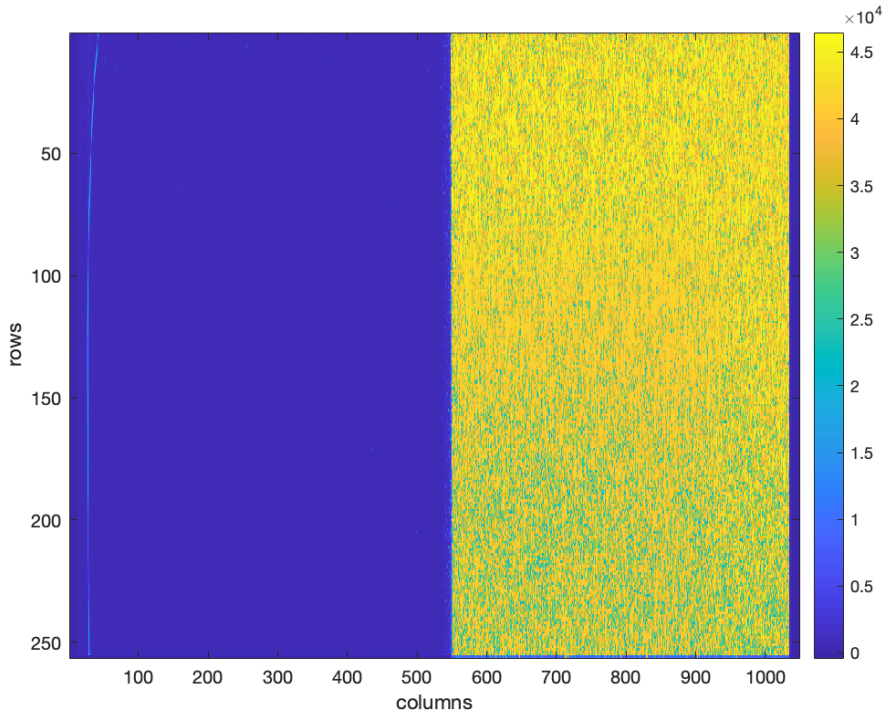


Figure 3.6: Double EOL Dark Frame Plot.

As stated in subsection 2.4.2, specific devices have different irradiation zones. Cross-referencing this with Figures 3.4 , 3.5, and 3.6, it becomes a bit clearer as to which detector was used for which corresponding dose. For the half the end-of-life dose (3.6 x 10^9 protons.cm^{-2}), detector 1 was used. For the double end-of-life dose (9.4 x 10^9 protons.cm^{-2}), detector 3 was used. Thus, detector 2 was used for end-of-life (4.8 x 10^9 protons.cm^{-2}).

The usage of half the image area for irradiation means that half the frame becomes less useful for this study. Furthermore, any stray pixels on the non-irradiated region may skew later data for later calculations and analysis, hence the region of interest has to be adjusted - shown by listing 3.5.

```
%% --- Usable Frame Data Range ---

radHALFEOL_Frames = rad.halfeol.dark.fames(30:230,600:1000,:);
radEOL_Frames = rad.eol.dark.frames(30:230,50:500,:);
radEOLx2_Frames = rad.eolx2.dark.frames(30:230,600:1000,:);
```

Listing 3.5: Section of code for adjusting ROI.

Chapter 4

Image Processing and Analysis Methods

This chapter portrays the process of bright pixel and dark frame extraction using MATLAB. Initially, the bright pixel approach was explored; however due to limitations of data, the focus was shifted to dark current instead.

4.1 Bright Pixel Approach

The initial approach was based around using bright pixels as a figure of merit for comparison between the ground test data and the in-flight mission data. However, upon receiving the ground testing data and looking at the bright pixels present at each dose, bright pixels could not be used as a basis for comparison. The reasoning behind this is detailed throughout the remainder of Section 4.1.

4.1.1 Frame Extraction

The initial step for the assessment of in-flight bright pixels was to focus efforts on finding bright pixels around a specific temperature. In order to identify a specific temperature to be used as a threshold, a plot of temperature against time was generated. Determining a threshold temperature serves as a marker to indicate where most of the data is concentrated, as this is a frequent operating temperature. The code written to achieve this plot can be seen in Listing 4.1 (Mason, 2025). For full code, see Appendix B.

```
%% ---Load the UVIS data from the HDF5 file---

fnm = 'UVIS_data_subset_forJames.h5';
%FrameNumber = h5read(fnm,'/FrameNumber');            % Frame
    number
FrameTime = h5read(fnm,'/FrameTime');                % Frame time
    in seconds
```

```
FrameTemp = h5read(fnm,'/FrameTemp');                % Frame
    temperature in degrees C
Integration = h5read(fnm,'/Integration');            % Exposure
    time
w_RawFrames  = h5read(fnm,'/RawFrames');               % Raw'
    frames across all orbits (contains bright pixels)
w_BrightPixelMap  = h5read(fnm,'/BrightPixelMap');     % The
    brightpixel map for each orbit
w_ProcessedFrames  = h5read(fnm,'/ProcessedFrames');   % Processed
    frames across all orbits (bright pixels removed)

% Going between Python and Matlab transposes the row and column
    dimensions
% - this function corrects for this.
[RawFrames,ProcessedFrames,BrightPixelMap] = reshape_frame_data(
    w_RawFrames,w_ProcessedFrames,w_BrightPixelMap);
[Time,isort] = sort(FrameTime,'ascend');
FrameNumber = (1:size(RawFrames,3));


[~, ~, nt] = size(RawFrames);
disp(['Total number of frames:', num2str(nt)])

%% Reformatting for appearance on plot

function [rdat1,rdat2,rdat3] = reshape_frame_data(dat1,dat2,dat3)
    [nx,ny,nz] = size(dat1);
    rdat1 = zeros(ny,nx,nz);
    rdat2 = zeros(ny,nx,nz);
    rdat3 = zeros(ny,nx,size(dat3,3));
    for i = 1:nz
        rdat1(:,:,i) = dat1(:,:,i)'; % Transpose the array
        rdat2(:,:,i) = dat2(:,:,i)'; % Transpose the array
        if i <= size(dat3,3)
            rdat3(:,:,i) = dat3(:,:,i)';
        end
    end
end

%% ---Temp Vs Time Plot (to identify a threshold Temp)---

fig = figure('Name','Temperature Vs Time Plot','NumberTitle','off')
    ;
plot(Time,FrameTemp(isort));
xlabel('Time [s]')
```

```
ylabel('Temperature [ C ]')
yline(-7.5, 'r--', 'Threshold -7.5 C ')  % Red dashed line at -7.5 C  with label to capture majority of frames (THRESHOLD)
```

Listing 4.1: Code for plotting the Temperature Vs Time.

The first half of the code detailed in Listing 4.1 is replicative of the process outlined in Listing 3.1, whereby the data was reshaped and the variable to load in were stated. The figure was then generated with the time being plotted on the x-axis, and the frame temperatures (sorted in ascending order) are plotted upon the graph. Upon review of the plot, it was decided that the threshold temperature should be set at -7.5 °C, as at this temperature point, the majority of the data was able to be captured. This was then overlaid onto the plot as a dotted red line to act as a visual indicator. It should be noted that the dataset used, at this point of the study, was the condensed version of the full in-flight dataset; hence the temperature threshold decision was based on data from the beginning of the mission and the most up-to-date. The middle segment of the mission is missing within this dataset.

Using this threshold temperature, frames in and around the -7.5 °C mark could be identified and stored. This code to perform the frame identification can be seen in Listing 4.2.

```
%% ---Identifying frames close to Threshold Temperature---

Threshold_temp = -7.5; % Stating the threshold temperature
Tol = 0.5; % Uncertainty of 0.5 C  in each temp measurement --------CHANGED from 0.1 C  to 0.5 C -------

% Finding which frames where the temperature is the threshold
% tolerance... (So essentially -8.0 C  to -7.0 C  using a 0.5 tolerance)
Approx_Threshold_Frames = find(abs(FrameTemp-Threshold_temp)< Tol);


disp('Frames that are close to -7.5 C :');
disp (Approx_Threshold_Frames); % Displays the frames near the threshold...
disp ('Corresponding times for each frame:');
disp (Time(Approx_Threshold_Frames))  % Displays the corresponding time for each frame...
```

Listing 4.2: Code for identifying frames close to the threshold temperature.

Listing 4.2 states the threshold temperature of -7.5 °C and then states a tolerance value. This represents the uncertainty in each temperature measurement, hence if there is a temperature that is 0.5 °C above or below the threshold, it was be considered upon

identifying the frames close to the threshold temperature. Hence, anything outside of this specified range will not be identified. The calculation can be seen in line 8 of Listing 4.2; the absolute value of the '**Threshold_temp**' was subtracted from '**FrameTemp**', and then checked to ensure that the difference was less that the specified 0.5 °C margin of error.

Now that these frames had been identified within the data, they needed to be extracted and stored separately. Listing 4.3 shows this.

```
%% Extracting Frames

ID_Frames = Approx_Threshold_Frames;
Temps = ProcessedFrames; % Stores processed frames array into Temps
    variable
keeparray = zeros(1064,1); % Setting all 0s for the entire array

%Loops over ID_Frames, to store the indices of the target frames.
   Sets those positions in keeparray to 1.... (1s are kept and 0s
   are removed).
for i = 1:length(ID_Frames)
    keeparray(ID_Frames(i))= 1;

end

       keeparray = logical(keeparray); % Sets the 1s and 0s to True
    and False
       keeparray = ~keeparray; % Inverts the keeparray, so instead
   of keeping these frames they are now removed
       Temps(:,:,keeparray) = []; % Only the frames close to
   temperature remain (Deletes all non-selected frames)

meanTemps= mean(Temps,2); % Average of temperature row-wise (across
    image height)...
meanTemps_variation= mean(meanTemps,1); % Average of temperature
   column-wise, thus leaving a mean temperature of each frame
meanTemps_1D = squeeze(meanTemps_variation); % Converted into a 1D
   vector

fig = figure('Name','Frame # Vs Mean Temperature Variation [ C ]','
   NumberTitle','off');
plot(1:length(meanTemps_1D), meanTemps_1D, 'o-');
xlabel('Frame #');
ylabel('Mean Temperature Variation [ C ]');
title('Mean Temperature of Selected Frames');
grid on;
```

Listing 4.3: Code for extracting frames close to the threshold temperature and plotting

the mean temperature variation.

Identified frames (‘**ID_Frames**’) were set to equal the ‘**Approx_Threshold_Frames**’ variable, meaning that the index of the frames identified in the Listing 4.2 are now saved in an array. The ‘**Temps**’ variable then stores the actual frame dimension data of the identified frames. The ‘**keeparray**’ variable sets all of the frames to a value of 0, which was useful for the loop that is run next.

A for loop was created to loop thorough all of the frames and set the frame index (position) to a value of one within the ‘**keeparray**’. By doing so, all of the identified frames in question will return a value of one whilst all others will return a zero. The ‘**keeparray**’ was then switched from ones and zeros to true and false, with the frames close to the threshold temperature being set to true. The ‘**keeparray**’ was then inverted, thus removing the wanted frames from ‘**keeparray**’ and storing them in the ‘Temps’ variable.

The‘**meanTemps**’ and ‘**meanTemps_variation**’ then filter through all of the frames row-wise and column-wise to find the mean temperature which was stored and compressed into a 1D array, as indicated by ‘meanTemps_1D’. The plot was then generated using the mean temperature value per frame (‘**1:length(meanTemps_1D)**’)as the y-axis and the corresponding frame number as the x-axis.

4.1.2 Bright Pixel Characterisation

Now that a subset of frames close to the threshold temperature had been extracted, bright pixels could start to be analysed and characterised. The best way to try and characterise a bright pixel would be to create a pixel noise threshold; hence it could be stated that all values above a certain Digital Number(DN) are bright pixels. In order to achieve this, the dark current must first be computed and then subtracted from the raw frame to produce an overall pixel noise.

The first section of code can be seen in Listing 4.4. For full code, see Appendix C.

```
%% --- Compute Mean Dark Current per Frame Plot---

MeanDarkCurrent = zeros(1, nt);  % Mean DN per frame
TempPerFrame = FrameTemp;        % Already in degrees C
ExpTime = Integration;           % Exposure time per frame, in
    seconds

% Mean DN per frame normalised by exposure time
for i = 1:nt
    frame = ProcessedFrames(:,:,i);      % Dark frame with no
    bright pixels
```

```
    MeanDN(i) = mean(frame(:), 'omitnan');  % Mean DN over all
    pixels
    MeanDarkCurrent(i) = MeanDN(i);
    MeanDarkCurrent(i) = MeanDN(i) * ExpTime(i);  % DN/s
end

%% --- Plot: Mean Dark Current vs Temperature ---

figure('Name','Mean Dark Current (DN) Vs. Tempertature ','
    NumberTitle','off');
plot(TempPerFrame, MeanDarkCurrent, 'o');
xlabel('Temperature [ C ]');
ylabel('Mean Dark Current [DN/s] (log scale)');
title('Mean Dark Current vs. Temperature (Log Scale)');
grid on;

%% --- Histogram : Dark Current Vs. Time (Singular Frame) ---

Dark_Current = RawFrames(:,:,end); % Looking at Final Frame
figure('Name','Dark Current Over Time ','NumberTitle','off');
h = histogram(Dark_Current,2000); % Resolution specified by number
    of bins at the end of this line
h.Parent.XScale = "log";

xlabel('Time [s]');
ylabel('Dark Current [DN]');
ylim([0,40000]);
title('Dark Current Over Time (Singular Frame)');
```

Listing 4.4: Code for computing the mean dark current as well plotting mean dark current against temperature and a histogram for dark current against time.

Prior to Listing 4.4, the dataset is loaded into MATLAB in the same manner as Listing 4.1. The mean dark current array was set to all zeros, the temperature per frame was stored as '**TempPerFrame**', and the integration time was stored as '**ExpTime**'. A for loop was then implemented to cycle through all of the processed frames and find the mean DN. The 'MeanDN' and exposure time are then used to calculate the mean dark current. The nominal formula for the mean dark current calculation is shown in Equation 4.1.

$$MeanDarkCurrent = \frac{MeanDN}{IntegrationTime} \tag{4.1}$$

However, when initially processed and stored before being received, the DN values had already been divided by the integration time twice. Hence, to undo this, the 'MeanDN' calculated in listing 4.4 must be multiplied by the integration time values to pro-

duce the mean dark current. A plot of the mean dark current against temperature was plotted to see the correlation between the two variables. The same was done for the dark current and time; however, this was done in the form of a histogram, portraying the point within a frame at which the dark current was at its peak value.

Histograms are also created to represent the pixel noise against time and the pixel count against pixel noise. The algorithm to generate the histograms can be seen in Listing 4.5.

```
%% --- Histogram : Pixel Noise Vs. Time (Singular Frame) ---

% Looking at Final Frame
Pixel_Noise = RawFrames(:,:,end)-MeanDN(end); % Essentially
    Background Noise as the meaDN has been sutracted (BRIGHT PIXELS)
figure('Name','Pixel Noise Over Time ','NumberTitle','off');
h = histogram(Pixel_Noise,3000); % Resolution specified by nunmber
    of bins at the end of this line
h.Parent.XScale = "log";

xlabel('Time [s]');
ylabel('Pixel Noise [DN]');
ylim([0,25000]);
title('Pixel Noise Over Time (Singular Frame)');

%% --- Histogram : Pixel Count Vs. Pixel Noise ---

BP_Vector = Pixel_Noise(:); % Bright Pixel Vector
medi = median(BP_Vector, 'omitnan'); % Finding the median value of
    all the values in the Bright Pixel Vector values
sigma = std(BP_Vector, 'omitnan'); % Stating the sigma value ot be
    the standard deviation of the Bright Pixel Vector values
threshold_DN = medi + (3 * sigma); % Creates the theshold to be 3
    sigma above the median...

figure('Name','Pixel Count Over Pixel Noise','NumberTitle','off');
h = histogram(BP_Vector(BP_Vector > 0), 3000);
h.Parent.XScale = "log";

xlabel('Pixel Noise [DN]');
ylabel('Pixel Count');
title('Pixel Noise Distribution in a Single Frame');

% Add 3 sigma threshold line
hold on;
yLimits = ylim;
```

```
plot([threshold_DN threshold_DN], yLimits, 'r--', 'LineWidth', 2);
hold off;

legend('Pixel Noise Histogram', '3   Threshold');
disp(['3-Sigma DN Threshold: ', num2str(threshold_DN)]);
```

Listing 4.5: Code for plotting two histograms; Pixel Noise against Time and Pixel Count against Pixel Noise.

Using the final frame in the dataset as a reference point in Listing 4.5, the overall pixel noise was calculated by subtracting the mean noise from the raw frames. This remaining pixel noise was just made up of random fluctuation and bright pixels. This was then visual represented on a histogram over time to depict the distribution of bright pixels as they appear on the positive side of the histogram as an asymmetric tail. More pixels with abnormally high dark current means more bright pixels, hence a larger tail will be seen.

Additionally, another histogram was created, but this time the focus was on pixel count against pixel noise. The '**Pixel_Noise**' was reshaped to a 1D array and stored in the '**BP_vector**' variable. The median and standard deviation value of all values within '**BP_vector**'. This was then used to calculate the noise threshold via use of Equation 4.2.

$$ThresholdDN = Median + 3\sigma \tag{4.2}$$

The Pixel count against pixel noise histogram was plotted with a 3σ threshold visually represented by a red line on the histogram. The idea behind using 3σ was that due to the histogram being a Gaussian distribution (Chen, 2024), the majority of the data will lie within the 3σ threshold. Anything above the 3σ threshold was of interest as this was likely to be where the brightest pixels will occur.

In order to capture the bright pixels above the 3σ threshold, a binary mask can be generated to define the region of interest. This workflow is shown in Listing 4.6.

```
%% Creating a Binary Mask using the 3-Sigma Threshold.

% Create binary mask of high-noise pixels (above or equal to
    threshold)
HighNoiseMask = Pixel_Noise >= threshold_DN;


figure('Name','High Noise Pixel Mask','NumberTitle','off');
imagesc(HighNoiseMask);
colormap(parula);
colorbar;
xlabel('columns')
ylabel('rows')
```

```
13 title('Mask of High-Noise Pixels (Above 3-sigma Threshold)');
14
15 FilteredData = RawFrames(:,:,end); % Looking at the filtered data
       applied to the end frame
16 FilteredData(~HighNoiseMask) = NaN;  % Keep only high-noise pixels,
       set rest to NaN
17
18 figure('Name','DN Values of High-Noise Pixels','NumberTitle','off')
       ;
19 imagesc(FilteredData);
20 colorbar;
21 title('DN Values of High-Noise Pixels in Final Frame');
22
23 %% Standard Deviation of those pixels within the final frame
24
25 High_Noise_Pixels = double(RawFrames(:,:,end)); % Doubles for more
       precision and looks at the end frame
26 High_Noise_Values = High_Noise_Pixels (HighNoiseMask);
27
28 Noise_STD = std(High_Noise_Values, 'omitnan');
29
30 disp(['Standard Deviation of High-Noise Pixels: ', num2str(
       Noise_STD)])
```

Listing 4.6: Code for creating a binary mask and applying to data to produce dark frames.

The '**HighNoiseMask**' stores all pixels with a noise greater than or equal to the 3σ threshold. This mask was then plotted visually as a dark frame to allow for the assessment of spatial distribution over the sensor; identification of clusters or cosmic rays can become much clearer this way. The '**HighNoiseMask**' was then applied to the end frame (reference frame), leaving only a dark frame with bright pixels that have been identified. Lastly, for Listing 4.6, the standard deviation between these bright pixels was calculated to evaluate the severity of the damage caused and to adjust the 3σ if necessary. All plots and figures produced from this subsection can be seen in Appendix D.

4.1.3 Pivot to Dark Current

The initial approach was to continue applying the noise threshold to each of the identified frames at the specified temperature and then apply this to the full size dataset; however, this technique worked out to be only short-term. The compressed in-flight data was received prior to the ground test data file, hence a large amount of work had already been carried out in regard to looking at bright pixels analysis of the ground test data had begun. Upon retrieval of the ground test data, the data was manipulated to output dark frames

throughout the irradiation process, as well as a few plots. It quickly became apparent that the bright pixels were nowhere near as distinguishable as expected when compared to the in-flight data. The in-flight ground pixels simply were not replicated by ground testing. A key factor behind this issue was damaged pixels/defects annealing. The inherent limitation that presents itself with irradiation ground testing was that the duration of the test campaign was very short when compared to the mission lifetime, hence it was not likely to fully replicate the long-term effects present over the longer duration. When applying this to annealing, the pixels/defects get very little time to anneal and thus will be in much higher quantities during ground testing. This was further reinforced by the dark frame plots in subsections 3.2.1 and 3.2.2.

As a result of this matter, dark current was chosen as the next best figure of merit to be used as a point of comparison between the in-flight data and the ground test data. Through the usage of dark current, this more closely relates to the Gow study on 'Proton Radiation Damage Assessment of a CCD for use in a Ultraviolet and Visible Spectrometer' (Gow et al., 2017).

4.2 Ground Test Data Approach

Due to dark current becoming the main figure of merit, the logical steps to take were to recreate the plots from the Gow study (Gow et al., 2017) as a baseline, then make the necessary adjustments to improve its suitability for comparison with the in-flight mission data. For full code, see Appendix E.

4.2.1 Plotting Dark Current against Temperature

Plotting dark current against temperature was a crucial step for characterisation of CCD behaviour within the space radiation environment. The usual expected behaviour for a CCD is for dark current to increase exponentially, given a rise in temperature. However, the plot chosen to create was an Arrhenius plot, whereby the natural log of dark current is taken as the y-axis, as the x-axis becomes the inverse of temperature (1 / Temperature). In this specific plot, 1000 / Temperature was used as the x-axis due to the magnitude of the values produced. The relationship obeys Arrhenius law and relationship as stated in Equations 4.3 and 4.4, which is as follows:

$$k(T) = A \exp\left(-\frac{E_a}{k_B T}\right) \tag{4.3}$$

hence:

$$I_d \propto (-\frac{E_a}{k_B T}) \tag{4.4}$$

where:
k(T) = rate of the thermally activated process (dark current in this case)
A = pre-exponential factor (scaling factor)
E_a = activation energy
k_B = Boltzmann constant (8.62 x 10^5 eV.K^{-1})
T = Temperature [K]
I_d = Dark Current

The purpose of an Arrhenius plot is to analyse the temperature dependence of a certain reaction rate, which in this case was the temperature dependence of dark current. For the purposes of identifying a trend in the data and comparing the datasets, an Arrhenius plot proves advantageous over a simple linear plot.

As stated in subsection 3.2.2, only half the image area proves to be useful due to the way that the ground test was conducted. Once this ROI has been adjusted to accommodate for this, the next step was to find the calibration gain of the CCDs used for this study. Calibration gain is defined as Equation 4.5:

$$g = \frac{electrons(e^-)}{DigitalNumber(DN)} \tag{4.5}$$

A CCD usually has an inherent calibration gain factor that is used for all further analysis. For this specific study, the CCD calibration was not directly stated; it was said to have been calculated, but no value was made evident. In order to get around this, a Photon Transfer Curve (PTC) was used to identify the gain value of each detector used. A PTC is a plot that is widely used to illustrate how the variance of pixel signal (noise) changes in relation to its brightness/light intensity level. The variance (σ^2) of the pixel signal was plotted against the mean output signal (DN), and the gradient of the slope produced provides the gain value. The expected behaviour for this plot was that the data points are not exactly linear and will flatten out at higher doses of radiation (Astier et al., 2019), however, a line of best fit can be used to best represent the data and get the desired gain value. A separate PTC was plotted for each detector as seen in Figures 4.1, 4.2, and 4.3, along with the code in Listing 4.7. The PTC curve code was provided by the Open University (Mason, 2025).

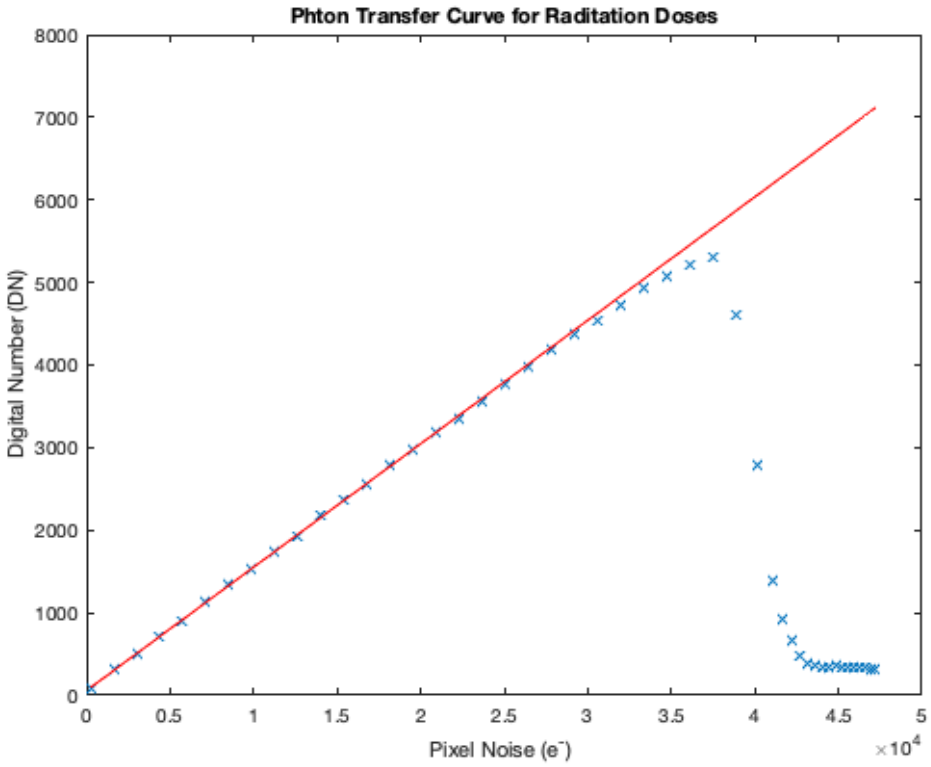


Figure 4.1: A figure to illustrate the PTC Plot for Half End of Life dose.

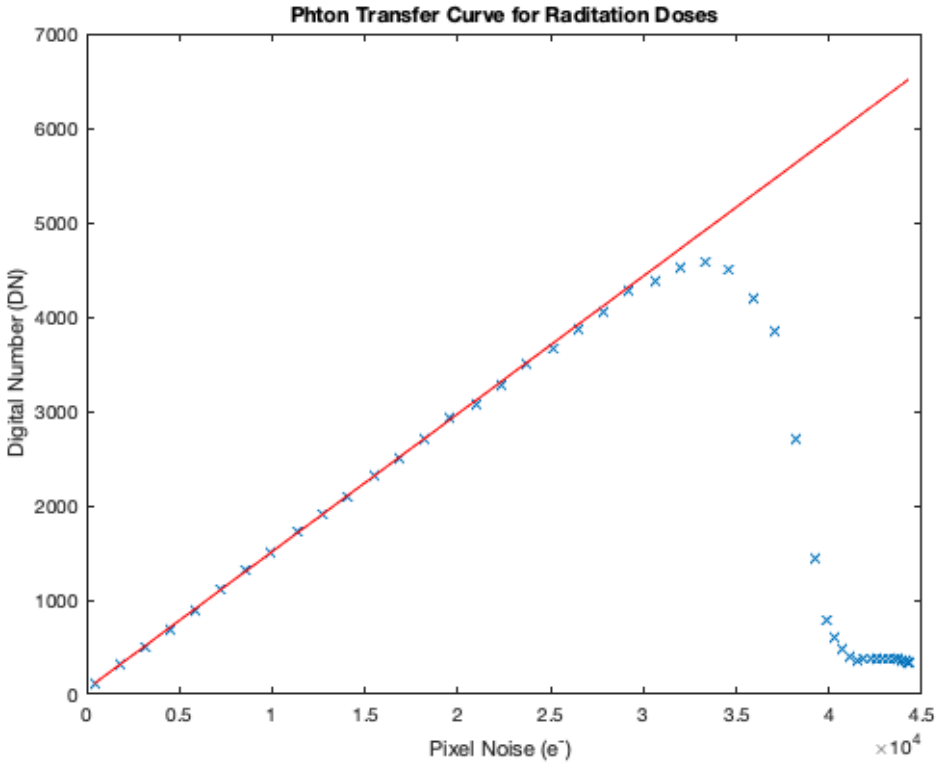


Figure 4.2: A figure to illustrate the PTC Plot for End of Life dose.

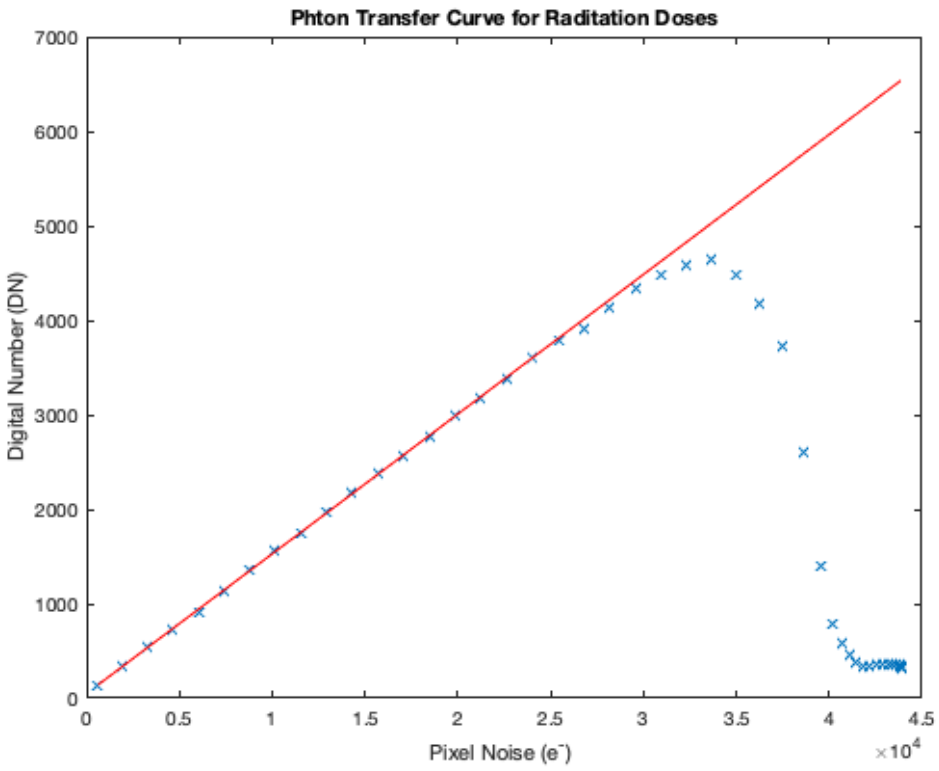


Figure 4.3: A figure to illustrate the PTC Plot for Double End of Life dose.

```
%% Get the PTC for radiation doses

get_electronsperDN(rad.eol.ptc)

function gain1 = get_electronsperDN(ptc)



    % get the subframe that was exposed
    % use the pair of images to get the variance in the frame by
    % subrtracting one from the other.
    nframes = size(ptc.images,3) /2;
    p = 1;
    for i = 1:2:size(ptc.images,3)
         frame1 = ptc.images(70:225,250:800,i);
         frame2 = ptc.images(70:225,250:800,i+1);
         sframes(:,:,p) = frame1 - frame2;
         mval(p) = mean(reshape(frame1, numel(frame1),1));
         mval2(p) = mean(reshape(frame2, numel(frame1),1));
         var(p) = std(reshape(sframes(:,:,p), numel(sframes(:,:,p))
    ,1))^2;
         p = p + 1;
    end

    % Plot the mean signal against the variance
    figure()
    plot(mval,var,'x')
    % fit the slope of the curve near the bottom with a linear fit
    y = mx +
    % c
    % The gradient (m) is the gain and 1/m gives e- per ADU
    p =polyfit(mval(1:10),var(1:10),1);
    pval = polyval(p,mval);
    hold on
    plot(mval,pval,'r')
    xlabel('Pixel Noise (e^-) ')
    ylabel('Digital Number (DN)')
    gain1 = 1 / p(1);

end
```

Listing 4.7: Code for computation gain via Photon Transfer Curve (PTC).

Listing 4.7 is the exact code for the EOL PTC. In order to get half EOL and double EOL PTCs and gain values, line 3 of Listing 4.7 must be replaced with '**get_electronsperDN (rad.halfeol.ptc)**' and '**get_electronsperDN (rad.eolx2.ptc)**' respectively. Addition-

ally, the gain of the PTC is equal to '**1/p**',where '**p**' is the gradient of the fitted line. The gain values derived from Figures 4.1, 3.5, and 4.3 are summarised in Table 4.1.

Radiation Dose	**Gain** (e^-/DN)
Half End-Of-Life (3.6 x 10^9 protons.cm^{-2})	6.69
End-Of-Life (4.8 x 10^9 protons.cm^{-2})	6.85
Double End-Of-Life (9.4 x 10^9 protons.cm^{-2})	6.76

Table 4.1: A table to show the calibration gains at their respective radiation doses.

Implementing these gains, the dark current values can be computed. This process is outlined in Listing 4.8 as below.

```
%%% ---Computing Dark Current--- %%%

function [invT, dark_c] = getDarkCurrent(frames, temps_C,
    integration, gain)
    % Compute mean DN per frame
    meanDN = mean(frames, [1 2]);

    % Using first integration time of array (as repeated)
    if numel(integration) > 1
        integration = integration(1);
    end

    % Dark current in e-/pixel/s
    dark_c_all = (meanDN * gain) ./ integration;

    % Average dark current per temperature
    [uniqueT, ~, ic] = unique(temps_C, 'stable');
    dark_c = zeros(size(uniqueT));
    for i = 1:length(uniqueT)
        dark_c(i) = mean(dark_c_all(ic == i));
    end

    % Inverse temperature (1000 / Temp in K) for Arrhenius plot
    invT = 1000 ./ (uniqueT + 273.15);

    % --- Sort points by increasing invT ---
    [invT, sortIdx] = sort(invT);
    dark_c = dark_c(sortIdx);
end
```

Listing 4.8: Code for computation of Dark Current.

The function defined for retrieving dark current was named '**get dark current**' and has inputs of frames, temperature in °C, integration time, and the gain as previously spe-

cified. The outputs of this function are '**invT**' and '**dark_c**', which are the inverse of temperature (1000/Temperature in Kelvin) and the average dark current at each temperature $e^{-}/pixel/s$.

Once the dark current function was defined, the mean Digital Number (DN) per frame had to be calculated within the function. The mean of all frames was taken across the first two dimensions, the rows and columns which was equal to the image area. Essentially, this scans across the rows and the columns, determines the DN of each pixel that makes up a frame, then outputs the mean DN of the entire frame (single value) and stores it in the variable of '**meanDN**'. As DN refers to signal level (how brightness is categorised for CCDs), this variable will now have an average signal per frame stored in individual 1D arrays per frame.

Additionally, the integration time for capturing dark frames is usually 15 seconds, which aligns well with this ground testing data; however, as the tests were run 3 times at each temperature, there are multiple versions of the same integration time stored in an array. This can cause problems later on with mismatched array sizes. To get around this problem, the solution was to create an if loop that states if the number of elements within the integration time array was greater than 1 (they all are due to multiple tests), the new integration time should simply be the first value within the array, thus removing the repeated values.

Using the meanDN, integration time, and gain values, the dark current for each frame can now be calculated. Equation 4.6 is used to do so.

$$DarkCurrent = \frac{meanDN \times gain}{IntegrationTime} \tag{4.6}$$

As per Equation 4.6, multiplying by the mean digital number by the gain allows conversion of signal from DN to electrons. Dividing this value by integration time produces the rate of dark current in $e^{-}/pixel/s$.

After computing the dark current rate, it has to be average per temperature due to the nature of the Arrhenius plot. Just like the integration times, the temperatures are repeated within the array due to multiple tests being conducted; however, instead of the all being one temperature, there were multiple temperatures with 3 repetitions of the same temperature within the array. Essentially, the array will look something like this, [-10, -10, -10, -5, -5, -5, 0, 0, 0, +5, +5, +5]. This would be the case for all temperatures within the array. This is where the '**unique**' function comes in. The '**unique**' function looks at the temperatures and picks the first occurrence of each temperature, thus reducing the array size to having all unique temperatures and no repetitions. Lines 17 to 20 in Listing 4.8 then rearrange the array size of the dark currents previously found and match them with the array size of the new '**unique**' temperatures. Then, a for loop was ran to get the

average dark current at each of the temperatures and was stored in the '**dark_c output**'.

The inverse temperature was calculated next in the function. Equation 4.7 illustrates how the inverse temperature was calculated.

$$InverseTemperature = \frac{1000}{Temperature + 273.15} \tag{4.7}$$

where:
Temperature = Unique Temperatures stored in 'uniqueT'

Demonstrated in Equation 4.7, the factor of 1000 was used to make the plot axis more readable, and the addition of 273.15 to the temperature was simply a conversion between Celsius and Kelvin. The dark currents are then sorted in ascending order of inverted temperatures, ready for plotting.

Now that the function for getting the dark current and the inverse temperature was in place, the different radiation dosages needed to be considered. The way to do this was via using the different detectors, as previously mentioned, each detector was exposed to a different radiation dose, with the control CCD receiving no dose at all. The code for this process is made evident in Listing 4.9.

```
%%% ---Dark currents per dose--- %%%

% Control
pre_int = pre.det4.integration(1);
[invT_ctrl, dark_ctrl] = getDarkCurrent(pre.det4.frames, pre.det4.
    temp, pre_int, gainHALFEOL);

% Half EOL
halfeol_int = rad.halfeol.dark.integration(1);
[invT_d1, dark_d1] = getDarkCurrent(radHALFEOL_Frames, rad.halfeol.
    dark.temp, halfeol_int, gainHALFEOL);

% EOL
eol_int = rad.eol.dark.integration(1);
[invT_d2, dark_d2] = getDarkCurrent(radEOL_Frames, rad.eol.dark.
    temp, eol_int, gainEOL);

% Double EOL
eolx2_int = rad.eolx2.dark.integration(1);
[invT_d3, dark_d3] = getDarkCurrent(radEOLx2_Frames, rad.eolx2.dark
    .temp, eolx2_int, gainEOLx2);
```

Listing 4.9: Code for computaion of Dark Current as a function of temperature for each radiation dose.

The Control CCD (detector 4) was used first, as this forms a reference point to compare the other detectors to. Once again, the integration time was set to take the first value within the array to avoid repeated values. The detector 4 variables of specific frames, temperatures, integration time, and calibration gain value were then retrieved. It is to be noted that the value of gain used for the control CCD was the same as the half EOL detector, as this data was not provided and, hence the closest approximation was half EOL. Once this data had been retrieved, it was run through the '**getDarkCurrent**' function, whereby the process outlined in Listing 4.8 was performed, and outputs of inverse temperature and dark current are stored in the variables '**invT_ctrl**' and '**dark_ctrl**' respectively. This same process was repeated for half EOL (detector 1), EOL (detector 2), and double EOL (detector 3).

In terms of producing the actual plot, the procedure is presented in Listing 4.10.

```
%%% Arrhenius Plot %%%
figure('Name','Arrhenius Plot of Dark Current vs Temperature','
    NumberTitle','off')

% Define consistent markers and colours
plotMarkers = {'o','s','^','d'};
plotColours  = lines(4); % MATLAB default colour order

semilogy(invT_ctrl, dark_ctrl, ['-' plotMarkers{1}],'LineWidth'
    ,0.5, 'Color', plotColours(1,:), 'DisplayName', 'Control CCD');
    hold on
semilogy(invT_d1, dark_d1, ['-' plotMarkers{2}], 'LineWidth',0.5,'
    Color', plotColours(2,:), 'DisplayName', 'Half EOL     2.4 10 ^9
     p.cm^{-2}')
semilogy(invT_d2, dark_d2, ['-' plotMarkers{3}],'LineWidth',0.5, '
    Color', plotColours(3,:), 'DisplayName', 'EOL     4.8 10 ^9 p.cm
    ^{-2}')
semilogy(invT_d3, dark_d3, ['-' plotMarkers{4}],'LineWidth',0.5, '
    Color', plotColours(4,:), 'DisplayName', 'Double EOL     9.4 10
    ^9 p.cm^{-2}')

xlabel('1000 / Temperature (K)')
ylabel('Dark Current (e^{-} pixel ^{-1} s ^{-1})', 'Interpreter','
    tex')
legend('Location','southwest')
grid on
title('Arrhenius Plot of Dark Current vs Temperature')
```

Listing 4.10: Code for Arrhenius Plot of Dark Current Vs Temperature.

As seen from Listing 4.10, the operation of producing an Arrhenius plot was relatively

straightforward. The figure was named, and then the plot markers and colours are defined to allow easy identification of the different data points from different detectors. Each line of data was plotted individually using the '**semilogy**' function which plots the natural log of all values on the y-axis (dark current in this scenario), and then finally, the x and y axes are labelled, the legend location specified, the grid turned on for easy analysis and the title of the plot displayed on the plot window.

4.2.2 Plotting Dark Current against Proton Fluence

Another crucial plot for the analysis of dark current was the plot of dark current against proton fluence. This plot directly illustrates how the radiation exposure (proton radiation in this study) affects the performance of the CCD via the metric of dark current. Additionally, from this plot, the lifetime of the mission can be assessed and justified based upon the performance of the CCD with regard to the radiation dose (proton fluence). If the effects of the EOL dose are seen to be too adverse, then this informs whether mitigation strategies should be developed in aid of reducing the levels of dark current.

Once again, like in subsection 4.2.1, the gains for each radiation dose are established first and stored in e^-/DN. From the ExoMars ground test paper (Gow et al., 2017), the temperatures and desired dark current level at 0°C could start being defined. This is indicated within Listing 4.11.

```
% Gain Values
gainHALFEOL = 6.69; % e-/DN
gainEOL = 6.85;
gainEOLx2 = 6.76;

targetTemps = [20, 10, 0, -10, -20];
desired_limit_0C = 2000;  % e-/pix/s @ 0 C
```

Listing 4.11: Code for gain values as well as establishing the target temperatures and the desired dark current limit.

As illustrated by Listing 4.11, the target temperatures can be seen ranging from +20°C to -20°C, however, it is important to note that in the Gow paper, the temperature range surpasses -20°C and goes down to -30°C. As the data has been pre-processed beforehand, no data was provided for -30°C as it was deemed unnecessary for this particular study due to prior knowledge pertaining to the in-flight data temperatures. Moreover, the Gow paper sets the desired dark current limit at 0°C to be equal to 2000 $e^-/pixel/s$ in relation to the objectives set out by UVIS (Gow et al., 2017). This value is seen to be a threshold for dark current, as ideally the EOL dark current needs to be below 2000 $e^-/pixel/s$ at 0°C. If this was proven to be the case, then the CCDs can be sent on the mission with

confidence that the dark current will not detrimentally compromise the performance of the device throughout the mission duration.

Once these variable have been stated, the proton fluences (radiation doses) must be converted from a string value to a double array, and the lifetime flux must be stated. This is reflected in Listing 4.12.

```
% Proton Fluences
HalfEOL_fluence = str2double(rad.halfeol.dose);
EOL_fluence    = str2double(rad.eol.dose);          % p cm^-2
DoubleEOL_fluence = str2double(rad.eolx2.dose);
lifetime_flux = 1.2 * EOL_fluence;     % p cm^-2 --lifetime margin
    1.2x

% Fluence axis (Pre = 0)
fluences = [0, HalfEOL_fluence, EOL_fluence, DoubleEOL_fluence];
```

Listing 4.12: Code for conversion of proton fluences

The correction in proton fluence was due to a formatting issue upon storage of the radiation doses within MATLAB. MATLAB reads in the radiation doses as text and not actual numbers; hence, when trying to use the proton fluences in calculations or plotting a value, MATLAB produces an error as the text has no computational value. Additionally, the lifetime flux was stated as 1.2 times the EOL proton fluence in line with the level set from the ground testing Gow paper (Gow et al., 2017). The use of lifetime flux provides a safety margin which becomes important when comparing the value at 1.2 times EOL to the desired dark current limit of 2000 ($e^{-}/pixel/s$). Line 8 of Listing 4.12 collects all of the proton fluences (radiation doses) and stores them in a variable called fluences, which was utilised at a subsequent stage of producing this plot.

At this point, a function called 'dark_vs_temp' was created to find the average dark current per temperature. This function performs the same process as lines 15 to 20 in Listing 4.8, but just stores them under different variable names. This was done so that these plots could be generated individually from each other, and any anomalies or outliers would not be carried over to the new plot.

However, after this, a new function was defined called '**nearest_dc_at**' which was a function for extracting temperatures within the dataset that are close to the target temperatures specified in Listing 4.11. The code for both this function and the previous '**dark_vs_temp**' function is shown in Listing 4.13.

```
%%% Compute dark current (averaged per unique T), then pick nearest
    T %
function [Tuniq, avgdc] = dark_vs_temp(frames, temps_C, integration
    , gain)
    meanDN = mean(frames, [1 2]);          % DN per frame
```

```
    if numel(integration) > 1, integration = integration(1); end
    dark_all = (meanDN * gain) ./ integration;      % e-/pix/s for
   each frame
    [Tuniq, ~, grp] = unique(temps_C(:), 'stable');
    avgdc = zeros(size(Tuniq));
    for k = 1:numel(Tuniq)
        avgdc(k) = mean(dark_all(grp == k));
    end
end

function dc_at_T = nearest_dc_at(frames, temps_C, integration, gain
   , Ttarget)
    [Tuniq, dcuniq] = dark_vs_temp(frames, temps_C, integration,
   gain);
    % pick closest temperature
    [~, idx] = min(abs(Tuniq - Ttarget));
    dc_at_T = dcuniq(idx);
end
```

Listing 4.13: Code for computing dark current and sorting temperatures.

When going through the dataset, the temperatures are never exactly the target values and seem to be within ± 0.5 °C. To make the plots more understandable and easy to compare in relation to the in-flight data, this function will allocate the nearby temperatures to their corresponding target temperature. For example, if the exact temperature is -20.7 ° C, then this function will allocate that value to -20 °C. As a result, all other data points relating to that original temperature will now read at -20 °C.

At these five temperatures, the dark current values must be collected and stored at pre-irradiation, half EOL, EOL, and double EOL. The code in Listing 4.14 achieves this.

```
%%% Collect dark current at the target temperatures %%%

pre_int = pre.det4.integration(1);

dc_pre_20  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
   pre_int, gainHALFEOL, 20);
dc_pre_10  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
   pre_int, gainHALFEOL, 10);
dc_pre_0   = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
   pre_int, gainHALFEOL, 0);
dc_pre_m10  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
   pre_int, gainHALFEOL, -10);
dc_pre_m20  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
   pre_int, gainHALFEOL, -20);

```

```
11
12 % Half-EOL
13 halfeol_int = rad.halfeol.dark.integration(1);
14 dc_hfeol_20  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
       temp, halfeol_int, gainHALFEOL, 20);
15 dc_hfeol_10  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
       temp, halfeol_int, gainHALFEOL, 10);
16 dc_hfeol_0   = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
       temp, halfeol_int, gainHALFEOL, 0);
17 dc_hfeol_m10  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
       temp, halfeol_int, gainHALFEOL, -10);
18 dc_hfeol_m20  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
       temp, halfeol_int, gainHALFEOL, -20);
19
20
21 % EOL
22 eol_int = rad.eol.dark.integration(1);
23 dc_eol_20  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
       eol_int, gainEOL, 20);
24 dc_eol_10  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
       eol_int, gainEOL, 10);
25 dc_eol_0   = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
       eol_int, gainEOL, 0);
26 dc_eol_m10  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
       eol_int, gainEOL, -10);
27 dc_eol_m20  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
       eol_int, gainEOL, -20);
28
29
30 % Double EOL
31 eolx2_int = rad.eolx2.dark.integration(1);
32 dc_x2_20  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
       eolx2_int, gainEOLx2, 20);
33 dc_x2_10  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
       eolx2_int, gainEOLx2, 10);
34 dc_x2_0   = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
       eolx2_int, gainEOLx2, 0);
35 dc_x2_m10  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
       eolx2_int, gainEOLx2, -10);
36 dc_x2_m20  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
       eolx2_int, gainEOLx2, -20);
37
38 % Series format for plot
39 dc_20  = [dc_pre_20,  dc_hfeol_20,  dc_eol_20,  dc_x2_20];
40 dc_10  = [dc_pre_10,  dc_hfeol_10,  dc_eol_10,  dc_x2_10];
```

```
41 dc_0   = [dc_pre_0,   dc_hfeol_0,   dc_eol_0,   dc_x2_0];
42 dc_m10  = [dc_pre_m10,  dc_hfeol_m10,  dc_eol_m10,  dc_x2_m10];
43 dc_m20  = [dc_pre_m20,  dc_hfeol_m20,  dc_eol_m20,  dc_x2_m20];
```

Listing 4.14: Code for collecting dark current values at the target temperatures.

Implementing the same framework structure as in Listing 4.9, the variables (frames, temperatures , integration times, gain values, and target temperatures) are run through the pre-defined function ('nearest_dc_at'), and in this case, they are stored based on their radiation dose and associated temperature. Lines 39 - 43 of Listing 4.14, build a 1 x 4 vector and arrange the dark current values in order of ascending proton fluence (radiation dose).

All of this was now culminated together to produce the dark current against proton fluence plot, as is portrayed in Listing 4.15.

```
1 %%%  Plot %%%
2 figure('Name','Dark Current Vs Proton Fluence','NumberTitle','off')
3
4 % Make a fine grid of fluences for smooth curve
5 fine_fluences = linspace(min(fluences), max(fluences), 200);
6 % Apply spline interpolation
7 dc20_spline = spline(fluences, dc_20, fine_fluences);
8 dc10_spline = spline(fluences, dc_10, fine_fluences);
9 dc0_spline = spline(fluences, dc_0, fine_fluences);
10 dcm10_spline = spline(fluences, dc_m10, fine_fluences);
11 dcm20_spline = spline(fluences, dc_m20, fine_fluences);
12
13
14 % Plot smooth spline lines
15 semilogy(fine_fluences, dc20_spline,  '-', 'LineWidth',0.5, '
   DisplayName','20  C '); hold on
16 semilogy(fine_fluences, dc10_spline,  '-', 'LineWidth',0.5, '
   DisplayName','10  C ');
17 semilogy(fine_fluences, dc0_spline,  '-', 'LineWidth',0.5, '
   DisplayName','0  C ');
18 semilogy(fine_fluences, dcm10_spline,  '-', 'LineWidth',0.5, '
   DisplayName','-10  C ');
19 semilogy(fine_fluences, dcm20_spline,  '-', 'LineWidth',0.5, '
   DisplayName','-20  C ');
20
21
22 % Plot original fluences as markers
23 semilogy(fluences, dc_20,  'o', 'MarkerSize',6, 'MarkerFaceColor','
   auto', 'DisplayName','20  C ');
24 semilogy(fluences, dc_10,  'o', 'MarkerSize',6, 'MarkerFaceColor','
```

```
    auto', 'DisplayName','10  C ');
semilogy(fluences, dc_0,   's', 'MarkerSize',6, 'MarkerFaceColor','
    auto', 'DisplayName','0  C ');
semilogy(fluences, dc_m10,  'o', 'MarkerSize',6, 'MarkerFaceColor',
    'auto', 'DisplayName','-10  C ');
semilogy(fluences, dc_m20,  'o', 'MarkerSize',6, 'MarkerFaceColor',
    'auto', 'DisplayName','-20  C ');


% Vertical lines: EOL and 1.2x EOL (lifetime requirement point)
xline(EOL_fluence,   '--', 'EOL','LabelOrientation','horizontal');
xline(lifetime_flux, '--', '1.2   EOL','LabelOrientation','
    horizontal');

% Horizontal line: desired limit at 0 C
yline(desired_limit_0C, ':', 'Desired 0  C  limit', '
    LabelHorizontalAlignment','left', 'Color','r');

grid on
xlabel('Proton fluence (p cm^{-2})');
ylabel('Dark current (e^{-}\cdot pixel^{-1}\cdot s^{-1})', '
    Interpreter','tex');
title('Dark current vs proton fluence at 20  C , 10  C , 0  C , -10
     C , -20  C ');
legend('Location','southeast');

%%% Labelling EOL Points %%%
dc20_EOL = dc_20(fluences == EOL_fluence);
dc10_EOL = dc_10(fluences == EOL_fluence);
dc0_EOL  = dc_0(fluences == EOL_fluence);
dcm10_EOL = dc_m10(fluences == EOL_fluence);
dcm20_EOL = dc_m20(fluences == EOL_fluence);


Annotations = {
    sprintf('20  C : %.2e e^- pixel ^{-1} s ^{-1}', dc20_EOL),
    sprintf('10  C : %.2e e^- pixel ^{-1} s ^{-1}', dc10_EOL),
    sprintf('0  C : %.2e e^- pixel ^{-1} s ^{-1}', dc0_EOL),
    sprintf('-10  C : %.2e e^- pixel ^{-1} s ^{-1}', dcm10_EOL),
    sprintf('-20  C : %.2e e^- pixel ^{-1} s ^{-1}', dcm20_EOL) };


% Text Positions
ys = [dc20_EOL, dc10_EOL, dc0_EOL, dcm10_EOL, dcm20_EOL];
xs = repmat(EOL_fluence,1,numel(ys));
```

```
62 text(xs*1.05, ys, Annotations, 'VerticalAlignment','bottom','
   FontSize',10);
```

Listing 4.15: Code for producing dark current Vs proton fluence plot.

In order to create smooth curves for this proton fluence plot, interpolation had to be done. Line 5 of Listing 4.15 essentially finds the minimum and maximum value of proton fluence across the dark currents and divides that specified range into 200 evenly spaced points. Hence, a smooth curve that fits the data could now be generated. Using the spline function (a piecewise cubic), interpolation occurs to find in-between values, allowing the curve to be much smoother than it would otherwise be with only 4 data points. This interpolation process was iterated for the dark currents at each temperature. Overlaid on top of these smooth curves are the original 4 dark current points at their respective proton fluences.

To enhance the readability of the graph and present more information on the same plot, two vertical lines were overlaid onto the plot, with one illustrating the proton fluence at EOL and the other at the 20% safety margin threshold (1.2 x EOL). Additionally, a horizontal line was plotted at 2000 DN to indicate the desired dark current threshold at 0 °C. This enables easy identification of whether or not the CCDs performance at EOL was satisfactory in accordance with the expected mission objectives.

After this, the EOL points were labelled according to their temperatures, with small text beside each data point indicating the EOL dark current at that specific temperature.

4.3 In-Flight Mission Data Approach

As mentioned in Subsection 3.2.1, a smaller dataset was used for getting to grips with the in-flight data. However, once the full dataset was received, this data was used for all further data extraction. Upon retrieving the in-flight dataset, it was discovered that due to hardware limitations, the file size was too large (too many frames) to load into MATLAB as one dataset and perform the necessary processes. To work around this issue, a code was developed to extract the raw frames at a set temperature, thus organising the frames into smaller, more manageable datasets that are more usable for the purpose of this study. The first part of this code was to create a loop to find frames within the range of target temperatures as seen in Listing 4.16. For full code, see Appendix F.

```
1 %% Load the UVIS data from the HDF5 file
2 fnm = 'dark_frames_raw_combined_80_compressed.h5';
3 FrameTime = h5read(fnm,'/ObservationalData/FrameTime');
        % Frame time in seconds
4 FrameTemp = h5read(fnm,'/ObservationalData/FrameFitTEMP');
    % Frame temperature in degrees C
```

```
5  Integration = h5read(fnm,'/ObservationalData/Integration');
       % Exposure time
6
7  % create loops for multiple temperatures
8  user_temperatures = (-15:5:25);
9  delta_temperature = 0.5;
10 for i = 1:numel(user_temperatures)
11     RawFrames = getframes(fnm,user_temperatures(i),
       delta_temperature,FrameTemp,FrameTime,Integration);
12 end
```

Listing 4.16: Code for looping through the desired temperature range.

As seen from Listing 4.16, the full dataset was loaded in, along with the necessary variable being read in from the main file. The 'user_temperatures' variable was then created to cycle through our desired temperatures. In this case the chosen temperature range was -15 °C to +25 °C in increments of 5 °C, hence the full range was -15°C, -10°C, -5°C, 0°C, 5°C, 10°C, 15°C, 20°C and 25°C. It is to be noted that these temperatures do not align with those stated upon ground testing as previous assessment of the smaller in-flight dataset illustrated that temperatures such as -20 °C, -25 °C, and -30 °C are never actually reached during the mission. The 'delta_temperature' (change in temperature) was set to 0.5, meaning that a tolerance of ± 0.5 °C was given for each temperature. Lines 10 to 12 in Listing 4.16 creates a for loop to iterate through all of the temperatures within the previously specified range, considers the ± 0.5 °C tolerance, and checks if there are any frames that meet this criteria.

A function for retrieving and extracting these frames into separate, smaller subsets of data was created. This frame extraction operation is demonstrated in Listing 4.17. This section of code was received from the Open University (Mason, 2025).

```
1  function RawFrames = getframes(fnm,user_temp,deltaT,temps,times,
       intT)
2      info = h5info(fnm, '/ObservationalData/RawFrames');
3      full_size = info.Dataspace.Size;
4      X = full_size(1);
5      Y = full_size(2);
6      Z = full_size(3);
7      ix = abs(temps - user_temp);
8      ii = find(ix < deltaT);
9      FrameTemp = temps(ii);
10     FrameTime = times(ii);
11     Integration = intT(ii);
12
13     total_number_of_frames = numel(ii);
14
```

```
15      RawFrames = zeros(X,Y,total_number_of_frames);
16      ProcessedFrames = zeros(X,Y,total_number_of_frames);
17      iframe = 1;
18      nframes = 3000;
19      num_slices = ceil(Z / nframes);
20      p = 1;
21      for i = 1:num_slices
22          if (iframe + nframes) > Z
23              nframes = Z - iframe;
24          end
25          nframe = iframe - 1 + nframes;
26          iTemp = temps(iframe:nframe);
27          ix = abs(iTemp - user_temp);
28          selected_frames = find(ix < deltaT);
29          if isempty(selected_frames) == 0
30              q = (p-1) + numel(selected_frames);
31              dat = h5read(fnm,'/ObservationalData/RawFrames', [1 1
    iframe], [X Y nframes]);
32              RawFrames(:,:,p:q) = dat(:,:,selected_frames);
33              clear dat
34              dat = h5read(fnm,'/ObservationalData/ProcessedFrames',
    [1 1 iframe], [X Y nframes]);
35              ProcessedFrames(:,:,p:q) = dat(:,:,selected_frames);
36              p = q + 1;
37          end
38          iframe = iframe - 1 + nframes;
39      end
40      % save the data into a matlab file for easier access
41      if user_temp < 0
42          sfnm  = ['Frames_at_m',num2str(abs(user_temp)),'C.mat'];
43      else
44          sfnm  = ['Frames_at_p',num2str(abs(user_temp)),'C.mat'];
45      end
46      save(sfnm,"FrameTime","FrameTemp","Integration","RawFrames","
    ProcessedFrames","-v7.3")
47  end
```

Listing 4.17: Code for extracting frames at the desired temperature range.

A function was created called '**getframes**' which feeds in the following inputs that have been pre-defined: the flight data file, the desired temperatures, the temperature error margin, the times of frame capture, and the integration (exposure) time of the frames. The output of this was then stored in the '**RawFrames**' variable.

Within the function, the metadata about the raw frames are stored in '**info**' and then the dimensions of the dataset are extracted. This is seen in lines 2 to 6 of Listing 4.17,

whereby '**x**' looks at the pixels along the rows, '**y**' looks at the pixels along the columns, and '**z**' looks at the total number of frames. Now these frames have been analysed, the **'absolute'** function was used to find frames close to the specified temperature, taking into account the tolerance of ± 0.5 °C, and stores them in '**ix**', then '**ii**' keeps the indices of these specific frames stored. Using the frames stored in '**ii**', their respective frame temperatures, frame times and integration times are fetched and stored.

Using the '**numel**' function, the total number of frames in '**ii**' was stored for use in the next part of the code. '**RawFrames**' and '**ProcessedFrames**' arrays are created and then filled entirely with the numerical value of zero; hence, an example array may now look like this [0, 0, 0]. By setting all values of the array to zero, this pre-allocates the memory necessary for when data was stored in these arrays at a later stage. The variable '**iframe**' was set to the value of one, initiating the code to start from the first frame. The '**nframes**' variable indicates how many frames were going to be loaded in at once. In this case, 3000 frames was chosen as the highest number of frames, due to processing power limitations. Using a higher number of frames would not be feasible. The specified number of frames was then separated into smaller, more manageable segments ('**numslices**'). Within '**numslices**', the total number of frames was divided by the previously specified '**nframes**', which provides the number of loops needed to cover the full dataset. This was all encapsulated within the '**ceil**' function to ensure that the final number was rounded up to allow for the entire dataset to be looped through. Additionally, the variable 'p' was set to the value of one, which was the initial position used later for saved the raw and processed frames.

A for loop was then created to loop through all of the dataset in segments of 3000 frames. Enclosed within this for loop, an if loop was configured to reduce the value of '**nframes**' if the remaining number of frames in the last segment was less than 3000 frames. By doing this, the data retrieved will end exactly at the last frame, rather than having empty datasets of frames that do not exist. Lines 25 to 28 in Listing 4.17 define the selection parameters of the desired frames. The '**nframe**' variable ensures the limit of frames looked at was the 3000th (the last in this loop), and then the temperatures for these frames are extracted and stored in the '**iTemp**' variable. Next, the difference between the actual frame temperature and the target temperatures were calculated and stored in '**ix**', and then which ever frames were within the ± 0.5 °C of the target temperatures were stored in '**selected_frames**'.

Another if loop was created to now store all of these frames in the pre-allocated '**RawFrames**' and '**ProcessedFrames**' arrays. Line 29 of Listing 4.17 acts as fail-safe mechanism, stating that the next section of code must only be run if the frame temperatures are within the specified error margin range. The save index was then updated via

the usage of defining 'q', which essentially allocated the correct number of element to the correct array and then changed the overall number for the next iteration of the for loop, so that the existing frames are not overwritten. An example of this save index could be that 15 frames in the first 3000 have been identified close to that temperature, '**q**' will be equal to 15, and then the next iteration will start with '**p**' being equal to 16. The raw frames and processed frames were then read in from the main file in increments of 3000; the '**selected_frames**' are then extracted in stored in the corresponding '**RawFrames**' or '**ProcessedFrames**' array. The 'iframe' variable was then updated to a value of 3000, so that when restarting the loop, the next 3000 frames can be analysed.

Moreover, the final section of this code saves the selected frames according to user temperatures in separate MATLAB files, which can be accessed individually and integrated with other sections of code used for plots. In these files, the relevant metadata is stored as well, such as frame time, temperature of frame, integration time, and whether or not it is a raw or processed frame.

4.3.1 Plotting Dark Current against Temperature over the Mission Duration

For the in-flight data, an Arrhenius plot was chosen against due to it not being able to clearly reflect the evolution of the CCDs' behaviour over the longer duration of time. As a result, a linear plot of dark current against temperature was chosen, but with colour grading to portray the duration of years for the corresponding dark currents at their specified temperatures. The process taken to generate the dark current against temperature plot can be seen within Listing 4.18 (Mason, 2025).

```
close all
clear all


%% Load the UVIS mean DC from the flight data
fnm = 'UVIS_dark_frames_mean.h5';
FrameTime = h5read(fnm,'/ObservationalData/FrameTime');
       % Frame time in seconds
FrameTemp = h5read(fnm,'/ObservationalData/FrameTEMP');
       % Frame temperature in degrees C
MeanDC = h5read(fnm,'/ObservationalData/MeanDarkCurrent');
      % Exposure time
%% Dark Current Vs Temperature Plot with increase in time

gain = 7.28;
ntime = (FrameTime - FrameTime(1)) / (365*24*3600);
```

```
14 figure()
15 scatter(FrameTemp,MeanDC*gain,6,ntime,"filled") % use ntime to get
      the color to show time
16 xlabel('Temperature (^{\circ}C )','FontSize',12)
17 ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})','FontSize',12)
18 title ('Dark Current Vs Temperature Plot with increase in time')
19 cb = colorbar;
20 caxis([0,9])
21 ylabel(cb, 'time / years',  'FontSize',16)
```

Listing 4.18: Code for Plotting the In-Flight Dark current against temperature.

As per Listing 4.18, the in-flight data was loaded in a similar way to as shown in Listing 4.16. The key difference here was the particular file that was being loaded into MATLAB. The '**UVIS_dark_frames_mean.h5**' file contains 90256 frames, hence it is quite a large file in comparison to the earlier condensed set of 1064 frames. Due to the computational limits of dealing with such a large file, in order to find the mean dark current value for each frame, the dataset had to be run through supercomputers at the Open University, before it was compressed and returned for MATLAB analysis.

Through the use of another PTC plot, the gain for the in-flight CCDs used was estimated to be 7.28 e^{-}/DN. The '**ntime**' variable was used for finding the overall elapsed time from when the frame was taken to the beginning of the mission in years. This was achieved by subtracting '**FrameTime(1)**' (the time that the first frame was taken), from the entire array of frame times to establish the elapsed time. The outcome of this was in the unit of seconds; hence, to convert to years, the overall value was divided by $365 \times 24 \times 3600$. A scatter plot was then produced, calling in the inputs of frame temperature on the x-axis, the dark current adjusted for gain on the y-axis, and the elapsed time of the mission represented as different coloured data points. Lines 19 to 21 of Listing 4.18 add a colour bar along the secondary y-axis to indicate the scale of colours representing the different years.

4.3.2 Plotting Dark Current against Mission Duration

Due to it being quite difficult to distinguish the exact values of dark current from such a populated scatter plot of dark current against temperature over the duration of the mission, a secondary plot was generated with just the mean dark current values over the mission duration at 0°C. 0°C was chosen due to commonality between the datasets, but also due to most of the data being concentrated around this temperature point. This allowed for easier identification of trends upon investigating the dark current behaviour of the in-flight data. Before the plot could be generated, some filtering and time increment had to be done as reflected in Listing 4.19.

```
%% Filter flight data
% Create mask for non-zero dark current values
non_zero_dc_mask = (MeanDC > 0);

% Apply the filter to all flight data arrays
MeanDC_nonzero = MeanDC(non_zero_dc_mask);
FrameTemp_nonzero = FrameTemp(non_zero_dc_mask);
ntime_nonzero = ntime(non_zero_dc_mask);
FrameTime_nonzero = FrameTime(non_zero_dc_mask);

fprintf('Original data points: %d\n', length(MeanDC));
fprintf('Data points with DC > 0: %d\n', sum(non_zero_dc_mask));
fprintf('Zero dark current points removed: %d\n', sum(MeanDC == 0))
    ;
fprintf('Percentage with DC > 0: %.1f%%\n', 100 * sum(
    non_zero_dc_mask)/length(MeanDC));

%% Time increment analysis
years = (0:0.5:9); % half a year windows
point = zeros(length(years)-1, 2);

for i = 1:numel(years)-1
    i1 = find(ntime_nonzero > years(i));
    i2 = find(ntime_nonzero(i1) < years(i+1));

    if ~isempty(i2)
        i3 = i1(i2);
        bdark{i} = MeanDC_nonzero(i3);
        btemp{i} = FrameTemp_nonzero(i3);

        figure()
        plot(btemp{i}, bdark{i}*gain, 'x')
        xlabel('Temperature ( C )')
        ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
        title(sprintf('Time Window: %.1f-%.1f years (DC > 0)', years
    (i), years(i+1)))

        if length(btemp{i}) > 3
            f = ezfit(btemp{i}, bdark{i}*gain, 'y = a * exp (b * x)
    + c', [10, 0.1, 10]);
            showfit(f)
            point(i,1:2) = [years(i), evalfit(f,0)];
        else
            point(i,1:2) = [years(i), NaN];
        end
```

```
    else
        point(i,1:2) = [years(i), NaN];
    end
end

% Remove NaN values
valid_idx = ~isnan(point(:,2));
point_clean = point(valid_idx,:);
```

Listing 4.19: Code for filtering dark current and performing time increment analysis.

The mean dark current data had to be filtered to only include positive values above 0 $e^{-}/pixel/s$. This proves advantageous when fitting the data, as the zero dark current value will not skew the fit. The filter was then applied to all arrays so that they would all match in size and have no values related to the zero dark current. This was then checked and printed in the command window to ensure that the filter was applied correctly.

Time increment analysis was then performed, whereby the vector '**years**' was created to define the boundaries of 0 to 9 years in intervals of half a year. The variable '**point**' then preallocates an empty array to hold the starting year of each measurement as well as the respective dark current value. A for loop was developed to loop over the specified year range in the half a year increments, with '**i**' being the index number for each half a year interval. By looking through the '**ntime_nonzero**' array, if the values were greater than the minimum value for the time interval that was about to be looked over, then it was stored in '**i1**'. If the values are less than the maximum value for the time interval that was about to be looked over, then it was stored in '**i2**'. For example, if the time interval was between 2 and 2.5 years, then '**i1**' will look at the various times above 2 years, and '**i2**' will look at the times in '**i1**' and only extract those less than 2.5 years.

The if loop then checks whether the '**i2**' is empty. If data was found within the array then the final indices are extracted into '**i3**'. The corresponding dark current and frame temperature for the data in '**i3**' was stored in '**bdark{i}**' and '**btemp{i}**' respectively. From this, a figure was created for each time interval, plotting dark current along the y-axis and temperature along the x-axis. An exponential curve was then fitted to each of these plots at x = 0 °C on the condition that more than three data points exist. If the data has three or less data points, a good fit cannot be produced hence, these points are stored separately and removed from the plot to ensure a successful fit for the other data. By having multiple plots of dark current behaviour in half-year intervals, any trends or anomalies can be identified and pinpointed quicker than looking through the entire dataset. These plots can be seen in Appendix G.

In addition to this, a plot was created to plot the dark current estimated from these fits against the mission duration to show the year by year accumulation of dark current

onboard ExoMars TGO. The process for this plot is demonstrated in Listing 4.20.

```
%% Dark current at 0 C  Vs Mission Duration (using only DC > 0 data
   )
figure()
plot(point_clean(:,1), point_clean(:,2), 'x')
xlabel('Time (Years)')
ylabel('Dark Current at 0 C  (e{^-} pixel^{-1} s^{-1})')
title('Dark current at 0 C  Vs Mission Duration')
grid on
```

Listing 4.20: Code for filtering dark current against mission duration.

As seen in Listing 4.20, a new figure was generated to plot the dark current from the previously fitted data at the specified time intervals. A linear fit of the form $ax+b$ was applied using the EZyFit package in MATLAB, which allows for the assessment of dark current at zero and the growth rate of dark current per year.

4.4 Data Comparison Approach

In order to compare the two sets of data and their relative dark currents, a scatter plot was created which combined the in-flight data over the 9 years with the ground testing data at their varied radiation doses. The algorithm is detailed in Listing 4.21. This section of code was provided by the Open University (Mason, 2025).

```
close all
clear all
%% Load the UVIS mean DC from the flight data
fnm = 'UVIS_dark_frames_mean.h5';
FrameTime = h5read(fnm,'/ObservationalData/FrameTime');
       % Frame time in seconds
FrameTemp = h5read(fnm,'/ObservationalData/FrameTEMP');
       % Frame temperature in degrees C
MeanDC = h5read(fnm,'/ObservationalData/MeanDarkCurrent');
      % Exposure time

%%  Load the ground data
load('uvis_ground_data_final_1.mat')
rad = gnd.em.irradiation;
csl = gnd.csl;

gain = 7.28;
ntime = (FrameTime - gnd.csl.ephemeris_time) / (365*24*3600); %
   Time Normalised

```

```
% Filter CSL data for temperatures >= 3 degrees and early mission
    data (<1 year)
i3 = find(csl.MeanTemp >= 3);
ii = find(ntime < 1.0);
Temp_0 = [csl.MeanTemp(i3), FrameTemp(ii)'];
DC_0 = [csl.MeanDC(i3), MeanDC(ii)'];

% Combine both CSL datasets for fitting (normal + filtered)
combined_temp = [csl.MeanTemp, csl.MeanTemp(i3)];
combined_dc = [csl.MeanDC, csl.MeanDC(i3)];

% Fit the combined data to get a middle-ground fit
ffit_combined = ezfit(combined_temp, combined_dc, 'y = a * exp (b *
    x) + c', [11, 0.1, 0.7]);
Yfit_combined = ffit_combined.m(1) * exp(ffit_combined.m(2)
    *(-20:2:24)) + ffit_combined.m(3);
```

Listing 4.21: Code for loading in both ground and in-flight data.

The in-flight data was loaded in the same way as previously; however, the ground data was loaded in slightly differently. The ground data loaded in this instance was the '**uvis_ground_data_final_1.mat**', which was a larger file consisting of more data points and calibration data in the form of data from the Center Spatial De Liège (CSL); who had a hand in the development of the Nomad UVIS. The engineering model irradiation data was loaded in from this file, as well as the CSL data. The calibration gain for the in-flight CCD was specified to be 7.28 e^{-}/DN and then the time frame used was normalised ('**ntime**'). '**ntime**' was calculated by subtracting the ground data ephemeris time from the in-flight data frame time to get the time difference between each in-flight measurement and the ground calibration data. Ephemeris was used specifically as this gives reference to J2000 (2000-Jan-01 12:00:00 TT), which was used as a reference for all space missions to allow the measurements to line up accurately and be on the same time scale. The difference between these two times was then divided by $365 \times 24 \times 3600$ to convert from seconds to years.

Due to pre-observation of the CSL data, it was adjusted by applying a filter. The CSL data at temperatures below 3 °C have been filtered out to get rid of the noisy data. A secondary filter was then applied for all of the points less than a year old. Then these filtered values are combined within 'Temp_0' and 'DC_0' to produce a baseline ground value at 0 years. This filtered data was then combined with the normal CSL data equivalent values to produce a non-biased fit that considers all the CSL data. This fitted data can be used as calibration data to provide a reference of what the dark current values were like just before/ at the point of launch in 2016.

Listing 4.22 demonstrates the algorithm produced for the pre-characterisation of ground data (Mason, 2025).

```
%% pre-characterisation
pre = gnd.em.pre;
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det1 = get_gnd_pre_meanDC(pre.det1,r1,r2,c1,c2);
mDC_det2 = get_gnd_pre_meanDC(pre.det2,r1,r2,c1,c2);
mDC_det3 = get_gnd_pre_meanDC(pre.det3,r1,r2,c1,c2);
%%% get the dark current from halfeol from expose and non-exposed
    region.
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det1_halfeol = get_gnd_pre_meanDC(rad.halfeol.dark,r1,r2,c1,c2)
    ;
mDC_det1_halfeol_notexposed = get_gnd_pre_meanDC(rad.halfeol.dark,
    r1,r2,50,500);
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det2_eol = get_gnd_pre_meanDC(rad.eol.dark,r1,r2,50,500);
mDC_det2_eol_notexposed = get_gnd_pre_meanDC(rad.eol.dark,r1,r2,c1,
    c2);
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det3_eolx2 = get_gnd_pre_meanDC(rad.eolx2.dark,r1,r2,c1,c2);
mDC_det3_eolx2_notexposed = get_gnd_pre_meanDC(rad.eolx2.dark,r1,r2
    ,50,500);
mDC_fm = get_gnd_fm_meanDC(gnd.fm.det1);
figure()
plot(pre.det1.temp,mDC_det1,'xk')
hold on
plot(pre.det2.temp,mDC_det2,'xb')
plot(pre.det3.temp,mDC_det3,'xr')
xlabel('Temperature (^{\circ}C )')
ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
```

Listing 4.22: Code for pre-characterisation of ground data.

As per Listing 4.22, the pre-irradiation data was loaded first with the focus on a specific ROI (Rows 30 to 230 and Columns 600 to 1000) due only half the CCD effective area being exposed to the radiation as per Chapter 3. Over this ROI, the mean dark current was found for detectors 1, 2, and 3. Using the same ROI, the half EOL data was loaded in, however, the mean dark current was calculated for both the exposed and non exposed regions of detector 1. The same process was followed for EOL and double EOL with

their respective detectors. This was then plotted to ensure that the data had been loaded in correctly and to check that the values were correct before moving on to the main scatter plot.

As mentioned earlier, a scatter plot was produced to combine the in-flight and ground testing datasets. The process to produce this comparison plot can be seen in Listing 4.23 (Mason, 2025).

```
%% Scatter Plot for In-Flight and Ground Data
figure()
scatter(FrameTemp,MeanDC*gain,6,ntime,"filled") % use ntime to get
    the color to show time
hold on
scatter(rad.halfeol.dark.temp,mDC_det1_halfeol_notexposed*rad.
    halfeol.ptc.gain1,12,"c",'o','filled')
scatter(rad.halfeol.dark.temp,mDC_det1_halfeol*rad.halfeol.ptc.
    gain1,12,"r",'o','filled')
scatter(rad.eol.dark.temp,mDC_det2_eol_notexposed*rad.eol.ptc.gain1
    ,12,"g",'o','filled')
scatter(rad.eol.dark.temp,mDC_det2_eol*rad.eol.ptc.gain1,12,"m",'o'
    ,'filled')
scatter(rad.eolx2.dark.temp,mDC_det3_eolx2_notexposed*rad.eolx2.ptc
    .gain1,12,"k",'s','filled')
scatter(rad.eolx2.dark.temp,mDC_det3_eolx2*rad.eolx2.ptc.gain1,12,"
    k",'o','filled')

plot (combined_temp,combined_dc*gain,'x')
plot((-20:2:24),Yfit_combined*gain,'m') % Combined fit


xlabel('Temperature (^{\circ}C )')
ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
cb = colorbar;
caxis([0,9])
ylabel(cb,'Time (years)','FontSize',16)
title ('Dark Current Vs Temperature Plot with increase in time')
legend('Flight Data', 'Half EOL Not Exposed', 'Half EOL Exposed', '
    EOL Not Exposed', ...
        'EOL Exposed', 'EOLx2 Not Exposed', 'EOLx2 Exposed', '
    Combined CSL', 'Combined CSL Fit', 'Location', 'best')
set(gca,'FontSize',12)
'stop'

function mDC = get_gnd_pre_meanDC(det,r1,r2,c1,c2)
    for i = 1:size(det.frames,3)
        frames = det.frames(r1:r2,c1:c2,i);
```

```
30          mDC(i) =  mean(frames(:)) / det.integration(1);
31      end
32  end
33  function mDC = get_gnd_fm_meanDC(det)
34      for i = 1:size(det.frames,3)
35          frames = det.frames(90:230,50:1030,i);
36          mDC(i) =  mean(frames(:)) / det.integration(i);
37      end
38  end
```

Listing 4.23: Code for producing combined scatter plot of both datasets.

The first set of data to be plotted was the mean dark current from the in-flight data that has been multiplied by the calibration value. This produces the same base plot as Listing 4.18. The '**hold on**' function was used in order to overlay the other plots onto the same figure. The ground data for both the irradiated and non-irradiated sections were plotted on the same figure to allow for a comparison of dark current over the range of temperatures. The fitted data is comprised of the CSL data and the filtered CSL data was plotted in order to calibrate all of the data from year zero upwards (year of launch).

Lines 27 to 36 of Listing 4.23 define two functions for computing the mean dark current values. Each function is comprised of a for loop to cycle through the relevant dataset. The first function stored the mean dark current per frame for the ground test data via utilisation of Equation 4.1. The same process is performed for all of the flight model detectors. Despite not being used within the constraints of the ground testing, they still prove useful for reference to compare the accuracy of dark current between the EM and the FM to see if the EM was an accurate enough representation.

Chapter 5

Data Characterisation and Findings

This chapter outlines the overall characterisation of CCD performance based upon both the ground test data and the in-flight data. From this characterisation, the effectiveness of the ground testing data can be assessed in comparison to the in-flight data.

5.1 Ground Test Dark Current

As outlined in Chapter 4, for the ground test data, two plots were produced : an Arrhenius for dark current against temperature and a plot for dark current against proton fluence. The plots generated within this section are meant to be representative of the plots produced in the Gow paper on proton radiation damage assessment (Gow et al., 2017) as the ground data provided by the Open University was from that specific study. Figure 5.1 illustrates the Arrhenius dark current against temperature plot.

Each line from the bottom upwards represents an increase in radiation dose as reiterated by the legend attached to the plot. Dark current sits in the y-axis in $e^-/pixel/s$ and inverse temperature sits on the x-axis in kelvin, ranging from $3.4K^{-1}$ to 4 K^{-1} (approx. 20 °C to - 23°C). As expected, at each dose of radiation, it is evident that as the inverse temperature increases, the magnitude of dark current present decreases. The temperature is seen as a dominant driver for dark current in this plot and is further emphasised by the same trend being present despite the increase in radiation dose.

For the half end-of-life (2.4 x 10^9 protons.cm^{-2}) and end-of-life doses (4.8 x 10^9 protons.cm^{-2}), the linear decrease is followed perfectly throughout the entire range of temperatures. However, at the double end-of-life dose (9.4 x 10^9 protons.cm^{-2}), there is a slight discrepancy in dark current. At all of the temperature points, the dark current decreases by a range of 50-60%; however, from the second data point to the third, the percentage decrease is much lower. The second data point along the double end-of-life line has a dark current value of 7144 $e^-/pixel/s$ at an inverse temperature of 3.53 K^{-1} (scaled

at 1000/Temperature (K)). The third point along the same line has a dark current value of 5017 $e^{-}/pixel/s$ at an inverse temperature of 3.66 K^{-1} (scaled at 1000/Temperature (K)). Thus, between these two points, there is a percentage decrease of approximately 30%.

Comparing this to the same plot present within the proton radiation damage assessment paper, the same shift in percentage change is observable, but on a much smaller scale. From the plot, it seems that the percentage change is essentially negligible, whereas the in Figure 5.1, this change seems to appear more distinct. A further discrepancy with Figure 5.1, is that there seems to be an anomalous point present within the control CCD data. Once again, comparing this to the proton radiation damage assessment paper, this anomaly simply is not present. Figure 5.2 without a linear fit provides easier visualisation of these discrepancies.

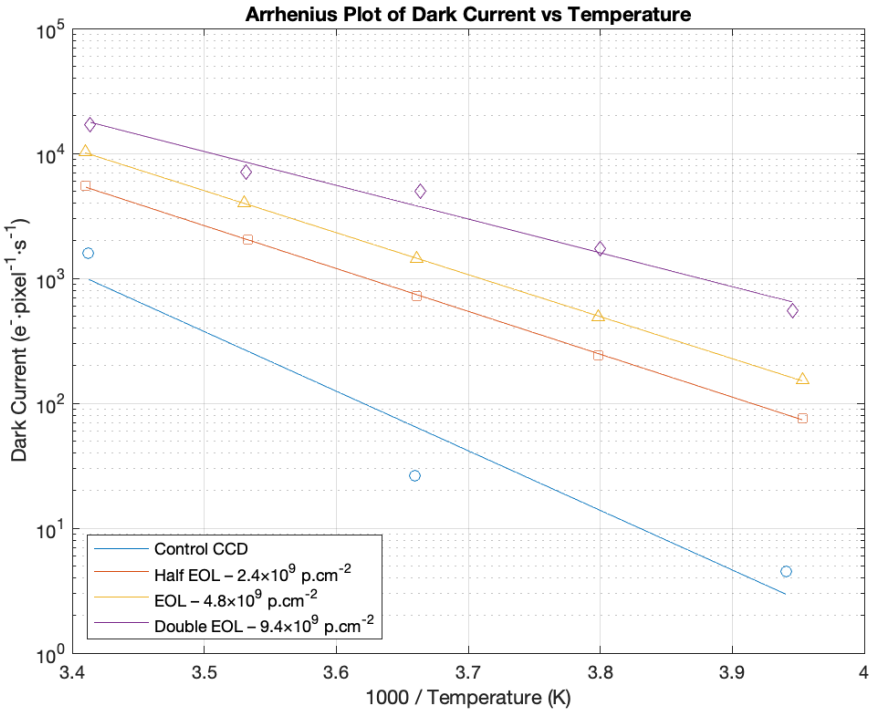


Figure 5.1: Ground Testing Arrhenius Dark Current Vs Temperature Plot with Linear Fits

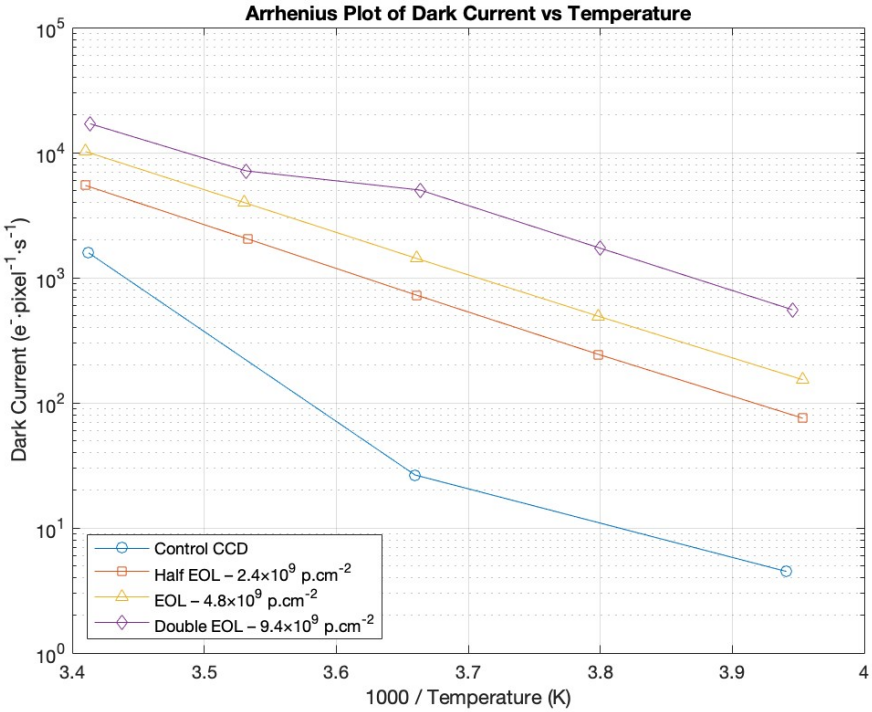


Figure 5.2: Ground Testing Arrhenius Dark Current Vs Temperature Plot.

A likely cause as to why there are discrepancies between the plots seen in the proton radiation damage assessment paper and these plots is that the gain value used for the CCDs are different. Overall, these results have lower dark current values in comparison to those exhibited in the paper. The calibration gain values used for the ground data was estimated using multiple PTCs; however, the exact gains for the devices used for ground testing were no provided by the Open University. This could signify that a larger gain was used in experimentation, or that there is an unknown factor that has not been taken into account when constructing these ground data plots.

As previously stated, the second plot produced for the ground test data was the dark current against proton fluence (radiation dose). This is depicted in Figure 5.3.

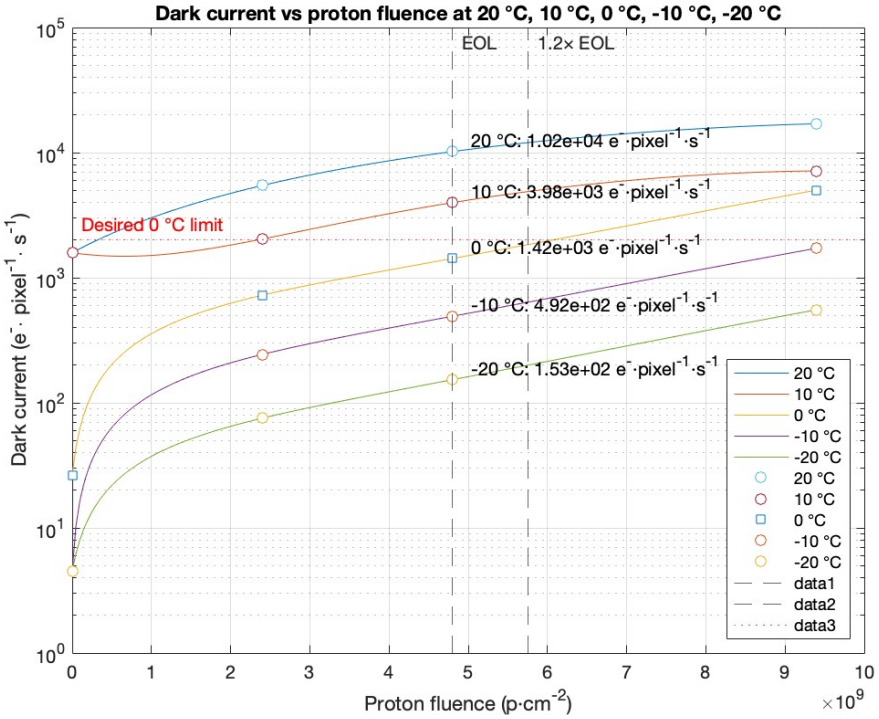


Figure 5.3: Ground Testing Dark Current Vs Proton Fluence

Figure 5.3 shows how dark current (as a function of proton fluence) evolves at the given temperatures. As is to be expected with CCD behaviour, as the radiation dose (damage caused) increases, the dark current present within the device also increases. Each line is a different temperature and was all sorted in ascending order. It is important to highlight that in Figure 5.3 as previously mentioned, there is data at five different temperatures, whilst the proton radiation damage assessment only uses three and excludes excessive temperatures from the plot. There are also dotted lines demonstrating the EOL and 1.2 times EOL radiation doses.

Moreover, at 0°C, the threshold for dark current has not been reached until after the EOL dose. The EOL equivalent dark current is $1424 e^{-}/pixel/s$ which is below the specified 2000 $e^{-}/pixel/s$. In theory, from a ground testing point of view, this means that the CCD will still perform as intended at EOL. Considering the 1.2 times EOL safety margin applied to testing, the 0°C has a dark current value will still be below the set threshold as it possesses a value of 1843 $e^{-}/pixel/s$. Up to this point, the CCD is compliant to the dark current threshold level, thus was likely to have been proven more flightworthy as a result. Above 1.2 times the EOL dose, the dark current is seen to dominate the dark frame due to the CCD attaining too much radiation damage; hence, the CCD can't be used, as the image from Nomad-UVIS at this point will be far too saturated.

As with the Arrhenius dark current against temperature plot, the dark current against

proton fluence plot also shows dark current values that are all below the prescribed values in the proton radiation damage assessment paper. As both plots have this deviation in dark current, this provides further evidence to suggest that the PTC gains are too low or that there is another factor that needs to be taken in account. At 10 °C and -10 °C on Figure 5.3, the dark current is seen to exhibit even further abnormal behaviour as they start at the same values as the upper and lower bound temperatures, respectively. Upon review of the code, the 10 °C and -10 °C data was processed using the same pipeline as the other temperatures. Therefore, this is likely to suggest that there is a lack of data for 10 °C and -10 °C at a proton fluence of 0 $e^{-}/pixel/s$ within the dataset, or that the data for those temperatures has not been stored correctly.

5.2 In-Flight Dark Current

As explained in Chapter 4, for the in-flight dataset, two plots were produced. The first plot investigating dark current vs temperature over the duration of the mission and the other, whilst the other analysed simply the dark current against the mission duration at 0 °C. Figure 5.4 conveys the first plot.

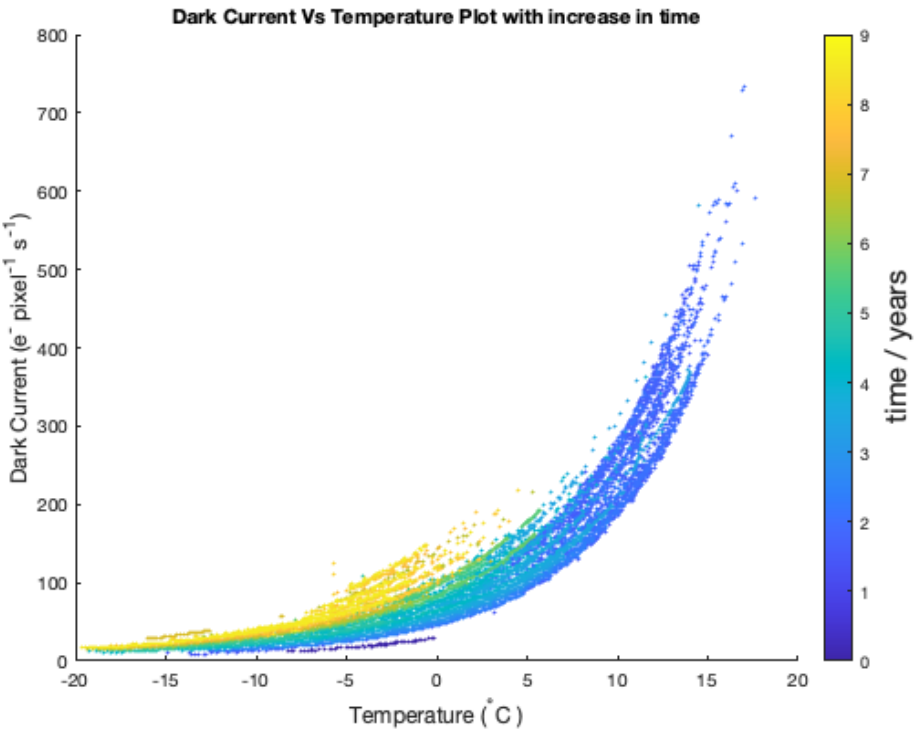


Figure 5.4: In-Flight Mission Dark Current Vs Temperature Plot

Figure 5.4 demonstrates all of the in-flight data in a colour-coded exponential curve.

As per the colour bar on the right-hand side, dark blue/blue values demonstrate data points from 0 to 2 years (2016 - 2018). The transition from light blue to light green is representing data points between 2 to 6 years (2018 - 2022). And finally, the light green to yellow is from 6 to 9 years (2022 - 2025). The overall trend exhibited here is the same as the ground data presented in Figures 5.1 and 5.2, just without the inverted temperature and a different fit. Due to the in-flight data consisting of so many years of data, it is difficult to observe a singular data point. As a result, dark current is looked at more holistically, as opposed to letting a singular point define the trend for a specific dark current and temperature. This plot takes 0 years to be from launch, hence from 2016. It is evident that the over dark current has increased over the space of nine years; however, the exact change is hard to distinguish without taking the mean values.

A clearer view of the in-flight dark current behaviour over time is exhibited in 5.5.

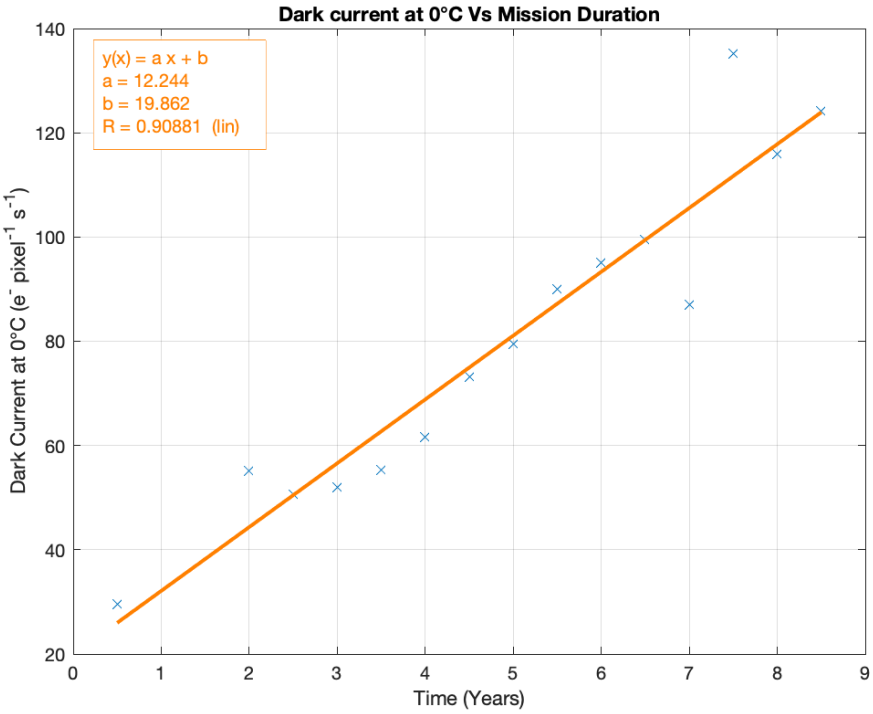


Figure 5.5: In-Flight Dark Current Vs Mission Duration Plot.

Through applying a linear fit to the data points within Figure 5.5, the linear increase of dark current (via accumulation) year upon year of the mission becomes very apparent. At launch (zero years), the fit estimates the dark current to be approximately 19.9 $e^-/pixel/s$ and is estimated to rise at a rate of 12.2 $e^-/pixel/s$ each year. It ought to be observed that there are outliers present within Figure 5.5, more notably at 2 years, 7 years and 7.5 years with dark current values of 55.1 $e^-/pixel/s$, 87.0 $e^-/pixel/s$ and 135.2 $e^-/pixel/s$, respectively. Years 2 and 7.5 could have been caused by an SEE, hence a sudden spike

in dark current levels. Additionally, year 7 could be due to the pixels annealing, hence the damage is masked, leading to a decrease in dark current. However, this may also not be the case as indicated by the steep 55.4% increase in the half a year following on from that.

5.3 Direct comparison of Ground Vs In-flight

Once again, as per Chapter 4, in order to directly compare the ground dataset and the in-flight dataset, a scatter plot was produced with each dataset overlaid and sorted according to the mission duration. The result of this comparison can be seen in Figure 5.6, with a magnified view showing the region of interest evident in Figure 5.7. The full code for this section can be seen in Appendix H.

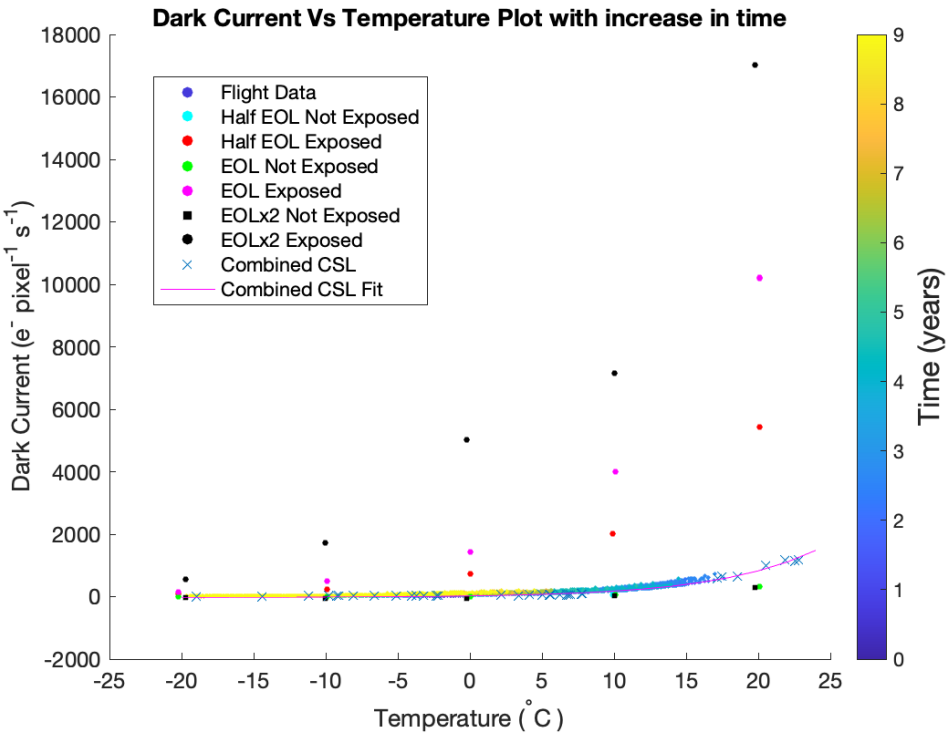


Figure 5.6: Combined Scatter Plot for In-Flight and Ground Data.

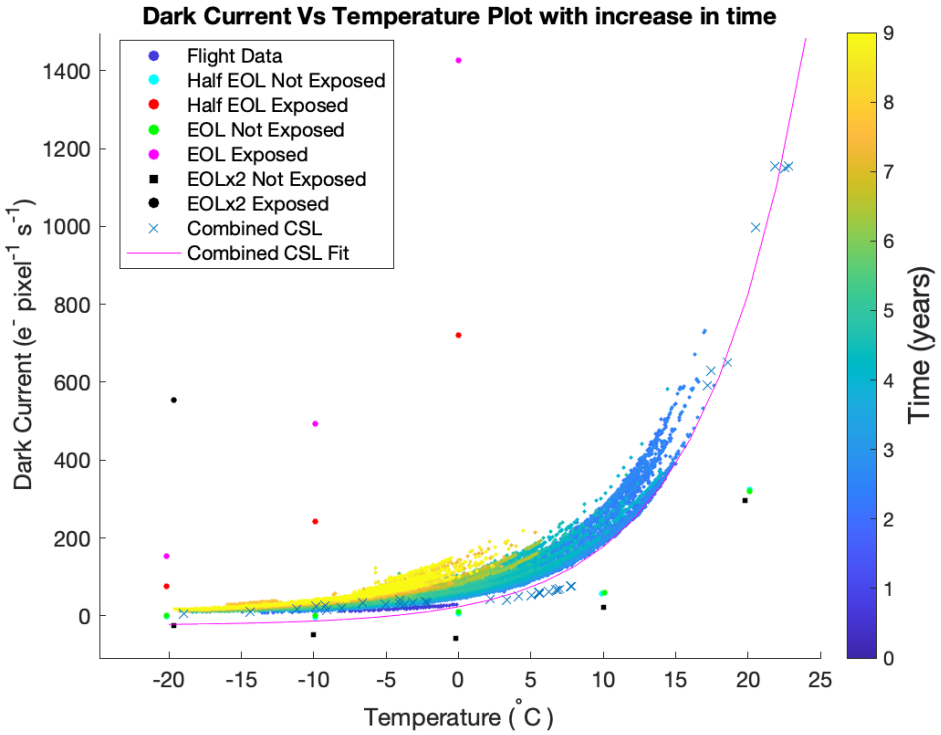


Figure 5.7: Enlarged ROI section of Combined Scatter Plot for In-Flight and Ground Data.

In Figure 5.6, the in-flight data is presented in the same way as Figure 5.4, with the ground data added as well. The data for half EOL, EOL, and double EOL is presented on this figure, as well as the exposed and non-exposed region for each dose. There is also a combined fit line, which is expanded upon later within this section. The data at each radiation dose produces an exponential fit, which is to be expected, as it demonstrates the accumulation of dark current over time. However, the ground data points appear to stray further and further away from the in-flight data with each incremental increase in temperature and produce much larger dark current values. This is likely indicative of an accelerated testing bias, in this case, where the radiation damage to the CCD is heightened during ground testing because it occurs over a much shorter duration than in-flight, resulting in considerably higher levels of dark current.

Additionally, this discrepancy in dark current essentially provides evidence towards radiation damage over a long period of time, not being equivalent to short and sharp radiation exposures. In-flight, the CCD could be annealing over time, as the CCD is not constantly being bombarded with massive doses of radiation; it fluctuates. Consequently, the damage caused in-flight will be less for the same radiation dose, as the in-flight CCD is not being irradiated continuously or in short bursts and isn't observed over a short period of time (like in ground testing). Further to this, the in-flight CCD goes through a thermal

cycle every observation (from -17 °C up to +10 °C), which during ground testing, the CCDs do not get the chance to do. This leads to questioning whether thermal cycling is actually masking the severity of the radiation damage.

Using 0 °C as a point of comparison, the dark current of both datasets can be compared as demonstrated in Tables 5.1 and 5.2.

Radiation Dose	Ground Data Dark Current
Half EOL (Approx. Equivalent to 2 years)	721.00 $e^-/pixel/s$
EOL (Approx. Equivalent to 4 years)	1425.37 $e^-/pixel/s$
Double EOL (Approx. Equivalent to 8 years)	5013.85 $e^-/pixel/s$

Table 5.1: Results table for dark current at each radiation dose for ground data (at 0 °C).

Mission Duration	In Flight Dark Current
2 Years	55.06 $e^-/pixel/s$
4 Years	61.63 $e^-/pixel/s$
8 Years	115.97 $e^-/pixel/s$

Table 5.2: Results table for dark current at each equivalent ground testing timestamps for in-flight data (at 0 °C).

Upon comparison of Tables 5.1 and 5.2, the dark current values are much higher than for the ground testing; more than a factor of 20 greater. The likely cause of this is partially due to the accelerated testing bias and the length of exposure, but more so due to a difference in radiation exposure. This drastic difference in dark current suggests that the radiation dose that Nomad-UVIS is exposed to through the duration of the mission is actually much lower than initially predicted when carrying out the ground testing. Therefore, this is likely to mean that the radiation environment around Mars is significantly less damaging than previously predicted (using the SPENVIS model).

Another cause to consider is the radiation shielding used. As per the ground testing, the model used for the irradiation of the CCD incorporated the shielding and was a factor considered upon deciding the estimated dose to use at EOL. If less shielding (in terms of thickness) was used in the ground test model, then more radiation is likely to have interacted with the CCD, thus giving rise to more damage. With that being the case, however, the dark current is still out by a significant factor in compared to the in-flight data, hence radiation shielding may only be an underlying factor within a bigger issue.

From Figure 5.7, the increase in dark current as the ExoMars TGO mission progresses becomes far more apparent. Moreover, the CSL fit used for the combined data points can be seen in the figure, represented by the magenta exponential line. The CSL data in this case can be used as a baseline for launch (year 0), as specified in Chapter 4. The double EOL non-exposed values were not used as part of the calibration for the plot, as they

present some negative values. This is likely to be an error in the dataset, hence it was ignored.

A significant aspect when interpreting this data is that the in-flight radiation doses are unknown. Tables 5.1 and 5.2 are able to provide an estimated comparison; however, due to having no dosimeter onboard the ExoMars TGO, there is no way to accurately tell what the radiation dose is per year of the mission.

Upon review of the combined CSL fit on Figure 5.7, there seems to be a gap in the dataset between 10 °C and 18 °C, and then a few points of data. After these points lie another short gap before the dark current spikes to much greater values. The points sat around the 1000 $e^-/pixel/s$ to 1200 $e^-/pixel/s$ mark are likely to be anomalous data, as the CSL data has not been exposed to any radiation. Hence, for analysis purposes, it was ignored. A very probable cause of this could be 'fake' noise in the CCD, better known as fixed pattern noise. Fixed pattern noise occurs due to defects present in the silicon upon manufacture of the device, causing issues when reading off dark current data at both low and high temperatures. The dark frames of the CSL data at cold and hot temperatures can be seen in 5.8 and 5.9, respectively.

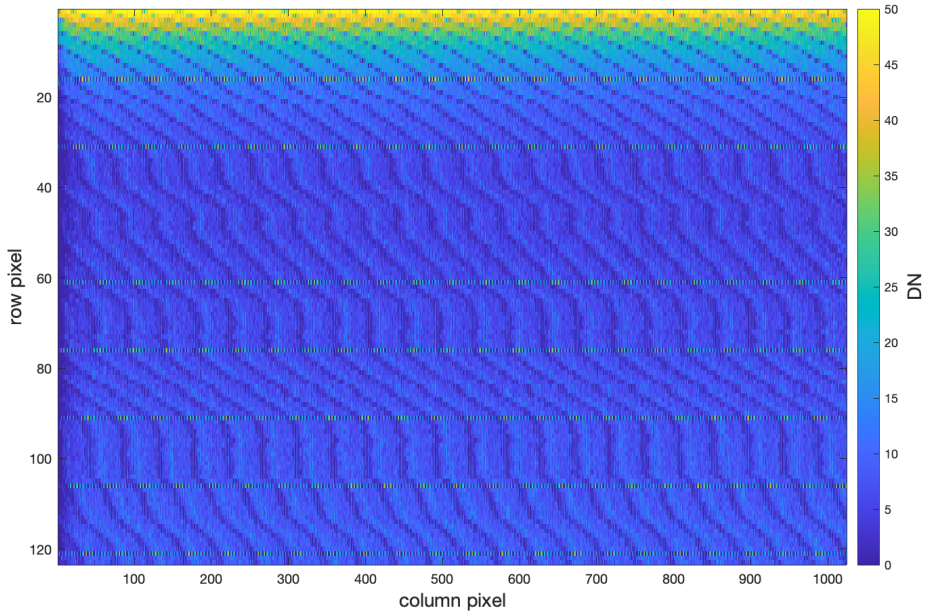


Figure 5.8: Cold Dark Frame for CSL Data

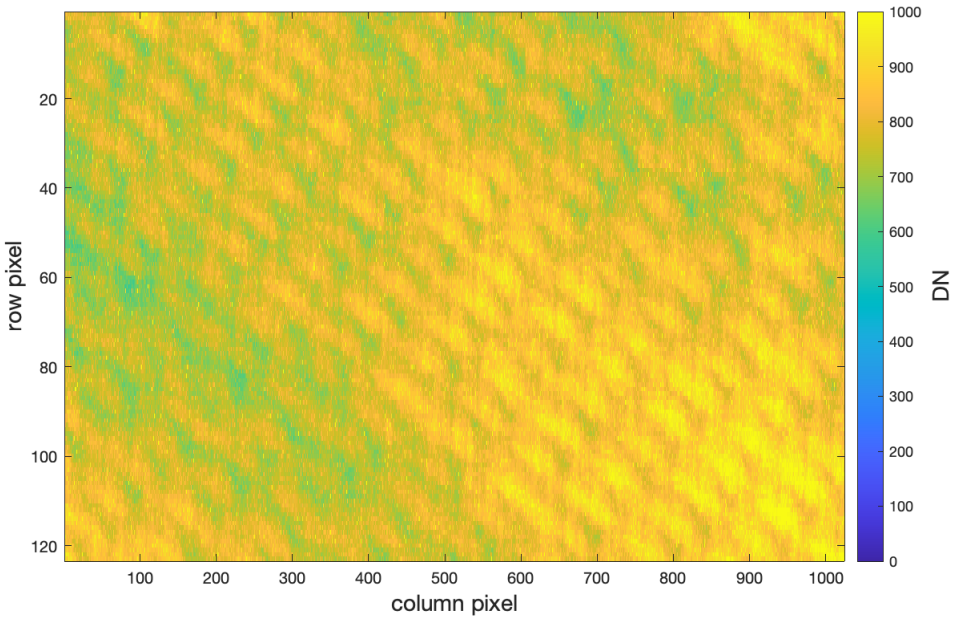


Figure 5.9: Hot Dark Frame for CSL Data

The cold frame, illustrated by Figure 5.8, exhibits diagonal wavy lines across all rows and columns with horizontal lines interrupting the pattern. From this, it can be said that the cold frame is dominated by dark current throughout as a direct result of this fixed pattern noise. The hot frame, portrayed by Figure 5.9, demonstrated fixed pattern noise but on the other end of the temperature spectrum. The frame is over-saturated and pixels (in particular bright pixels) cannot necessarily be distinguished from this due to the cluster formations and the diagonal pattern once again. It is to be noted that the fixed pattern noise is not present on the in-flight data, only on the CSL data. This could suggest that the devices used for this CSL dataset were not a good representation of the flight models sent up in ExoMars TGO.

Figure 5.10 was received by the Open University for a study on radiation damage under cryogenic temperatures (Murray et al., 2014); a separate study. Whilst not mentioned directly in this paper, Figure 5.10 is still derived from the same study. Figure 5.10 shows a device that is placed in a dark environment within a chamber, and frames are taken throughout the night. This proves relevant from a perspective of the assessing the effects of noise. It simply reiterates the effect any sort of anomalous noise can have on a measured value, in this case, electrons.

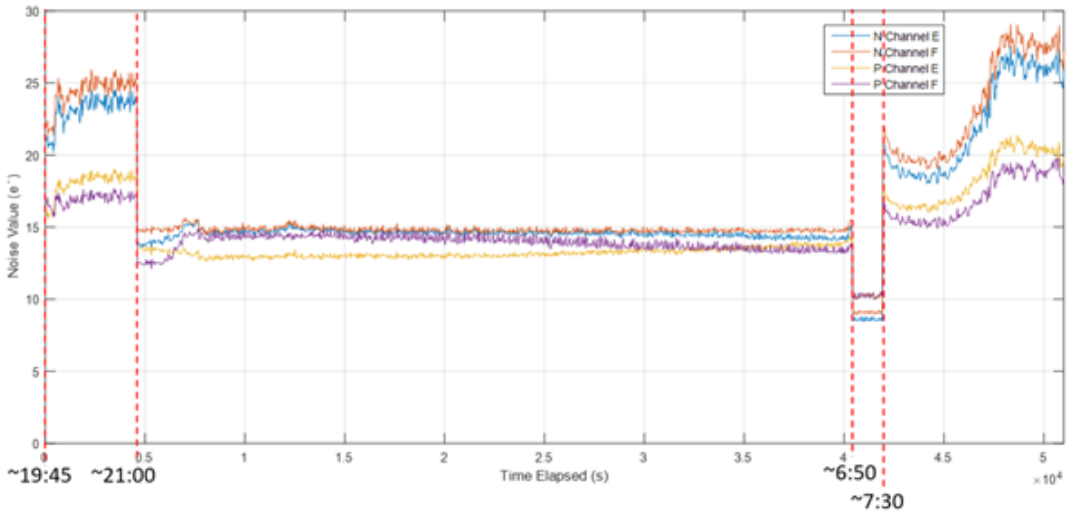


Figure 5.10: Electron variation due to noise fluctuation.

Within Figure 5.10, the noise seems to be increasing from approximately 19:45 to 21:00; however, at 21:00 the noise suddenly drops. This is due to the facility shutting down at night, but the experiment is still being conducted. Overnight, the noise level stays relatively the same up until 6:50, then there is a sharp drop in noise due to the facility being turned back on again. From 7:30 onwards, the noise seems to return to the same levels as before 21:00 the previous night. In summary, the background noise and other factors can have a great contribution to the measured amounts of noise readout from a device.

Chapter 6

Conclusions and Future Recommendations

6.1 Concluding Remarks

Upon review of this study, it is evident that the ground testing data consistently overestimates the dark current levels compared to the in-flight data, thus indicating that ground testing produces greater damage than actually experienced around the Martian orbit. This highlights a key limitation in current ground testing, where the same radiation dose delivered over a short period of time causes artificially elevated dark current due to insufficient time for effects such as annealing.

Clear trends observed include the exponential increase of dark current with temperature and radiation dose, and a much more gradual dark current accumulation over mission duration in-flight. Dark current emerges as a more reliable figure of merit for radiation damage than bright pixels, due to the over-saturation of the ground testing frames (dismissing annealing and temporal effects), making them less comparable between ground and flight datasets.

Current ground testing proves somewhat useful but is still not an accurate representation of in-mission radiation damage. The accelerated, constant-dose exposure during ground tests does not fully replicate the radiation environment of space, causing discrepancies evident in the data. While Flight Models (FM) provide a more accurate response due to their higher-grade silicon, the Engineering Models (EM) remain valuable as for preliminary testing due to their cost effectiveness.

6.2 Future Recommendations

In future, further research could benefit from exploring additional avenues for characterisation of radiation damage beyond dark current, looking at other figures or merit. A more thorough investigation into the process of bright pixel annealing and its correlation with the CCD degradation would be helpful to provide better insight into how radiation damage accumulates over time. A useful expansion from this study would be to include data from other deep space missions to enable a broader and more comprehensive comparison, strengthening the current findings.

In parallel, the development of a predictive model to accurately forecast long-term degradation would be a valuable tool for all future missions, as certain aspects of the mission can be planned and designed around the forecasted data. In a similar vein, a mission run to assess the radiation dose at various areas of space would be useful, as from this, a space radiation map could be developed, allowing for minimal radiation exposure trajectory design. Moreover, to ensure more precise monitoring of the radiation environment, on future missions, it should be compulsory for all spacecraft to be equipped with onboard dosimeters, enabling accurate dose measurements throughout the lifetime of the mission. Finally, incorporating thermal cycling into ground testing protocols would improve the realism and reliability of simulated results, helping to better replicate the space radiation environment.

Bibliography

Ahmed, S. (2022). An investigation of the radiation damage experienced by the ccds on the gaia focal plane.

Ahmed, S., Hall, D. J., Crowley, C. M., Skottfelt, J. M., Dryer, B., Seabroke, G., Hernández, J., & Holland, A. D. (2022). Understanding the evolution of radiation damage on the gaia ccds after 72 months at l2. *https://doi.org/10.1117/1.JATIS.8.1.016003*, *8*, 016003. https://doi.org/10.1117/1.JATIS.8.1.016003

Arons, A., & Peppard, M. (1965). Einstein's proposal of the phton concept - a translation of the annalen der physik paper of 1975. *American Journal of Physics*, *33*. https://astro1.panet.utoledo.edu/~ljc/PE_eng.pdf

Astier, P., Antilogus, P., Juramy, C., Breton, R. L., Guillou, L. L., & Sepulveda, E. (2019). The shape of the photon transfer curve of ccd sensors. *A&A*, *629*, 36. https://doi.org/10.1051/0004-6361/201935508

Benton, E. R., & Benton, E. V. (2001). Space radiation dosimetry in low-earth orbit and beyond. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, *184*, 255–294. https://doi.org/10.1016/S0168-583X(01)00748-0

Carter, J. A., Sembay, S., Nitti, S., Walach, M.-T., Milan, S., Soobiah, Y., Oksavik, K., Forsyth, C., & Taylor, M. (2025). A holistic approach to the smile mission and smile public engagement. *Space Science Reviews*, *221*. https://link.springer.com/content/pdf/10.1007/s11214-025-01175-5.pdf

Chen, S. (2024). The gaussian distribution: Derivation method and its applications in selected examples. https://doi.org/10.54254/2753-8818/38/20240595

Dryer, B. J. (2013). Characterisation of cmos aps technologies for space applications.

ECSS. (2008, July). Ecss-q-st-70-06c – particle and uv radiation testing for space materials (31 july 2008) — european cooperation for space standardization. https://ecss.nl/standard/ecss-q-st-70-06c-particle-and-uv-radiation-testing-for-space-materials/

ECSS. (2018, February). Ecss-q-st-20c rev.2 – quality assurance (1 february 2018) — european cooperation for space standardization. https://ecss.nl/standard/ecss-q-st-20c-rev-2-quality-assurance-1-february-2018/

ECSS. (2020, June). Ecss-e-st-10-04c rev.1 – space environment (15 june 2020) — european cooperation for space standardization. https://ecss.nl/standard/ecss-e-st-10-04c-rev-1-space-environment-15-june-2020/

ESA. (2016, March). Esa - exomars/tgo operations. https://www.esa.int/Enabling_Support/Operations/ExoMars_TGO_operations?utm_source=chatgpt.com

ESA. (2025). Exomars/tgo — ops portal. https://esoc.esa.int/content/exomarstgo

Farley, T. A. (1964). The van allen radiation belt.

Fleck, B., Müller, D., Haugan, S., Duarte, L. S., Siili, T., & Gurman, J. B. (2006). 10 years of soho. *ESA Science*. https://soho.nascom.nasa.gov/publications/ESA_Bull126.pdf

Gow, J. P. D., Mason, J., Leese, M., Hathi, B., & Patel, M. (2017). Proton radiation damage assessment of a ccd for use in a ultraviolet and visible spectrometer.

Gow, J. P., & Murray, N. J. (2017). Charge transfer inefficiency mitigation in a ccd by trap pumping. *Proceedings of the European Conference on Radiation and its Effects on Components and Systems, RADECS*, *2016-September*, 1–4. https://doi.org/10.1109/RADECS.2016.8093128

Hathaway, D. H. (2015). The solar cycle. *Living Rev. Solar Phys*, *12*, 4. https://doi.org/10.1007/lrsp-2015-4

Hu, S. (2017). Solar particle events and radiation exposure in space. https://sohowww.nascom.nasa.gov/publications/ESA_Bull126.pdf

ISO. (2010). Iso15856 - space systems-space environment-simulation guidelines for radiation exposure of non-metallic materials.

Kerbiriou, X., Barthe, M. F., Esnouf, S., Desgardin, P., Blondiaux, G., & Petite, G. (2007). Silicon displacement threshold energy determined by electron paramagnetic resonance and positron annihilation spectroscopy in cubic and hexagonal polytypes of silicon carbide. *Journal of Nuclear Materials*, *362*, 202–207. https://doi.org/10.1016/J.JNUCMAT.2007.01.023

Mason, J. (2025). Exomars tgo nomad-uvis matlab code.

Murray, N. J., Holland, A. D., Gow, J. P. D., Hall, D. J., Stefanov, K. D., Dryer, B. J., Barber, S., & Burt, D. J. (2014). Assessment of the performance and radiation damage effects under cryogenic temperatures of a p-channel ccd204s. *https://doi.org/10.1117/12.206 9154*, 212–225. https://doi.org/10.1117/12.2069289

NASA. (2003, March). Comparison of martian radiation environment with international space station — nasa jet propulsion laboratory (jpl). https://www.jpl.nasa.

gov/images/pia04258-comparison-of-martian-radiation-environment-with-international-space-station/

NASA. (2024, September). What is the solar wind? - nasa science. https://science.nasa.gov/sun/what-is-the-solar-wind/

Parsons, S., Hall, D., Holland, A., Verhoeve, P., Sembay, S., Randall, G., Hetherington, O., Buggey, T., & Soman, M. (2020). Radiation damage testing status of the ccds for the smile sxi.

Polidan, E. J., Waczynski, A., Marshall, P., Johnson, S. D., Marshall, C., Reed, R., Kimble, R. A., Delo, G., Schlossberg, D., Russell, A. M., Beck, T., Wen, Y., Yagelowich, J., & Hill, R. J. (2004). A study of hot pixel annealing in the hubble space telescope wide field camera 3 ccds. *https://doi.org/10.1117/12.552111*, *5487*, 289–298. https://doi.org/10.1117/12.552111

Prod'homme, T., Holl, B., Lindegren, L., & Brown, A. G. (2012). The impact of ccd radiation damage on gaia astrometry - i. image location estimation in the presence of radiation damage. *Monthly Notices of the Royal Astronomical Society*, *419*, 2995–3017. https://doi.org/10.1111/J.1365-2966.2011.19934.X

Reeves, G. D., Delzanno, G. L., & Marksteiner, Q. R. (2020). Earth's radiation belts: The hazards to satellites and what can be done to mitigate the risks? www.amostech.com

Seabroke, G., Holland, A., & Cropper, M. (2008). Modelling radiation damage to esa's gaia satellite ccds. *https://doi.org/10.1117/12.790968*, *7021*, 515–526. https://doi.org/10.1117/12.790968

Sexton, F. W. (2003, June). Destructive single-event effects in semiconductor devices and ics. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1208579

Sihver, L., & Mortazavi, S. M. (2019). Radiation risks and countermeasures for humans on deep space missions. *IEEE Aerospace Conference Proceedings*, *2019-March*. https://doi.org/10.1109/AERO.2019.8742175

Skottfelt, J., Hall, D., Gow, J., Murray, N., Holland, A., & Prod'homme, T. (2016). Comparing simulations and test data of a radiation damaged ccd for the euclid mission. *High Energy, Optical, and Infrared Detectors for Astronomy VII*, *9915*, 991529. https://doi.org/10.1117/12.2232696

Skottfelt, J., Wander, M., Cropper, M. S., Dryer, B., Hall, D. J., Hayes, R., Kelman, B., Kitching, T., Kohley, R., Lagattuta, D. J., Lee-Payne, Z., Liebing, P., Massey, R. J., McCracken, H. J., Nakajima, R., & Nightingale, J. W. (2024). Tracking radiation damage of euclid vis detectors after one year in space. *https://doi.org/10.1117/12.3017996*, *13092*, 163–172. https://doi.org/10.1117/12.3017996

Szafraniec, M., Azzollini, R., Cropper, M., Pottinger, S., Khalil, A., Hailey, M., Hu, D., Plana, C., Cutts, A., Hunt, T., Kohley, R., Walton, D., Theobald, C., Sharples, R., Schmoll, J., & Ferrando, P. (2016). On-ground characterization of the euclid's ccd273-based readout chain. *https://doi.org/10.1117/12.2232922*, *9915*, 372–383. https://doi.org/10.1117/12.2232922

Teledyne. (2020). How a charge coupled device (ccd) image sensor works 50 years of the ccd. https://www.teledyneimaging.com/media/1300/2020-01-22_e2v_how-a-charge-coupled-device-works_web.pdf

Teledyne. (2024, July). Ccd30-11 bsi ccd sensor. https://www.teledynespaceimaging.com/en-us/Products_/Documents/ccd-datasheets/CCD30-11%20BSI%20Datasheet%20(v1).pdf

Vandaele, A. C., Neefs, E., Drummond, R., Thomas, I. R., Daerden, F., Lopez-Moreno, J. J., Rodriguez, J., Patel, M. R., Bellucci, G., Allen, M., Altieri, F., Bolsée, D., Clancy, T., Delanoye, S., Depiesse, C., Cloutis, E., Fedorova, A., Formisano, V., Funke, B., ... Thibert, T. (2015). Science objectives and performances of nomad, a spectrometer suite for the exomars tgo mission. *Planetary and Space Science*, *119*, 233–249. https://doi.org/10.1016/J.PSS.2015.10.003

Widenhorn, R., Blouke, M. M., Weber, A., Rest, A., & Bodegom, E. (2002). Temperature dependence of dark current in a ccd. *https://doi.org/10.1117/12.463446*, *4669*, 193–201. https://doi.org/10.1117/12.463446

Willame, Y., Depiesse, C., Mason, J. P., Thomas, I. R., Patel, M. R., Hathi, B., Leese, M. R., Bolsée, D., Wolff, M. J., Trompet, L., Vandaele, A. C., Piccialli, A., Aoki, S., Ristic, B., Neefs, E., Beeckman, B., Berkenbosch, S., Clairquin, R., Mahieux, A., ... Bellucci, G. (2022). Calibration of the nomad-uvis data. *Planetary and Space Science*, *218*, 105504. https://doi.org/10.1016/j.pss.2022.105504

Xapsos, M. A., Barth, J. L., Stassinopoulos, E. G., Burke, E. A., & Gee, G. B. (1999). Space environment effects: Model for emission of solar protons (esp)-cumulative and worst-case event fluences.

Appendix A

Ethical Approval Letter

See overleaf.



3 April 2025

Dear Mr Ramdeo ,

Reference: CURES/24727/2025
Project ID: 28150
Title: Analysis of space-flight radiation damage to ExoMars Trace Gas Orbiter NOMAD-UVIS spectrometer

Thank you for your application to the Cranfield University Research Ethics System (CURES).

We are pleased to inform you your CURES application, reference CURES/24727/2025 **has been reviewed. You may now proceed with the research activities you have sought approval for.**

If you have any queries, please contact CURES Support.

We wish you every success with your project.

Regards,

CURES Team

Figure A.1: CURES Ethical Approval Letter

Appendix B

Code for Frame Extraction

```
close all
clear all

%% Load the UVIS data from the HDF5 file
fnm = 'dark_frames_raw_combined_80_compressed.h5';
%FrameNumber = h5read(fnm,'/FrameNumber');            % Frame number
FrameTime = h5read(fnm,'/ObservationalData/FrameTime');       % Frame time in seconds
FrameTemp = h5read(fnm,'/ObservationalData/FrameFitTEMP');          % Frame temperature in degrees C
Integration = h5read(fnm,'/ObservationalData/Integration');       % Exposure time

% for a single temperature
% user_temp = -7.0;
% delta_temperature = 0.5;
% frames =
% getframes(fnm,user_temp,delta_temperature,FrameTemp,FrameTime,Integration);extra

% create loops for multiple temperatures
user_temperatures = (-15:5:25);
delta_temperature = 0.5;
for i = 1:numel(user_temperatures)
    RawFrames = getframes(fnm,user_temperatures(i),delta_temperature,FrameTemp,FrameTime,Integration);
end

function RawFrames = getframes(fnm,user_temp,deltaT,temps,times,intT)
```

```
    info = h5info(fnm, '/ObservationalData/RawFrames');
    full_size = info.Dataspace.Size;
    X = full_size(1);
    Y = full_size(2);
    Z = full_size(3);
    ix = abs(temps - user_temp);
    ii = find(ix < deltaT);
    FrameTemp = temps(ii);
    FrameTime = times(ii);
    Integration = intT(ii);

    total_number_of_frames = numel(ii);

    RawFrames = zeros(X,Y,total_number_of_frames);
    ProcessedFrames = zeros(X,Y,total_number_of_frames);
    iframe = 1;
    nframes = 3000;
    num_slices = ceil(Z / nframes);
    p = 1;
    for i = 1:num_slices
        if (iframe + nframes) > Z
            nframes = Z - iframe;
        end
        nframe = iframe - 1 + nframes;
        iTemp = temps(iframe:nframe);
        ix = abs(iTemp - user_temp);
        selected_frames = find(ix < deltaT);
        if isempty(selected_frames) == 0
            q = (p-1) + numel(selected_frames);
            dat = h5read(fnm,'/ObservationalData/RawFrames', [1 1
iframe], [X Y nframes]);
            RawFrames(:,:,p:q) = dat(:,:,selected_frames);
            clear dat
            dat = h5read(fnm,'/ObservationalData/ProcessedFrames',
[1 1 iframe], [X Y nframes]);
            ProcessedFrames(:,:,p:q) = dat(:,:,selected_frames);
            p = q + 1;
        end
        iframe = iframe - 1 + nframes;
    end
    % save the data into a matlab file for easier access
    if user_temp < 0
        sfnm  = ['Frames_at_m',num2str(abs(user_temp)),'C.mat'];
    else
        sfnm  = ['Frames_at_p',num2str(abs(user_temp)),'C.mat'];
```

```
68      end
69      save(sfnm,"FrameTime","FrameTemp","Integration","RawFrames","
        ProcessedFrames","-v7.3")
70  end
```

Listing B.1: Code for Frame Extraction and Storage

Appendix C

Code for Bright Pixel Characterisation

```
close all
clear all

%% ---Load the UVIS data from the HDF5 file---

fnm = 'UVIS_data_subset_forJames.h5';
FrameTime = h5read(fnm,'/FrameTime');                % Frame time
    in seconds
FrameNumber = (1:numel(FrameTime));                  % Frame number
Integration = h5read(fnm,'/Integration');            % Exposure
    time
w_TEMP = h5read(fnm,'/FrameTemp');                % Frame
    temperature in degrees C
w_RawFrames = h5read(fnm,'/RawFrames');             % Raw' frames
    across all orbits (contains bright pixels)
w_ProcessedFrames = h5read(fnm,'/ProcessedFrames');   % Processed
    frames across all orbits (bright pixels removed)


%Correcting the Python to Matlab orientation - Going between Python
     and Matlab transposes the row and column dimensions
RawFrames_Unsorted = reshape_frame_data_single(w_RawFrames);
ProcessedFrames_Unsorted = reshape_frame_data_single(
    w_ProcessedFrames);

% Sort by time
[Time,isort] = sort(FrameTime,'ascend');
Time = FrameTime(isort);
FrameTemp = w_TEMP(isort);
RawFrames = RawFrames_Unsorted (:,:,isort);
ProcessedFrames = ProcessedFrames_Unsorted (:,:,isort);
```

```
[~, ~, nt] = size(RawFrames);
disp(['Total number of frames: ', num2str(nt)])


%% Reformatting for appearance on plot

function [rdat1] = reshape_frame_data_single(dat1)
    [nx,ny,nz] = size(dat1);
    rdat1 = zeros(ny,nx,nz);
    for i = 1:nz
        rdat1(:,:,i) = dat1(:,:,i)'; % transpose the array
    end
end

%% --- Compute Mean Dark Current per Frame Plot---

MeanDarkCurrent = zeros(1, nt);  % Mean DN per frame
TempPerFrame = FrameTemp;        % Already in degrees C
ExpTime = Integration;           % Exposure time per frame, in
   seconds

% Mean DN per frame normalised by exposure time
for i = 1:nt
    frame = ProcessedFrames(:,:,i);      % Dark frame with no
   bright pixels
    MeanDN(i) = mean(frame(:), 'omitnan');  % Mean DN over all
   pixels
    MeanDarkCurrent(i) = MeanDN(i);
    MeanDarkCurrent(i) = MeanDN(i) * ExpTime(i);  % DN/s
end

%% --- Plot: Mean Dark Current vs Temperature ---

figure('Name','Mean Dark Current (DN) Vs. Tempertature ','
   NumberTitle','off');
plot(TempPerFrame, MeanDarkCurrent, 'o');
xlabel('Temperature [ C ]');
ylabel('Mean Dark Current [DN/s] (log scale)');
title('Mean Dark Current vs. Temperature (Log Scale)');
grid on;

%% --- Histogram : Dark Current Vs. Time (Singular Frame) ---

Dark_Current = RawFrames(:,:,end); % Looking at Final Frame
figure('Name','Dark Current Over Time ','NumberTitle','off');
```

```
66 h = histogram(Dark_Current,2000); % Resolution specified by number
       of bins at the end of this line
67 h.Parent.XScale = "log";
68
69 xlabel('Time [s]');
70 ylabel('Dark Current [DN]');
71 ylim([0,40000]);
72 title('Dark Current Over Time (Singular Frame)');
73
74 %% --- Histogram : Pixel Noise Vs. Time (Singular Frame) ---
75
76 % Looking at Final Frame
77 Pixel_Noise = RawFrames(:,:,end)-MeanDN(end); % Essentially
       Background Noise as the meaDN has been sutracted (BRIGHT PIXELS)
78 figure('Name','Pixel Noise Over Time ','NumberTitle','off');
79 h = histogram(Pixel_Noise,3000); % Resolution specified by nunmber
       of bins at the end of this line
80 h.Parent.XScale = "log";
81
82 xlabel('Time [s]');
83 ylabel('Pixel Noise [DN]');
84 ylim([0,25000]);
85 title('Pixel Noise Over Time (Singular Frame)');
86
87 %% --- Histogram : Pixel Count Vs. Pixel Noise ---
88
89 BP_Vector = Pixel_Noise(:); % Bright Pixel Vector
90 medi = median(BP_Vector, 'omitnan'); % Finding the median value of
       all the values in the Bright Pixel Vector values
91 sigma = std(BP_Vector, 'omitnan'); % Stating the sigma value ot be
       the standard deviation of the Bright Pixel Vector values
92 threshold_DN = medi + (3 * sigma); % Creates the theshold to be 3
       sigma above the median...
93
94 figure('Name','Pixel Count Over Pixel Noise','NumberTitle','off');
95 h = histogram(BP_Vector(BP_Vector > 0), 3000);
96 h.Parent.XScale = "log";
97
98 xlabel('Pixel Noise [DN]');
99 ylabel('Pixel Count');
100 title('Pixel Noise Distribution in a Single Frame');
101
102 % Add 3   threshold line
103 hold on;
104 yLimits = ylim;
```

```
105 plot([threshold_DN threshold_DN], yLimits, 'r--', 'LineWidth', 2);
106 hold off;
107
108 legend('Pixel Noise Histogram', '3   Threshold');
109 disp(['3-Sigma DN Threshold: ', num2str(threshold_DN)]);
110
111 %% Creating a Binary Mask using the 3-Sigma Threshold.
112
113 % Create binary mask of high-noise pixels (above or equal to
    threshold)
114 HighNoiseMask = Pixel_Noise >= threshold_DN;
115
116
117 figure('Name','High Noise Pixel Mask','NumberTitle','off');
118 imagesc(HighNoiseMask);
119 colormap(parula);
120 colorbar;
121 xlabel('columns')
122 ylabel('rows')
123 title('Mask of High-Noise Pixels (Above 3-sigma Threshold)');
124
125 FilteredData = RawFrames(:,:,end); % Looking at the filtered data
    applied to the end frame
126 FilteredData(~HighNoiseMask) = NaN;  % Keep only high-noise pixels,
     set rest to NaN
127
128 figure('Name','DN Values of High-Noise Pixels','NumberTitle','off')
    ;
129 imagesc(FilteredData);
130 colorbar;
131 title('DN Values of High-Noise Pixels in Final Frame');
132
133 % stdFiltered = std(FilteredData(:), 'omitnan');
134 % meanFiltered = mean(FilteredData(:), 'omitnan');
135
136 %% Standard Deviation of those pixels within the final frame
137
138 High_Noise_Pixels = double(RawFrames(:,:,end)); % Doubles for more
    precision and looks at the end frame
139 High_Noise_Values = High_Noise_Pixels (HighNoiseMask);
140
141 Noise_STD = std(High_Noise_Values, 'omitnan');
142
143 disp(['Standard Deviation of High-Noise Pixels: ', num2str(
    Noise_STD)])
```

```
144
145 %% plot the dark frame with Bright pixels
146
147 fig = figure('Name','Raw Dark Frame with Bright pixels','
    NumberTitle','off');
148 imagesc(RawFrames(:,:,end))
149 %clim([0 40000])
150 xlabel('columns')
151 ylabel('rows')
152 title('Raw Dark Frame with Bright pixels')
153
154 %% plot the dark frame without Bright pixels
155
156 fig = figure('Name','Processed Dark Frame without Bright pixels','
    NumberTitle','off');
157 imagesc(ProcessedFrames(:,:,end))
158 %clim([0 10])
159 xlabel('columns')
160 ylabel('rows')
161 title('Processed Dark Frame without Bright pixels')
```

Listing C.1: Code for applying 3-Sigma Threshold

Appendix D

Plots for Bright Pixel Analysis

See overleaf.

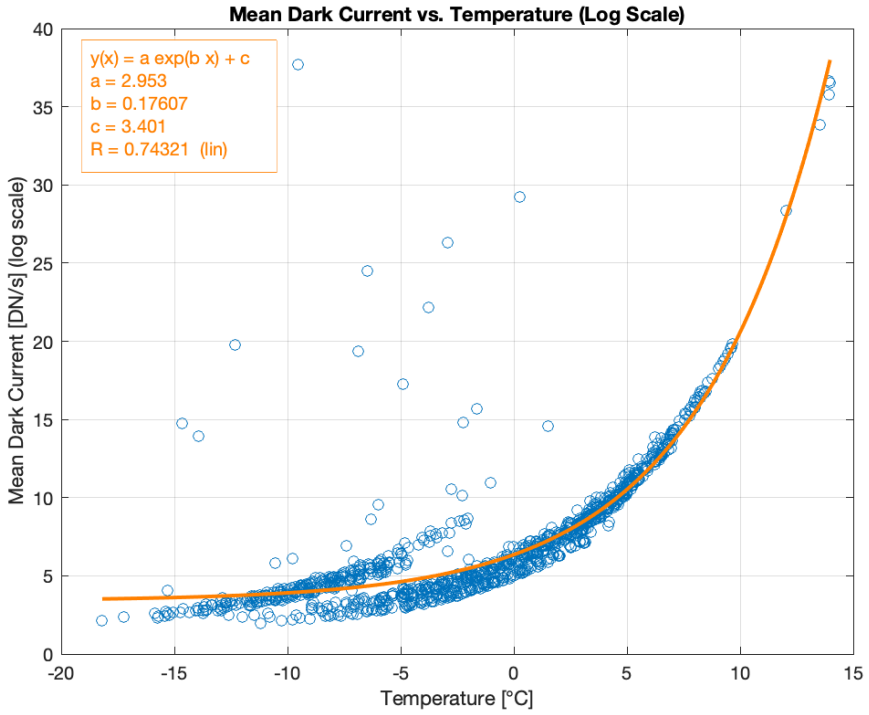


Figure D.1: Mean Dark Current vs Temp plot with exponential fit.

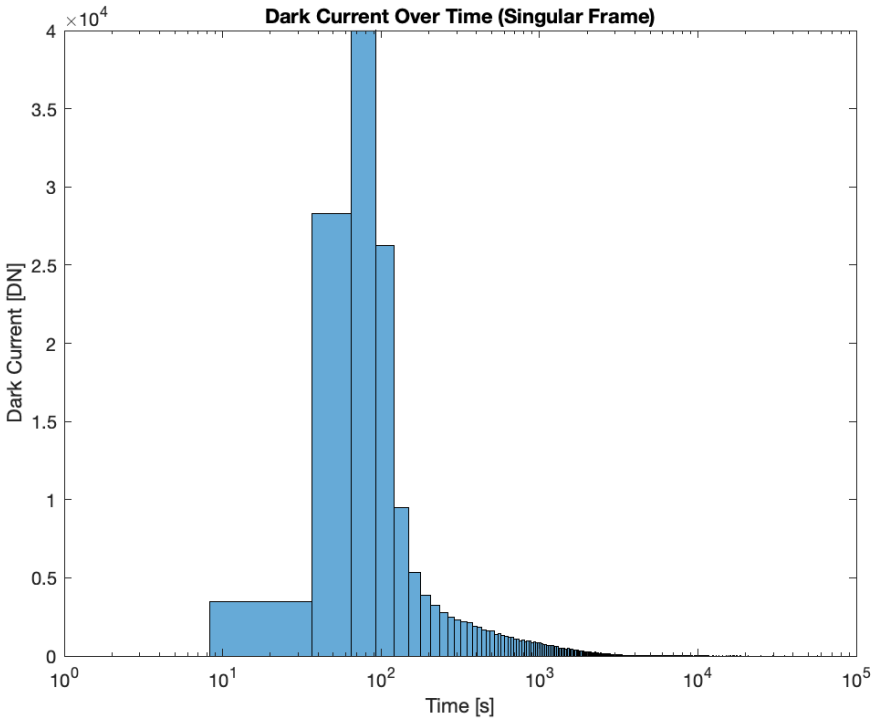


Figure D.2: Mean Dark Current over singular frame time histogram.

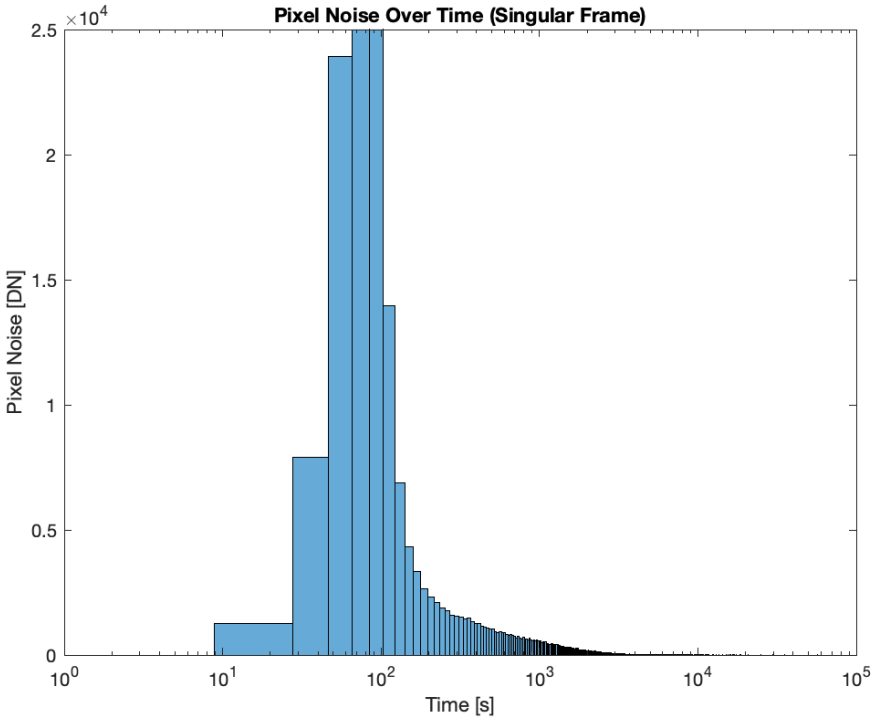


Figure D.3: Pixel Noise over singular frame time histogram.

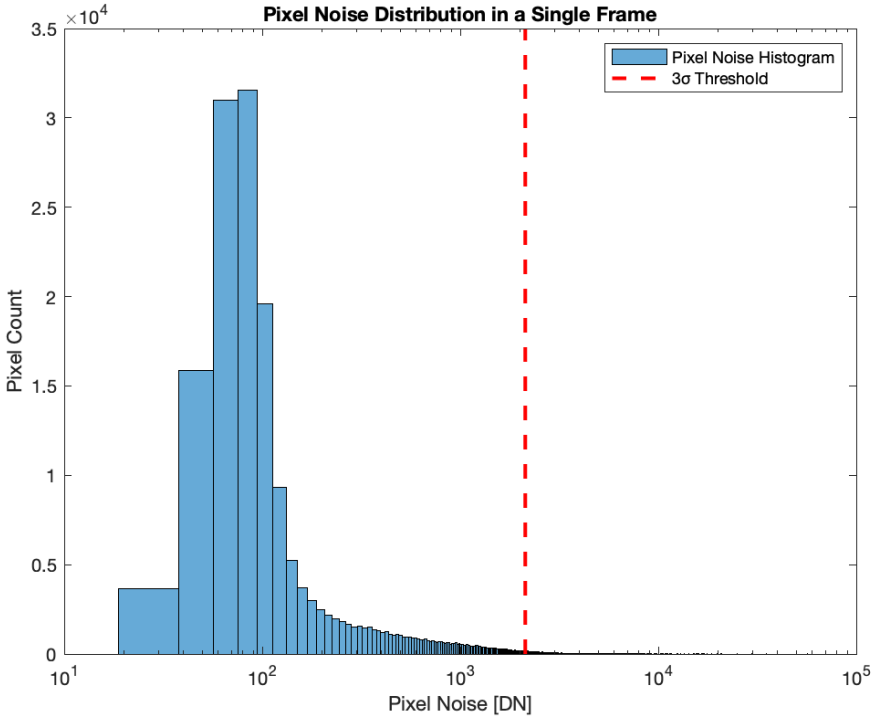


Figure D.4: Pixel Noise Distribution over over a singular frame with the 3-Sigma Threshold applied

Appendix E

Full Code For Ground Test Data

```
close all
clear all

load('uvis_ground_data.mat')

% The data is separated into Engineering Model (EM) detectors and
    Flight
% Model (FM) detectors

% EM data
%%%%%% pre-radiation data %%%%%%%%%%%%%%%%%
pre = gnd.em.pre;
% top level of pre is the 4 detectors
%pre.det1
%pre.det2
%pre.det3
%pre.det4

% The structure for each detector is as follows;
%pre.det1.integration
%pre.det1.temp
%pre.det1.frames


%%%%%%% radiation data %%%%%%%%%%%%%%%%%%%%
rad = gnd.em.irradiation;
% top level of rad
%rad.halfeol
%rad.eol
%rad.eolx2
% The structure in rad for each dose is as follows;
```

```
% rad.halfeol.dose => radiation dose
% rad.halfeol.dark.frames => dark frames
% rad.halfeol.dark.temp => temperature
% rad.halfeol.dark.integration => exposure time


%%%%%%%%% FM data
fm = gnd.fm;
% top level of fm is the 4 detectors
%fm.det1
%fm.det2
%fm.det3
%fm.det4

% The structure for each detector is as follows;
%fm.det1.integration
%fm.det1.temp
%fm.det1.frames

%% --- Figures to demonstrate the dark frames after different
    dosages of radiation ---

% EOL Plot
figure('Name','EOL Dark Frame','NumberTitle','off')
imagesc(rad.eol.dark.frames(:,:,8))
xlabel('columns')
ylabel('rows')
colorbar()

% Half EOL Plot
figure('Name','Half EOL Dark Frame','NumberTitle','off')
imagesc(rad.halfeol.dark.frames(:,:,8))
xlabel('columns')
ylabel('rows')
colorbar()

% Double EOL PLot
figure('Name','Double EOL Dark Frame','NumberTitle','off')
imagesc(rad.eolx2.dark.frames(:,:,8))
xlabel('columns')
ylabel('rows')
colorbar()

%% --- Usable Frame Data Range ---

```

```
radHALFEOL_Frames = rad.halfeol.dark.frames(30:230,600:1000,:);
radEOL_Frames = rad.eol.dark.frames(30:230,50:500,:);
radEOLx2_Frames = rad.eolx2.dark.frames(30:230,600:1000,:);

%% --- Dark Current Vs Temp Plot ---

% Gain Values
gainHALFEOL = 6.69; % e-/DN
gainEOL = 6.85;
gainEOLx2 = 6.76;

%%% Computing Dark Current %%%

function [invT, dark_c] = getDarkCurrent(frames, temps_C,
    integration, gain)
    % Compute mean DN per frame
    meanDN = mean(frames, [1 2]);

    % Using first integration time of array (as repeated)
    if numel(integration) > 1
        integration = integration(1);
    end

    % Dark current in e-/pixel/s
    dark_c_all = (meanDN * gain) ./ integration;

    % Average dark current per temperature
    [uniqueT, ~, ic] = unique(temps_C, 'stable');
    dark_c = zeros(size(uniqueT));
    for i = 1:length(uniqueT)
        dark_c(i) = mean(dark_c_all(ic == i));
    end

    % Inverse temperature (1000 / Temp in K) for Arrhenius plot
    invT = 1000 ./ (uniqueT + 273.15);

    % --- Sort points by increasing invT ---
    [invT, sortIdx] = sort(invT);
    dark_c = dark_c(sortIdx);
end


%%% Dark currents per dose %%%

% Control
```

```
pre_int = pre.det4.integration(1);
[invT_ctrl, dark_ctrl] = getDarkCurrent(pre.det4.frames, pre.det4.
    temp, pre_int, gainHALFEOL);

% Half EOL
halfeol_int = rad.halfeol.dark.integration(1);
[invT_d1, dark_d1] = getDarkCurrent(radHALFEOL_Frames, rad.halfeol.
    dark.temp, halfeol_int, gainHALFEOL);

% EOL
eol_int = rad.eol.dark.integration(1);
[invT_d2, dark_d2] = getDarkCurrent(radEOL_Frames, rad.eol.dark.
    temp, eol_int, gainEOL);

% Double EOL
eolx2_int = rad.eolx2.dark.integration(1);
[invT_d3, dark_d3] = getDarkCurrent(radEOLx2_Frames, rad.eolx2.dark
    .temp, eolx2_int, gainEOLx2);


%%% Arrhenius Plot %%%
figure('Name','Arrhenius Plot of Dark Current vs Temperature','
    NumberTitle','off')

% Define consistent markers and colours
plotMarkers = {'o','s','^','d'};
plotColours  = lines(4); % MATLAB default colour order

semilogy(invT_ctrl, dark_ctrl, ['-' plotMarkers{1}],'LineWidth'
    ,0.5, 'Color', plotColours(1,:), 'DisplayName', 'Control CCD');
    hold on
semilogy(invT_d1, dark_d1, ['-' plotMarkers{2}], 'LineWidth',0.5,'
    Color', plotColours(2,:), 'DisplayName', 'Half EOL     2.4 10 ^9
     p.cm^{-2}')
semilogy(invT_d2, dark_d2, ['-' plotMarkers{3}],'LineWidth',0.5, '
    Color', plotColours(3,:), 'DisplayName', 'EOL     4.8 10 ^9 p.cm
    ^{-2}')
semilogy(invT_d3, dark_d3, ['-' plotMarkers{4}],'LineWidth',0.5, '
    Color', plotColours(4,:), 'DisplayName', 'Double EOL     9.4 10
    ^9 p.cm^{-2}')

xlabel('1000 / Temperature (K)')
ylabel('Dark Current (e^{-} pixel ^{-1} s ^{-1})', 'Interpreter','
    tex')
legend('Location','southwest')
```

```
grid on
title('Arrhenius Plot of Dark Current vs Temperature')

%% --- Dark Current Vs Proton Fluence ---

% Gain Values
gainHALFEOL = 6.69; % e-/DN
gainEOL = 6.85;
gainEOLx2 = 6.76;

targetTemps = [20, 10, 0, -10, -20];
desired_limit_0C = 2000;  % e-/pix/s @ 0 C

% Proton Fluences
HalfEOL_fluence = str2double(rad.halfeol.dose);
EOL_fluence    = str2double(rad.eol.dose);          % p cm^-2
DoubleEOL_fluence = str2double(rad.eolx2.dose);
lifetime_flux = 1.2 * EOL_fluence;    % p cm^-2 --lifetime margin
    1.2x

% Fluence axis (Pre = 0)
fluences = [0, HalfEOL_fluence, EOL_fluence, DoubleEOL_fluence];

%%% Compute dark current (averaged per unique T), then pick nearest
     T %
function [Tuniq, avgdc] = dark_vs_temp(frames, temps_C, integration
    , gain)
    meanDN = mean(frames, [1 2]);          % DN per frame
    if numel(integration) > 1, integration = integration(1); end
    dark_all = (meanDN * gain) ./ integration;     % e-/pix/s for
    each frame
    [Tuniq, ~, grp] = unique(temps_C(:), 'stable');
    avgdc = zeros(size(Tuniq));
    for k = 1:numel(Tuniq)
        avgdc(k) = mean(dark_all(grp == k));
    end
end

function dc_at_T = nearest_dc_at(frames, temps_C, integration, gain
    , Ttarget)
    [Tuniq, dcuniq] = dark_vs_temp(frames, temps_C, integration,
    gain);
    % pick closest temperature
    [~, idx] = min(abs(Tuniq - Ttarget));
    dc_at_T = dcuniq(idx);
```

```
189 end
190
191
192 %%% Collect dark current at the target temperatures %%%
193
194 pre_int = pre.det4.integration(1);
195
196 dc_pre_20  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
    pre_int, gainHALFEOL, 20);
197 dc_pre_10  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
    pre_int, gainHALFEOL, 10);
198 dc_pre_0   = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
    pre_int, gainHALFEOL, 0);
199 dc_pre_m10  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
    pre_int, gainHALFEOL, -10);
200 dc_pre_m20  = nearest_dc_at(pre.det4.frames,   pre.det4.temp,
    pre_int, gainHALFEOL, -20);
201
202
203 % Half-EOL
204 halfeol_int = rad.halfeol.dark.integration(1);
205 dc_hfeol_20  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
    temp, halfeol_int, gainHALFEOL, 20);
206 dc_hfeol_10  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
    temp, halfeol_int, gainHALFEOL, 10);
207 dc_hfeol_0   = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
    temp, halfeol_int, gainHALFEOL, 0);
208 dc_hfeol_m10  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
    temp, halfeol_int, gainHALFEOL, -10);
209 dc_hfeol_m20  = nearest_dc_at(radHALFEOL_Frames, rad.halfeol.dark.
    temp, halfeol_int, gainHALFEOL, -20);
210
211
212 % EOL
213 eol_int = rad.eol.dark.integration(1);
214 dc_eol_20  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
    eol_int, gainEOL, 20);
215 dc_eol_10  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
    eol_int, gainEOL, 10);
216 dc_eol_0   = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
    eol_int, gainEOL, 0);
217 dc_eol_m10  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
    eol_int, gainEOL, -10);
218 dc_eol_m20  = nearest_dc_at(radEOL_Frames, rad.eol.dark.temp,
    eol_int, gainEOL, -20);
```

```
219
220
221 % Double EOL
222 eolx2_int = rad.eolx2.dark.integration(1);
223 dc_x2_20  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
    eolx2_int, gainEOLx2, 20);
224 dc_x2_10  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
    eolx2_int, gainEOLx2, 10);
225 dc_x2_0   = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
    eolx2_int, gainEOLx2, 0);
226 dc_x2_m10  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
    eolx2_int, gainEOLx2, -10);
227 dc_x2_m20  = nearest_dc_at(radEOLx2_Frames, rad.eolx2.dark.temp,
    eolx2_int, gainEOLx2, -20);
228
229 % Series format for plot
230 dc_20  = [dc_pre_20,  dc_hfeol_20,  dc_eol_20,  dc_x2_20];
231 dc_10  = [dc_pre_10,  dc_hfeol_10,  dc_eol_10,  dc_x2_10];
232 dc_0   = [dc_pre_0,   dc_hfeol_0,   dc_eol_0,   dc_x2_0];
233 dc_m10  = [dc_pre_m10,  dc_hfeol_m10,  dc_eol_m10,  dc_x2_m10];
234 dc_m20  = [dc_pre_m20,  dc_hfeol_m20,  dc_eol_m20,  dc_x2_m20];
235
236
237
238 %%%  Plot %%%
239 figure('Name','Dark Current Vs Proton Fluence','NumberTitle','off')
240
241 % Make a fine grid of fluences for smooth curve
242 fine_fluences = linspace(min(fluences), max(fluences), 200);
243 % Apply spline interpolation
244 dc20_spline = spline(fluences, dc_20, fine_fluences);
245 dc10_spline = spline(fluences, dc_10, fine_fluences);
246 dc0_spline = spline(fluences, dc_0, fine_fluences);
247 dcm10_spline = spline(fluences, dc_m10, fine_fluences);
248 dcm20_spline = spline(fluences, dc_m20, fine_fluences);
249
250
251 % Plot smooth spline lines
252 semilogy(fine_fluences, dc20_spline,  '-', 'LineWidth',0.5, '
    DisplayName','20  C '); hold on
253 semilogy(fine_fluences, dc10_spline,  '-', 'LineWidth',0.5, '
    DisplayName','10  C ');
254 semilogy(fine_fluences, dc0_spline,   '-', 'LineWidth',0.5, '
    DisplayName','0  C ');
```

```
255 semilogy(fine_fluences, dcm10_spline, '-', 'LineWidth',0.5, '
    DisplayName','-10 C');
256 semilogy(fine_fluences, dcm20_spline, '-', 'LineWidth',0.5, '
    DisplayName','-20 C');
257
258
259 % Plot original fluences as markers
260 semilogy(fluences, dc_20, 'o', 'MarkerSize',6, 'MarkerFaceColor','
    auto', 'DisplayName','20 C');
261 semilogy(fluences, dc_10, 'o', 'MarkerSize',6, 'MarkerFaceColor','
    auto', 'DisplayName','10 C');
262 semilogy(fluences, dc_0, 's', 'MarkerSize',6, 'MarkerFaceColor','
    auto', 'DisplayName','0 C');
263 semilogy(fluences, dc_m10, 'o', 'MarkerSize',6, 'MarkerFaceColor',
    'auto', 'DisplayName','-10 C');
264 semilogy(fluences, dc_m20, 'o', 'MarkerSize',6, 'MarkerFaceColor',
    'auto', 'DisplayName','-20 C');
265
266
267
268 % Vertical lines: EOL and 1.2x EOL (lifetime requirement point)
269 xline(EOL_fluence, '--', 'EOL','LabelOrientation','horizontal');
270 xline(lifetime_flux, '--', '1.2 EOL','LabelOrientation','
    horizontal');
271
272 % Horizontal line: desired limit at 0 C
273 yline(desired_limit_0C, ':', 'Desired 0 C limit', '
    LabelHorizontalAlignment','left', 'Color','r');
274
275 grid on
276 xlabel('Proton fluence (p cm^{-2})');
277 ylabel('Dark current (e^{-}\cdot pixel^{-1}\cdot s^{-1})', '
    Interpreter','tex');
278 title('Dark current vs proton fluence at 20 C , 10 C , 0 C , -10
     C , -20 C');
279 legend('Location','southeast');
280
281 %%% Labelling EOL Points %%%
282 dc20_EOL = dc_20(fluences == EOL_fluence);
283 dc10_EOL = dc_10(fluences == EOL_fluence);
284 dc0_EOL = dc_0(fluences == EOL_fluence);
285 dcm10_EOL = dc_m10(fluences == EOL_fluence);
286 dcm20_EOL = dc_m20(fluences == EOL_fluence);
287
288
```

```
Annotations = {
    sprintf('20  C : %.2e e^- pixel ^{-1} s ^{-1}', dc20_EOL),
    sprintf('10  C : %.2e e^- pixel ^{-1} s ^{-1}', dc10_EOL),
    sprintf('0  C : %.2e e^- pixel ^{-1} s ^{-1}', dc0_EOL),
    sprintf('-10  C : %.2e e^- pixel ^{-1} s ^{-1}', dcm10_EOL),
    sprintf('-20  C : %.2e e^- pixel ^{-1} s ^{-1}', dcm20_EOL) };


% Text Positions
ys = [dc20_EOL, dc10_EOL, dc0_EOL, dcm10_EOL, dcm20_EOL];
xs = repmat(EOL_fluence,1,numel(ys));
text(xs*1.05, ys, Annotations, 'VerticalAlignment','bottom','
   FontSize',10);
```

Listing E.1: Full Code for Ground Data.

Appendix F

Full Code section for In-Flight Data

```
close all
clear all

%% Load the UVIS mean DC from the flight data
fnm = 'UVIS_dark_frames_mean.h5';
FrameTime = h5read(fnm,'/ObservationalData/FrameTime');
       % Frame time in seconds
FrameTemp = h5read(fnm,'/ObservationalData/FrameTEMP');
       % Frame temperature in degrees C
MeanDC = h5read(fnm,'/ObservationalData/MeanDarkCurrent');
      % Exposure time
%% Dark Current Vs Temperature Plot with increase in time

gain = 7.28;
ntime = (FrameTime - FrameTime(1)) / (365*24*3600);
figure()
scatter(FrameTemp,MeanDC*gain,6,ntime,"filled") % use ntime to get
   the color to show time
xlabel('Temperature (^{\circ}C )','FontSize',12)
ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})','FontSize',12)
title ('Dark Current Vs Temperature Plot with increase in time')
cb = colorbar;
caxis([0,9])
ylabel(cb,'time / years','FontSize',16)

%% Filter flight data

% Create mask for non-zero dark current values
non_zero_dc_mask = (MeanDC > 0);

% Apply the filter to all flight data arrays
```

```
MeanDC_nonzero = MeanDC(non_zero_dc_mask);
FrameTemp_nonzero = FrameTemp(non_zero_dc_mask);
ntime_nonzero = ntime(non_zero_dc_mask);
FrameTime_nonzero = FrameTime(non_zero_dc_mask);

fprintf('Original data points: %d\n', length(MeanDC));
fprintf('Data points with DC > 0: %d\n', sum(non_zero_dc_mask));
fprintf('Zero dark current points removed: %d\n', sum(MeanDC == 0))
    ;
fprintf('Percentage with DC > 0: %.1f%%\n', 100 * sum(
    non_zero_dc_mask)/length(MeanDC));

%% Time increment analysis
years = (0:0.5:9); % half a year windows
point = zeros(length(years)-1, 2);

for i = 1:numel(years)-1
    i1 = find(ntime_nonzero > years(i));
    i2 = find(ntime_nonzero(i1) < years(i+1));

    if ~isempty(i2)
        i3 = i1(i2);
        bdark{i} = MeanDC_nonzero(i3);
        btemp{i} = FrameTemp_nonzero(i3);

        figure()
        plot(btemp{i}, bdark{i}*gain, 'x')
        xlabel('Temperature ( C )')
        ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
        title(sprintf('Time Window: %.1f-%.1f years (DC > 0)', years
    (i), years(i+1)))

        if length(btemp{i}) > 3
            f = ezfit(btemp{i}, bdark{i}*gain, 'y = a * exp (b * x)
    + c', [10, 0.1, 10]);
            showfit(f)
            point(i,1:2) = [years(i), evalfit(f,0)];
        else
            point(i,1:2) = [years(i), NaN];
        end
    else
        point(i,1:2) = [years(i), NaN];
    end
end

```

```
% Remove NaN values
valid_idx = ~isnan(point(:,2));
point_clean = point(valid_idx,:);

%% Dark current at 0 C  Vs Mission Duration (using only DC > 0 data
    )
figure()
plot(point_clean(:,1)+0.25, point_clean(:,2), 'x')
xlabel('Time (Years)')
ylabel('Dark Current at 0 C  (e{^-} pixel^{-1} s^{-1})')
title('Dark current at 0 C  Vs Mission Duration')
grid on
```

Listing F.1: Code for In-Flight Mission Dark Current Vs Temperature Plot

Appendix G

Incremental Time Analysis Fitted Figures

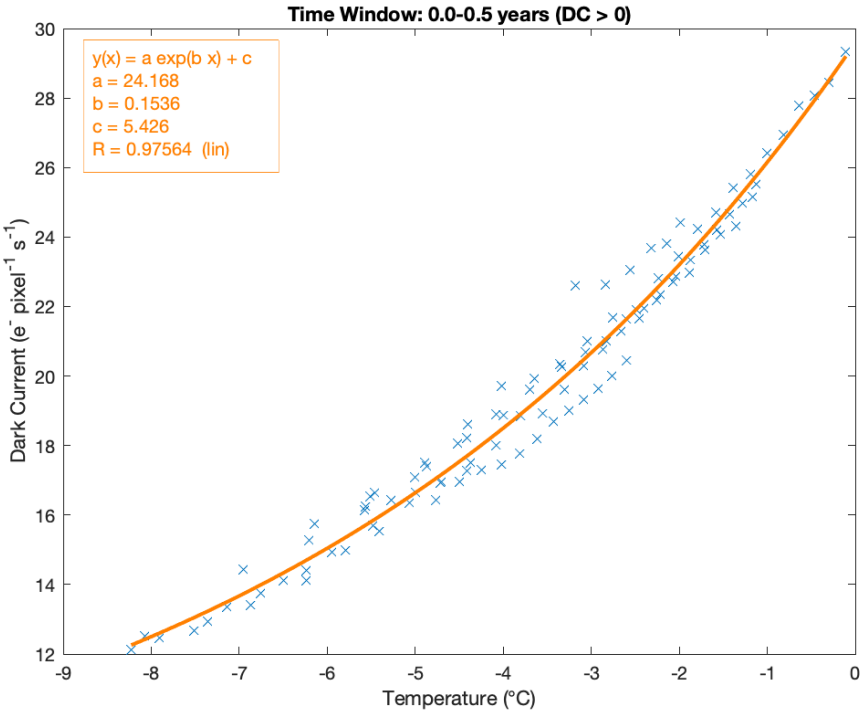


Figure G.1: Dark current vs temperature plot from 0-0.5 years (fitted).

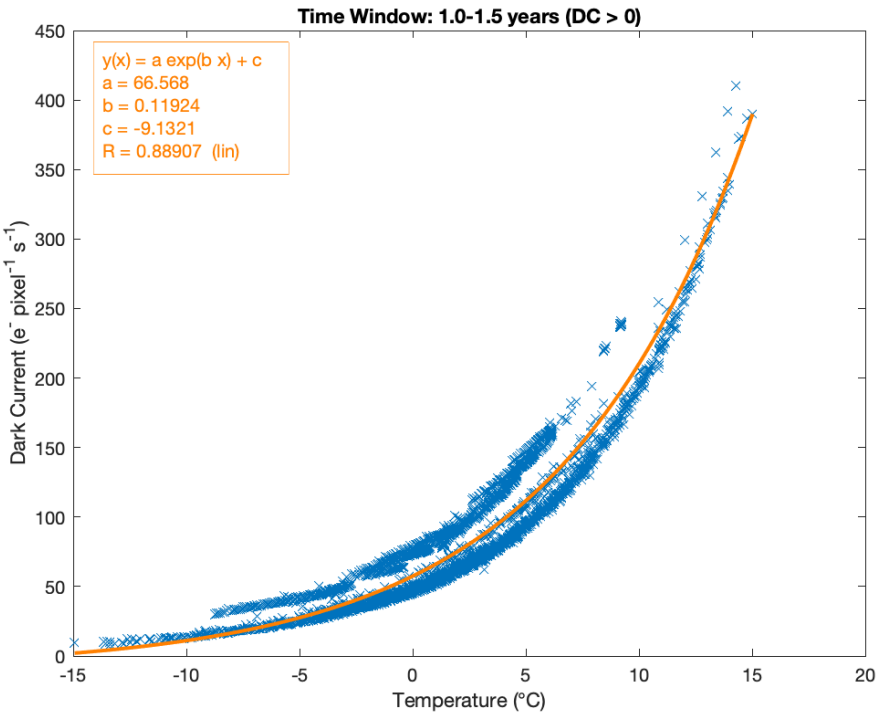


Figure G.2: Dark current vs temperature plot from 1-1.5 years (fitted).

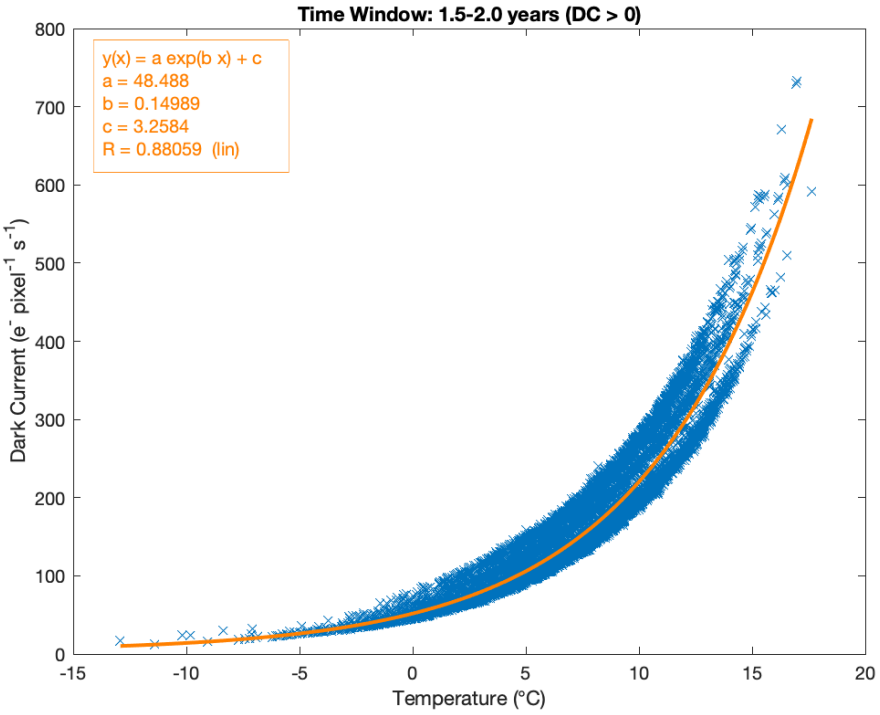


Figure G.3: Dark current vs temperature plot from 1.5-3 years (fitted).

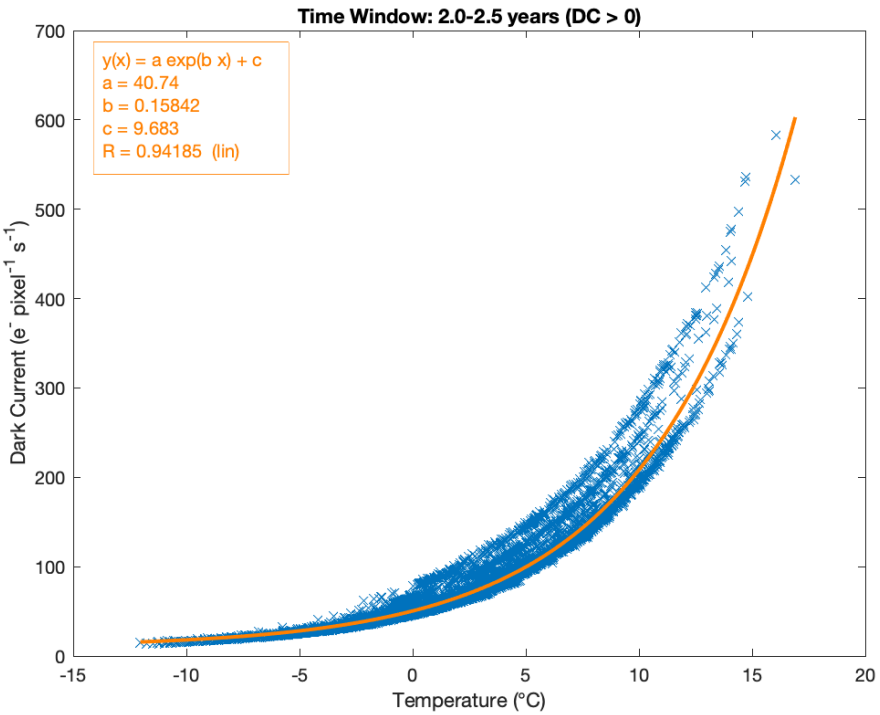


Figure G.4: Dark current vs temperature plot from 2-2.5 years (fitted).

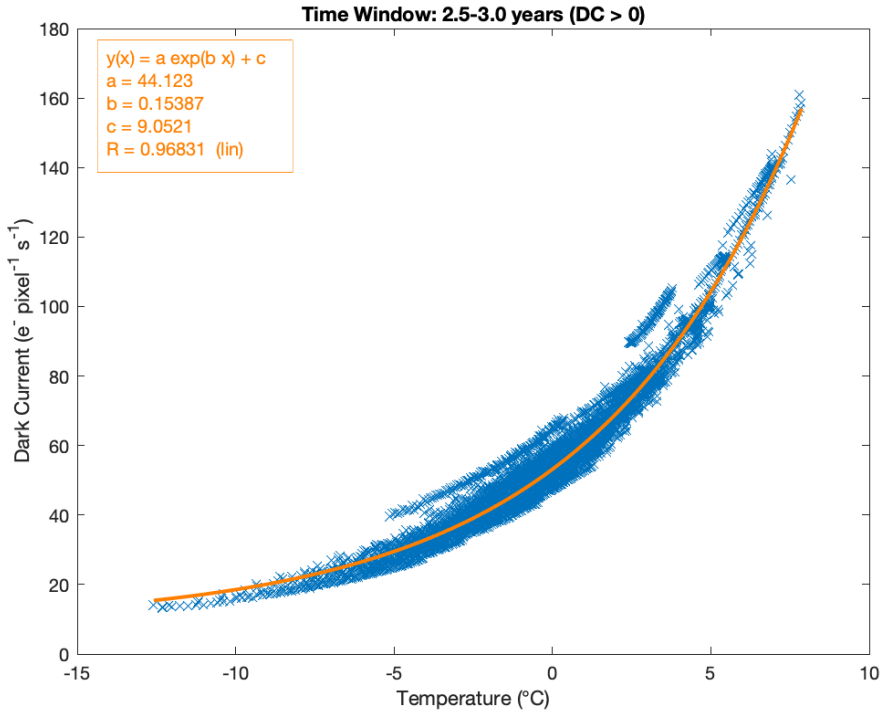


Figure G.5: Dark current vs temperature plot from 2.5-3 years (fitted).

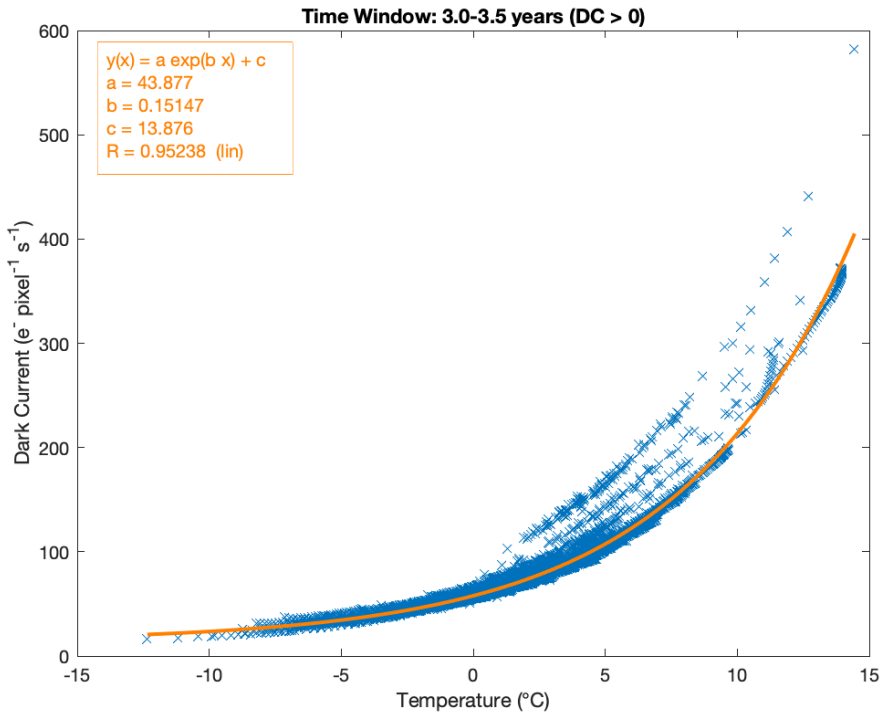


Figure G.6: Dark current vs temperature plot from 3-3.5 years (fitted).

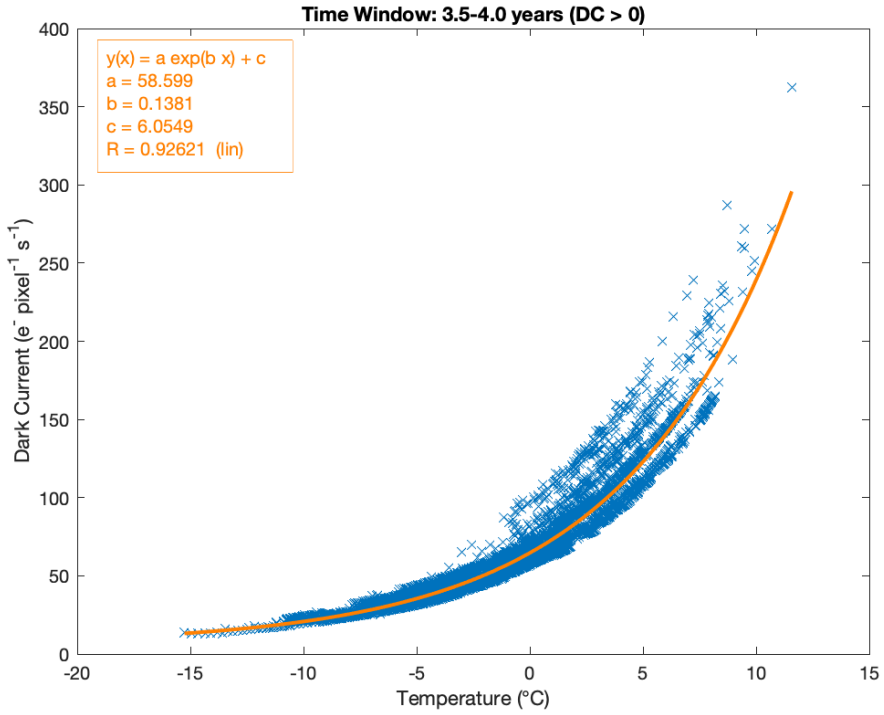


Figure G.7: Dark current vs temperature plot from 3.5-4 years (fitted).

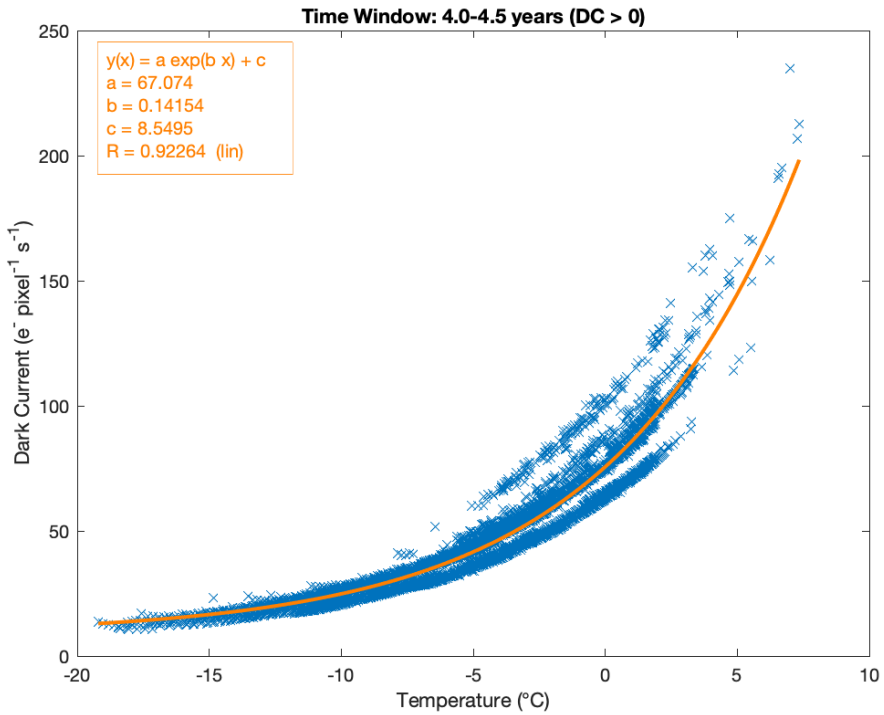


Figure G.8: Dark current vs temperature plot from 4-4.5 years (fitted).

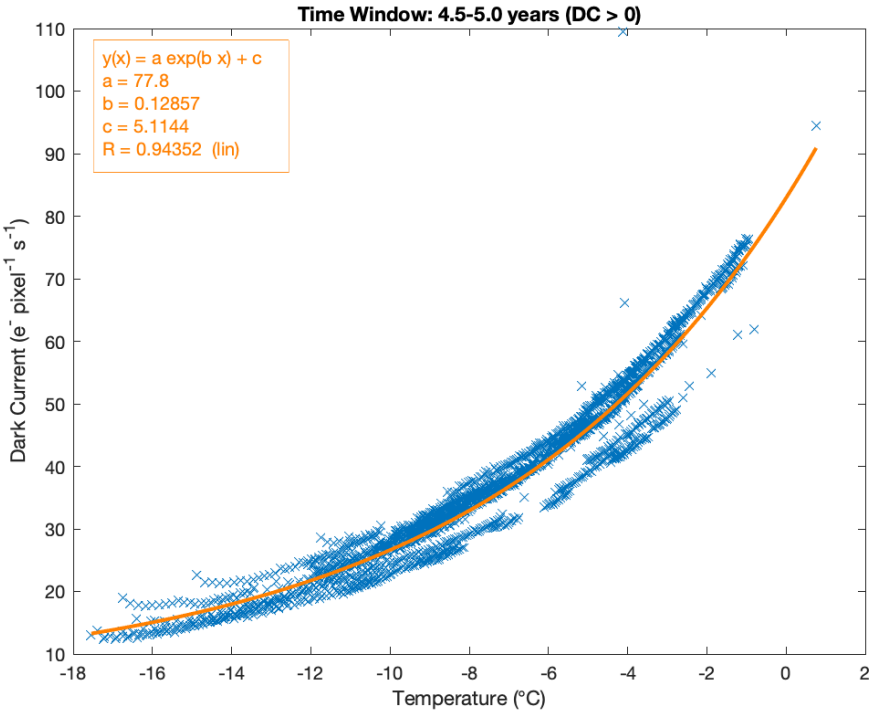


Figure G.9: Dark current vs temperature plot from 4.5-5 years (fitted).

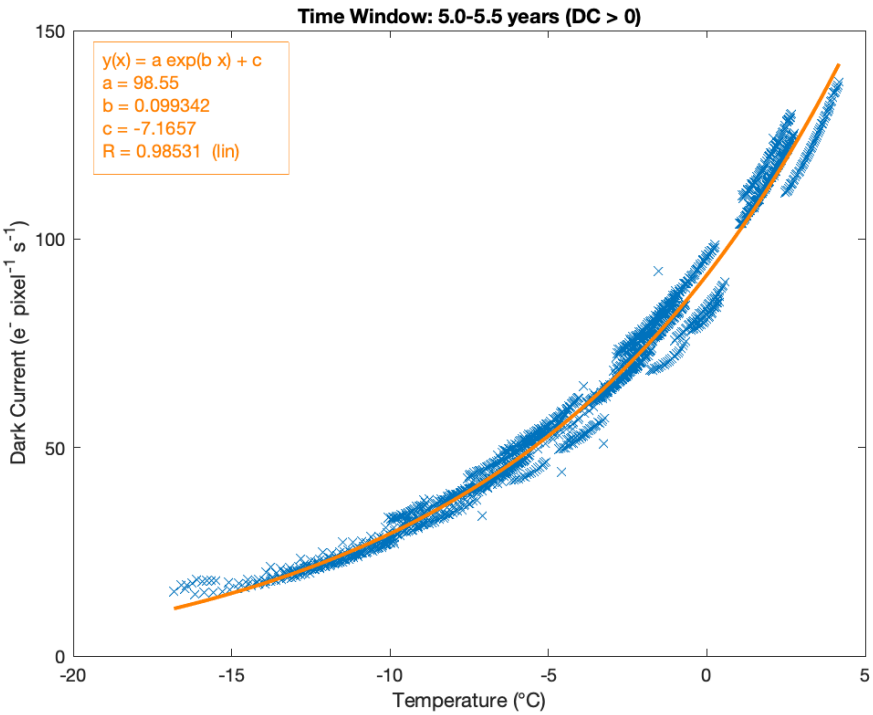


Figure G.10: Dark current vs temperature plot from 5-5.5 years (fitted).

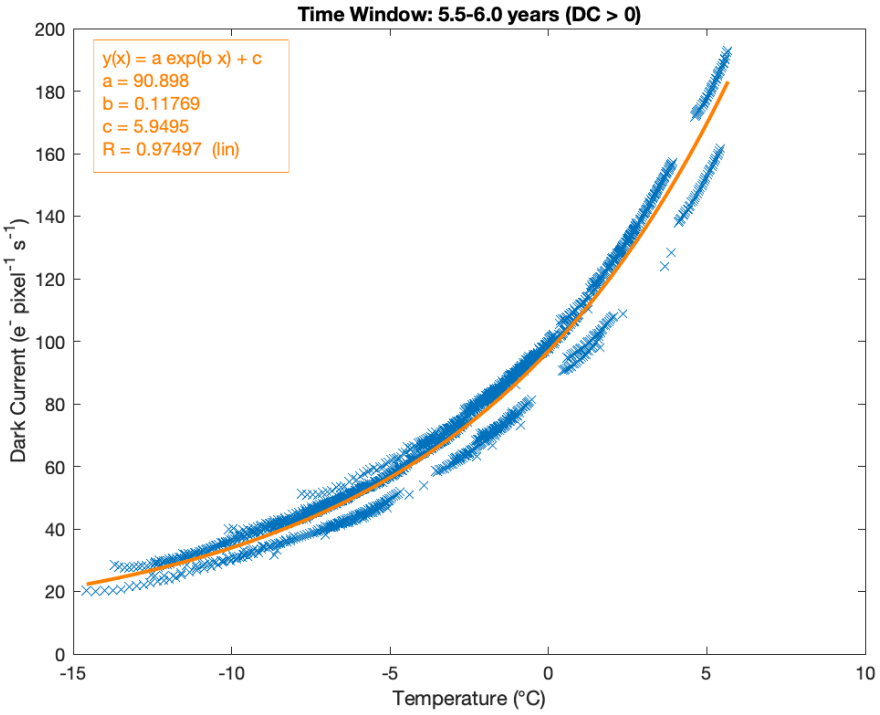


Figure G.11: Dark current vs temperature plot from 5.5-6 years (fitted).

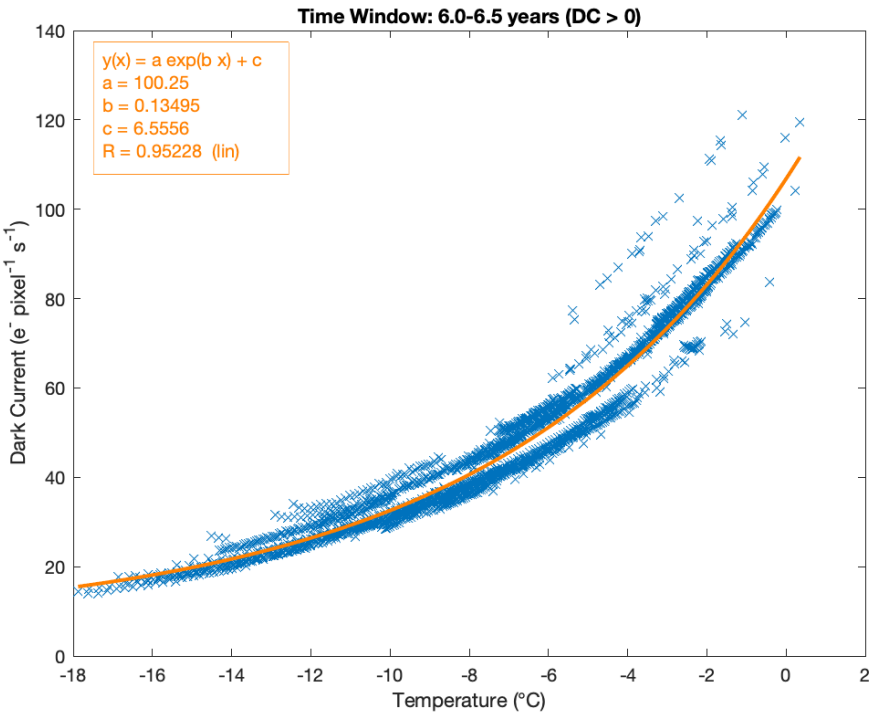


Figure G.12: Dark current vs temperature plot from 6-6.5 years (fitted).

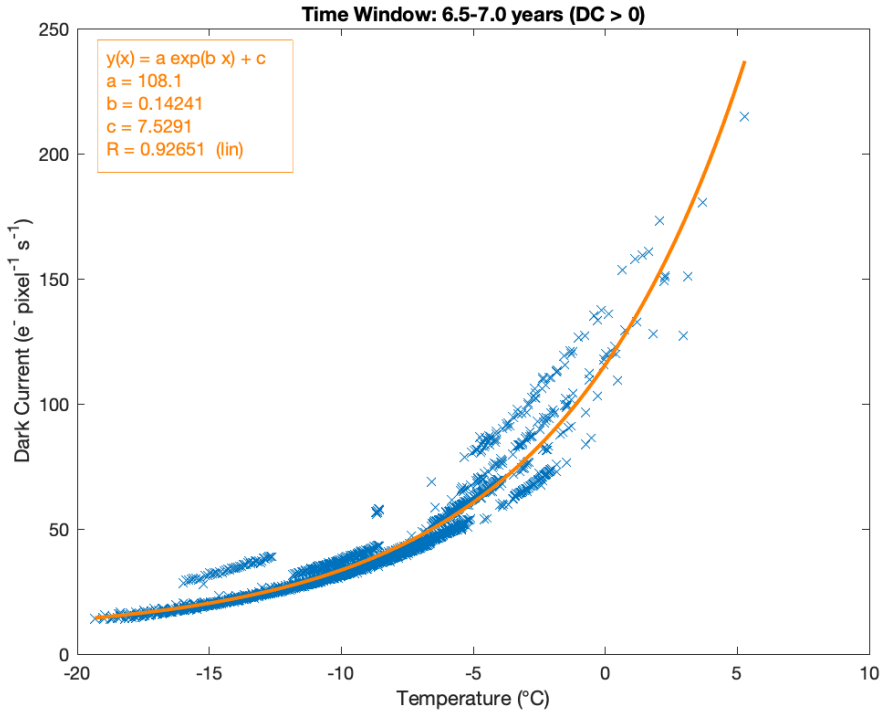


Figure G.13: Dark current vs temperature plot from 6.5-7 years (fitted).

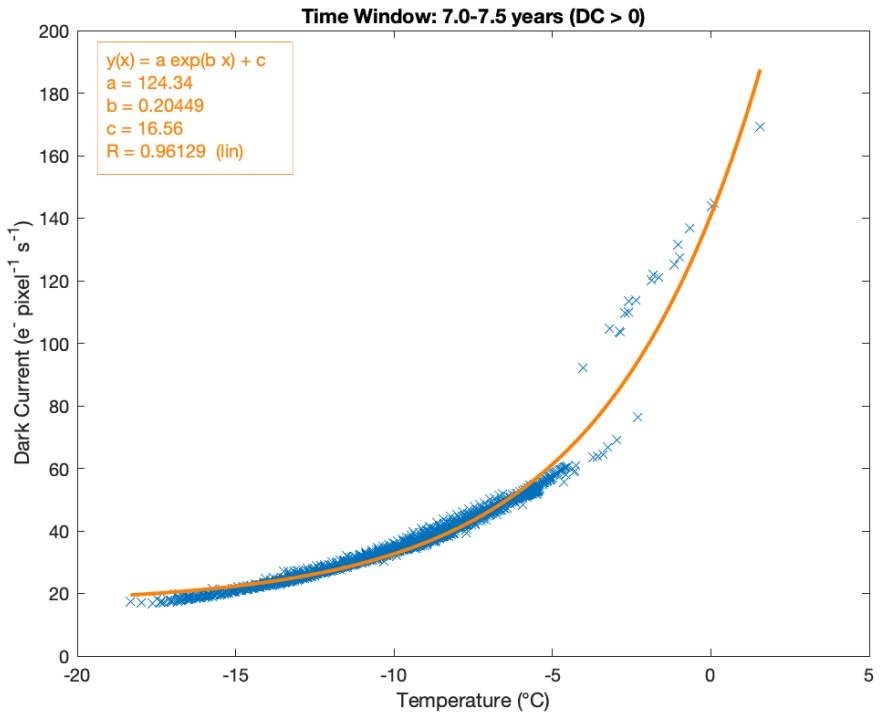


Figure G.14: Dark current vs temperature plot from 7-7.5 years (fitted).

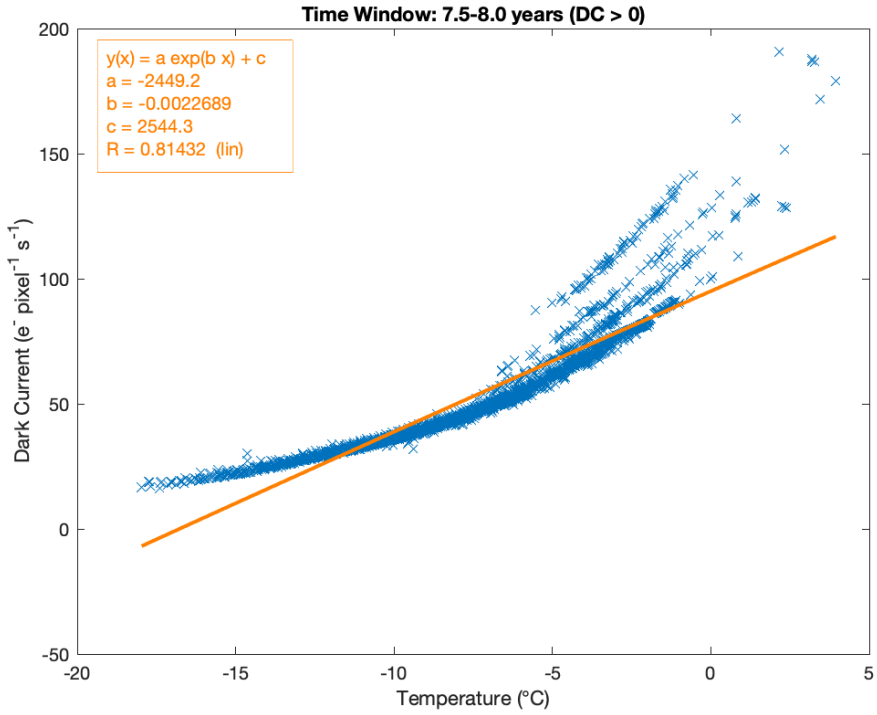


Figure G.15: Dark current vs temperature plot from 7.5-8 years (fitted).

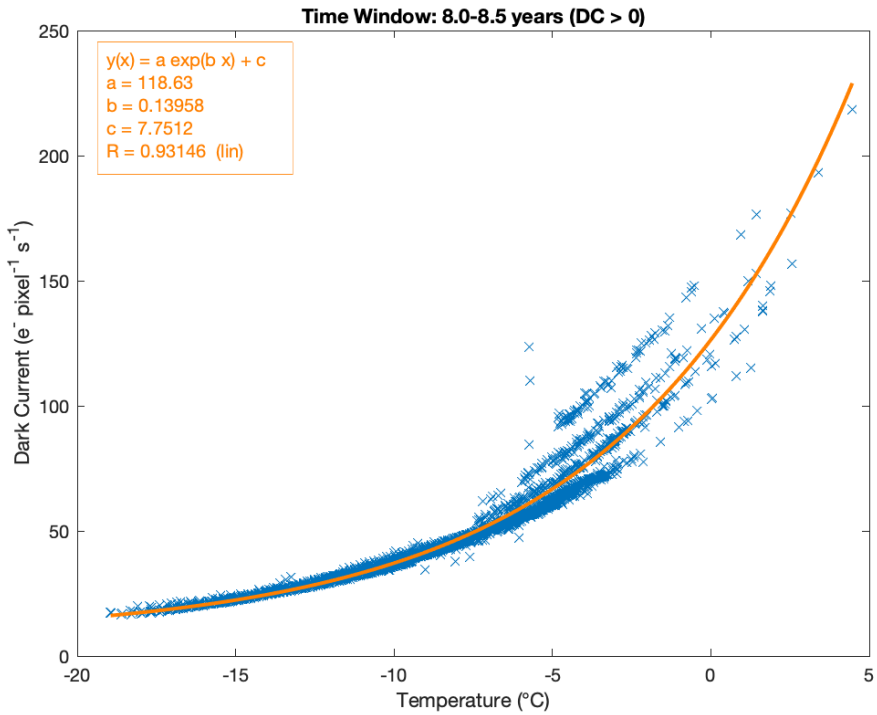


Figure G.16: Dark current vs temperature plot from 8-8.5 years (fitted).

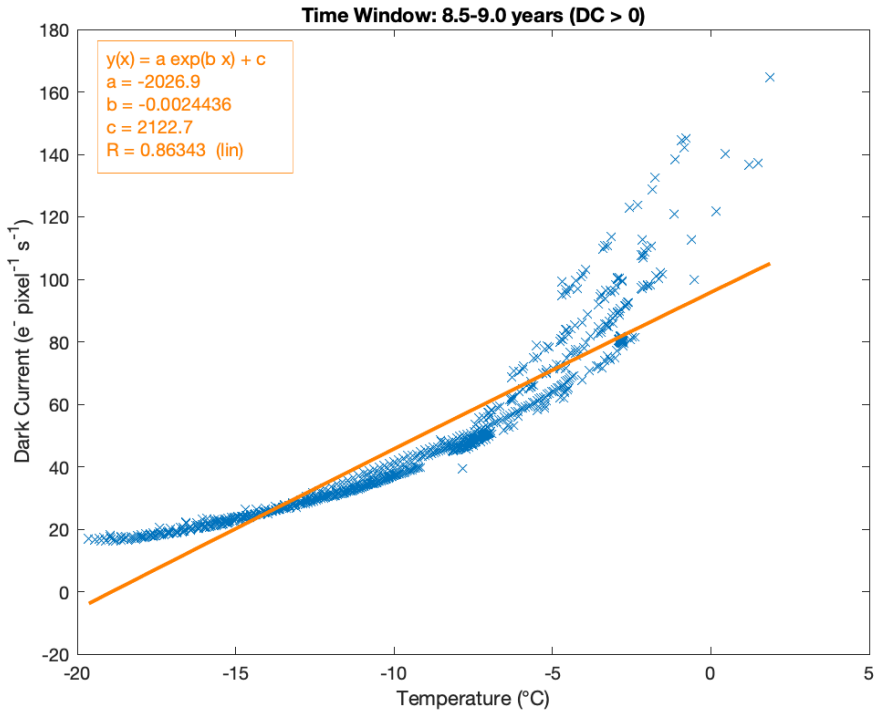


Figure G.17: Dark current vs temperature plot from 8.5-9 years (fitted).

Appendix H

Code for Comparison Plot

```
close all
clear all
%% Load the UVIS mean DC from the flight data
fnm = 'UVIS_dark_frames_mean.h5';
FrameTime = h5read(fnm,'/ObservationalData/FrameTime');
       % Frame time in seconds
FrameTemp = h5read(fnm,'/ObservationalData/FrameTEMP');
       % Frame temperature in degrees C
MeanDC = h5read(fnm,'/ObservationalData/MeanDarkCurrent');
      % Exposure time
%%  Load the ground data
load('uvis_ground_data_final_1.mat')
rad = gnd.em.irradiation;
csl = gnd.csl;

gain = 7.28;
ntime = (FrameTime - gnd.csl.ephemeris_time) / (365*24*3600); %
   Time Normalised

% Filter CSL data for temperatures >= 3 degrees and early mission
   data (<1 year)
i3 = find(csl.MeanTemp >= 3);
ii = find(ntime < 1.0);
Temp_0 = [csl.MeanTemp(i3), FrameTemp(ii)'];
DC_0 = [csl.MeanDC(i3), MeanDC(ii)'];

% Combine both CSL datasets for fitting (normal + filtered)
combined_temp = [csl.MeanTemp, csl.MeanTemp(i3)];
combined_dc = [csl.MeanDC, csl.MeanDC(i3)];

% Fit the combined data to get a middle-ground fit
```

```
ffit_combined = ezfit(combined_temp, combined_dc, 'y = a * exp (b * x) + c', [11, 0.1, 0.7]);
Yfit_combined = ffit_combined.m(1) * exp(ffit_combined.m(2)*(-20:2:24)) + ffit_combined.m(3);

% % Original fits for comparison
% ffit = ezfit(csl.MeanTemp, csl.MeanDC, 'y = a * exp (b * x) + c', [11, 0.1, 0.7]);
% Yfit = ffit.m(1) * exp(ffit.m(2)*(-20:2:24)) + ffit.m(3); % Fit for ground data
%
% f_new = ezfit(Temp_0, DC_0, 'y = a * exp (b * x) + c', [10, 0.1, 10]);
% Yfit_new = f_new.m(1) * exp(f_new.m(2)*(-20:2:24)) + f_new.m(3);

%% pre-characterisation
pre = gnd.em.pre;
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det1 = get_gnd_pre_meanDC(pre.det1,r1,r2,c1,c2);
mDC_det2 = get_gnd_pre_meanDC(pre.det2,r1,r2,c1,c2);
mDC_det3 = get_gnd_pre_meanDC(pre.det3,r1,r2,c1,c2);
%%% get the dark current from halfeol from expose and non-exposed region.
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det1_halfeol = get_gnd_pre_meanDC(rad.halfeol.dark,r1,r2,c1,c2);
mDC_det1_halfeol_notexposed = get_gnd_pre_meanDC(rad.halfeol.dark,r1,r2,50,500);
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det2_eol = get_gnd_pre_meanDC(rad.eol.dark,r1,r2,50,500);
mDC_det2_eol_notexposed = get_gnd_pre_meanDC(rad.eol.dark,r1,r2,c1,c2);
r1 = 30;r2 = 230;
c1 = 600; c2 = 1000;
mDC_det3_eolx2 = get_gnd_pre_meanDC(rad.eolx2.dark,r1,r2,c1,c2);
mDC_det3_eolx2_notexposed = get_gnd_pre_meanDC(rad.eolx2.dark,r1,r2,50,500);
mDC_fm = get_gnd_fm_meanDC(gnd.fm.det1);
figure()
plot(pre.det1.temp,mDC_det1,'xk')
hold on
plot(pre.det2.temp,mDC_det2,'xb')
```

```
plot(pre.det3.temp,mDC_det3,'xr')
xlabel('Temperature (^{\circ}C )')
ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
%% Scatter Plot for In-Flight and Ground Data
figure()
scatter(FrameTemp,MeanDC*gain,6,ntime,"filled") % use ntime to get the color to show time
hold on
scatter(rad.halfeol.dark.temp,mDC_det1_halfeol_notexposed*rad.halfeol.ptc.gain1,12,"c",'o','filled')
scatter(rad.halfeol.dark.temp,mDC_det1_halfeol*rad.halfeol.ptc.gain1,12,"r",'o','filled')
scatter(rad.eol.dark.temp,mDC_det2_eol_notexposed*rad.eol.ptc.gain1,12,"g",'o','filled')
scatter(rad.eol.dark.temp,mDC_det2_eol*rad.eol.ptc.gain1,12,"m",'o','filled')
scatter(rad.eolx2.dark.temp,mDC_det3_eolx2_notexposed*rad.eolx2.ptc.gain1,12,"k",'s','filled')
scatter(rad.eolx2.dark.temp,mDC_det3_eolx2*rad.eolx2.ptc.gain1,12,"k",'o','filled')

plot (combined_temp,combined_dc*gain,'x')
plot((-20:2:24),Yfit_combined*gain,'m') % Combined fit


xlabel('Temperature (^{\circ}C )')
ylabel('Dark Current (e{^-} pixel^{-1} s^{-1})')
cb = colorbar;
caxis([0,9])
ylabel(cb,'Time (years)','FontSize',16)
title ('Dark Current Vs Temperature Plot with increase in time')
legend('Flight Data', 'Half EOL Not Exposed', 'Half EOL Exposed', 'EOL Not Exposed', ...
        'EOL Exposed', 'EOLx2 Not Exposed', 'EOLx2 Exposed', 'Combined CSL', 'Combined CSL Fit', 'Location', 'best')
set(gca,'FontSize',12)
'stop'

function mDC = get_gnd_pre_meanDC(det,r1,r2,c1,c2)
    for i = 1:size(det.frames,3)
        frames = det.frames(r1:r2,c1:c2,i);
        mDC(i) =  mean(frames(:)) / det.integration(1);
    end
end
function mDC = get_gnd_fm_meanDC(det)
```

```
98      for i = 1:size(det.frames,3)
99          frames = det.frames(90:230,50:1030,i);
100         mDC(i) =  mean(frames(:)) / det.integration(i);
101     end
102 end
```

Listing H.1: Code for producing the comparison plot of both datasets