Project plan

High-accuracy radiation pressure modeling for LRO

Dominik Stiller

January 26, 2023

Nomenclature

α	View angle; angle between surface normals of source and target	rad
λ	Longitude	rad
ν	Shadow function; $\nu = 0$ means total eclipse, $\nu = 1$ means full radiation	–
Φ	Radiant power	W
ϕ	Latitude	rad
σ	Stefan–Boltzmann constant	$\mathrm{W/(m^2\,K^4)}$
θ	Incidence angle; angle between surface normal and incident radiation	rad
$\mathbf{n}$	Normal vector of a surface	–
$\mathbf{r}$	Vector from source to target; depends on context	m
$\hat{\mathbf{r}}$	Unit vector from source to target	–
A	Area on source that receives radiation	$\mathrm{m^2}$
C_a	Absorptivity	–
C_d	Diffuse reflectivity	–
C_r	Radiation pressure coefficient	–
C_s	Specular reflectivity	–
E	Irradiance/flux density	$\mathrm{W/m^2}$
E_s	Solar irradiance	$\mathrm{W/m^2}$
$E_{s,1\,\mathrm{AU}}$	Total solar irradiance (TSI) at 1 AU distance	$\mathrm{W/m^2}$
L_s	Solar luminosity	W

m	Mass	kg

1 Introduction

Scientific results obtained from a combination of LRO altimetry, GRAIL gravity field determination and Lunar Laser Ranging can in some cases lead to conflicting results on specific details on lunar geodetic properties (tides, rotation, etc.) Although minor, these discrepancies may not allow the exceptionally accurate data sets that are available to be processed to their inherent accuracy.

For this project, one possible contributor to this issue will be analyzed: errors in non-conservative force modelling of the spacecraft. In particular, this project will investigate the impact of various level of detail of the radiation pressure modelling of the LRO spacecraft, with the aim of contributing to a more robust error budget of the attained orbit determination results. This leads to the research question:

What is the quantitative influence of using high-accuracy radiation pressure models on the attainable orbit precision for the Lunar Reconaissance Orbiter?

The models will be implemented in Tudat, an open-source simulation framework for astrodynamics, developed by TU Delft.

2 Models

On the highest level, we divide radiation pressure models into sources and targets. Sources emit or reflect electromagnetic radiation onto the target, which experiences an acceleration. For sources, we regard direct solar, albedo and thermal radiation. For targets, we regard cannonball and paneled models with and without self-shadowing. Only radiation pressure due to incoming radiation and instantaneous reradiation is considered. Radiation pressure due to delayed thermal radiation of the spacecraft itself as described by Wetterer *et al.* [1] will not be treated.

Source models and target models can be developed independently, then mixed and matched. The interface between sources and targets consists of 2 quantities:

- Irradiance, or flux density, E from source at target
- Unit vector $\hat{\mathbf{r}}$ from source to target

These can be combined into the directional irradiance $\mathbf{E} = E\hat{\mathbf{r}}$. This assumes that all radiation is parallel, i.e. originates from a distant point, which is a good approximation for distant sources (e.g., the Sun at 1 AU distance). Sources for which the spatial extent is relevant (e.g., Earth albedo radiation in LEO) can be discretized into multiple point sources.

We treat all radiation equally as total flux, independently of wavelength. While most optical properties such as reflectivity are physically functions of wavelength, characterizing their dependence is challenging in practice. This leads us to using the same surface properties across wavelengths, even though albedo radiation is in the visible range while thermal radiation is infrared. However, we make provisions for wavelength-dependent extensions in the future.

2.1 Sources

The most significant source of radiation pressure in Earth and lunar orbits is direct solar radiation. The solar irradiance E_s can be found through the luminosity of the sun or total solar irradiance (TSI) at 1 AU:

$$E_s = \nu \frac{L_s}{4\pi \|\mathbf{r}\|^2} = \nu E_{s,1\,\mathrm{AU}} \frac{1\,\mathrm{AU}}{\|\mathbf{r}\|^2} \tag{1}$$

where the solar luminosity is taken to be $L_s = 3.828 \times 10^{26}\,\mathrm{W}$ [2]. This leads to the solar constant of $E_{s,1\,\mathrm{AU}} = 1360.8\,\mathrm{W/m^2}$ at $\|\mathbf{r}\| = 1\,\mathrm{AU}$ [3]. Note that this luminosity and irradiance are time averages and vary due to sunspot darkening and facular brightening [4]. Observational time series for TSI exist [5] such that the time-varying solar irradiance at any distance can be found using the inverse square law.

$\nu \in [0, 1]$ is the shadow function, scaling the received irradiance according to the visible portion of the sun, which may be occulted by other bodies. A conical model dividing space into regions of full sunlight, penumbra and umbra due to a single body is the standard [6]. This model could be extended to consider (partial) occultation by two bodies as described by Zhang *et al.* [7].

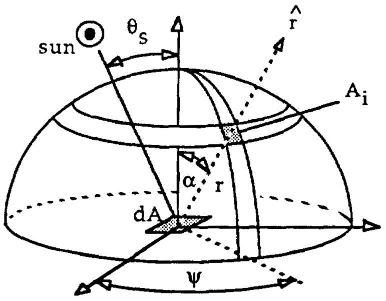

Figure 1: Geometry of albedo radiation. dA is the source element, A_i is the target.

Albedo radiation, reflected by planet surfaces, is much smaller but still significant. Albedo requires knowledge of properties of the radiation source (for our intents and purposes, the Sun) and the reflecting body. The solar irradiance E_s and angle between reflecting surface normal and Sun θ_s determine the incident irradiance onto the source surface element dA. The reflected radiation depends on the albedo distribution $a = a(\lambda, \phi)$ which may vary with longitude λ and latitude ϕ. The received radiation depends on the view angle α, which is the angle between the surface normals of source and target. This geometry is shown in Figure 1. The reflected radiance depends on the reflectance type. For Earth, purely diffuse Lambertian reflectance is a fair assumption [8]. More sophisticated reflectance considering land cover exist, for example using kernel-based bidirectional reflectance distribution functions (BRDF) as decribed by Lucht *et al.* [9]. The irradiance from dA at the target due to albedo is then given by [8]:

$$E = a \cos\theta_s E_s \frac{dA \cos\alpha}{\pi \|\mathbf{r}\|^2} \tag{2}$$

where $a \cos\theta_s E_s$ is the reflected irradiance. Note that albedo radiation only exists if dA receives sunlight. Shadow calculations could also be included but are more involved for albedo models, since both the incoming solar radiation and outgoing albedo radiation could be affected by occultation. Calculations are further complicated since common occultation models assume spherical sources and not flat surface elements.

The simplest choice for the lunar albedo is the average value of $a = 0.12$ [10]. A more detailed lunar albedo distribution is the 15x15 spherical harmonics model by Floberghagen *et al.* [11]. However, for calculations, paneling of the source is more convenient. Knocke *et al.* [8] introduce a spherical cap centered at the subsatellite point, which is divided into rings of panels of constant albedo, tangent to the source surface at their center. Equation (2) is then evaluated for each panel dA. We call this *dynamic paneling.* Alternatively, the whole body could be paneled independently of the satellite position (*static paneling*). Such an approach including evenly distributed panels is described by Wetterer *et al.* [1].

Similarly, the thermal radiation can be described, scaled by the emissivity e. Additionally, there is a factor of $1/4$, which is the ratio between receiving and emitting surface. Then the irradiance from dA at the target due to thermal radiation is given by [8]:

$$E = \frac{eE_s}{4}\frac{dA\cos\alpha}{\pi\|\mathbf{r}\|^2} \tag{3}$$

where $eE_s/4$ is the emitted exitance. Thermal radiation exists independent of incident sunlight and is therefore constant. The simplest model for lunar emissivity is a constant value of $e = 0.95$ [10].

Alternatively, a latitude- and local time-dependent temperature distribution of the lunar surface can be assumed [12]. By the Stefan–Boltzmann law, the irradiance at the target due to the thermal radiation is given by:

$$E = e\sigma T^4\frac{dA\cos\alpha}{\pi\|\mathbf{r}\|^2} \qquad T = \max\left(T_{\max}(\cos\theta_s)^{1/4}, T_{\min}\right) \tag{4}$$

where $T_{\max} = 375\,\mathrm{K}$ and $T_{\min} = 100\,\mathrm{K}$. Note that the maximum irradiance from Equation (4) is about four times higher than that from Equation (3) since $\sigma T_{\max}^4 \approx E_s$, but the irradiance from Equation (4) varies as the dA moves away from the subsolar point (θ_s increases) and cools down.

Instead of modeling outgoing planetary fluxes, they can also be observation-based. For Earth, CERES provides time series for shortwave and longwave fluxes with up to hourly and 1° resolution [13]. For the Moon, irradiance spectra have been published by Kieffer *et al.* [14] and Sun *et al.* [15]. However, they are constant in time and provide a single spectrum for only the Earth-facing lunar side. Therefore, they are of little use for radiation pressure models in lunar orbits, but can be used for Earth orbits.

2.2 Targets

The *cannonball model* is the simplest model for target acceleration due to radiation pressure. The target is modeled as a sphere such that lateral accelerations cancel and there is only an acceleration away from the source along $\hat{\mathbf{r}}$. The cross-sectional area A is independent of orientation, and surface properties (reflectance and absorptivity) are captured in the radiation pressure coefficient C_r. Then the acceleration of a target with mass m is given by [8]:

$$\mathbf{a} = C_r\frac{A}{m}\frac{E}{c}\hat{\mathbf{r}} \tag{5}$$

A more sophisticated paneled target model discretizes the spacecraft into n panels with area A and normal vector $\mathbf{n}$. This also means that the incidence angle θ differs per panel. Their surface is characterized by the absorptivity C_a, diffuse reflectivity C_d and specular reflectivity C_s, which obey $C_a + C_d + C_s = 1$. Anisotropy can be accounted for using BRDFs as described by Wetterer *et al.* [1]. However, we assume Lambertian diffuse reflectance and instantaneous Lambertian reradiation of absorbed radiation. Then the acceleration of the whole target due to all target panels and a single source is given by [16]:

$$\mathbf{a} = \frac{1}{m}\frac{E}{c}\sum_{j=1}^{n} A\cos\theta\left[(C_a + C_d)\left(\hat{\mathbf{r}} - \frac{2}{3}\mathbf{n}\right) - 2C_s\cos\theta\,\mathbf{n}\right] \tag{6}$$

where all quantities inside the summation except $\hat{\mathbf{r}}$ are specific to panel j. For the LRO, these panel properties are given by Smith *et al.* [17]. Self-shadowing could also be included here. Mazarico *et al.* [18] describe an algorithm to modify the effective area due to self-shadowing and describe the effect on the spacecraft trajectory as significant. Kenneally *et al.* [19] perform raytracing for self-shadowing with BRDFs on GPUs.

In case of a paneled source, the total acceleration is the vectorial sum of these contributions over all m source panels:

$$\mathbf{a} = \frac{1}{m}\sum_{i=1}^{m}\frac{E}{c}\sum_{j=1}^{n} A\cos\theta\left[(C_a + C_d)\left(\hat{\mathbf{r}} + \frac{2}{3}\mathbf{n}\right) + 2C_s\cos\theta\,\mathbf{n}\right] \tag{7}$$

where E is the irradiance due to the i-th source panel.

3 Options

Radiation pressure models range from the simple baseline model to our extended model, but even more configuration options are possible. An extensive overview over options for radiation pressure modeling is given in [20, Sec. 2]. This list contains all options that have been explored in literature and that Tudat may want to support in the future, hence provisions for extensibility should be made. However, only the **bold options** will be implemented in this project.

- Body:
  - **Mass**
  - **Position and orientation**
  - Shape (for occultation, spherical or oblate spheroid)
  - Athmosphere (for refraction influencing occultation)
  - **Radiation source and/or target**
  - **Temperature distribution (in case Lemoine thermal model is used)**
- Point source:

  - **Luminosity or TSI (constant or time-varying)**
  - Continuous or discrete emission spectrum (i.e. function of wavelength, binned or visible + infrared)

- Paneled source:
  - **Original radiation source**
  - **Albedo and emissivity distribution (constant, per panel or as spherical harmonics)**
  - **Thermal emission model (Knocke or Lemoine)**
  - **Albedo reflection law (constant** or BRDF, possibly depending on wavelength)
  - **Paneling resolution**
  - **Static or dynamic paneling**
  - **Occultation of albedo panels**
  - Observation-based fluxes (like CERES measurements) instead of modeled fluxes

- Cannonball target:
  - **Cross-sectional area**
  - **Radiation pressure coefficient**

- Paneled target:
  - **Area of each panel**
  - **Position and orientation of each panel (constant** or time-varying (for HGA or SA), from CK kernels or e.g. aligned with sun, position only relevant for self-shadowing and self-reflection)
  - With or **without** self-shadowing and self-reflection
  - **Absorptivity, specular reflectivity and diffuse reflectivity of each panel (constant** or depending on wavelength, possibly time-varying due to degradation)
  - **Reflection law (constant** or BRDF, possibly depending on wavelength)
  - **Thermal reradiation (instantaneous** or from temperature distribution considering heat conduction and generation, should be implemented as separate acceleration class if not instantaneous)

4 Verification & Validation

Verification will check whether the models presented in this document were implemented correctly, based on manual calculations and values from literature. Validation will check whether the mathematical models themselves give sensible results. Both will be implemented as unit tests. Existing radiation pressure unit tests within Tudat will be reused and adapted to avoid regression. However, existing tests include a lot of logic that itself may be flawed. Therefore, the reworked unit tests will be more straightforward, at the cost of duplicate code.

The lunar radiation model can be rougly validated with the peak lunar irradiance in LRO's lunar orbit of $1330\,\mathrm{W/m^2}$ [21]. To validate the simulation setup, I will also propagate LRO's orbit and check consistency with ephemerides from SPICE SPKs. While (possibly significant) differences are expected in both, the error should be reasonable and orders of magnitude of results similar.

5 Result analysis

The question to be answered is *What is the quantitative influence of using high-accuracy radiation pressure models on the attainable orbit precision for the Lunar Reconaissance Orbiter?* The answer will not include statements about absolute or relative precision improvements, since there is no ground truth. Rather, the answer will give tendencies about how different models and parameters influence orbital elements.

The simulation setup for gathering results will be varied to investigate different levels of accuracy. In the simplest form, the radiation pressure models only contain a direct solar radiation source and a cannonball target without occultation (*baseline model*) In the most complete form (*extended model*), the setup looks as follows:

- Sun:
  - Ephemeris from DE 421 (used by JPL for LRO ephemeris generation)
  - Gravity field
  - Direct solar radiation source
- Earth:
  - Ephemeris from DE 421
  - Gravity field
  - Occulting body for direct solar and lunar albedo radiation
- Moon:
  - Global origin

  - Ephemeris and MOON_PA frame from DE 421
  - Gravity field
  - Albedo radiation source (paneled Moon with albedo obtained from DLAM-1)
  - Thermal radiation source (paneled Moon)
  - Occulting body for direct solar radiation

- LRO:
  - Propagated (translational) for 565 min, corresponds to about 5 orbital revolutions
  - Rotational ephemeris
  - Initial ephemeris from LRO reprocessed spacecraft ephemeris (`fdf36_...`) during regular science mission at 50 km altitude, ensure no stationkeeping but eclipses occured during propagation period (start at 26 June 2010 06:00:00)
  - Paneled radiation pressure target with areas and coefficients from Smith *et al.* [17] (assume SA is pointed towards Sun and HGA is pointer towards Earth)
  - No self-shadowing, unless time permits

The result analysis is inspired by Vielberg *et al.* [20] for LEO satellites, but less involved since a lot of details (e.g. observed outgoing fluxes, observed solar irradiance, land coverage) do not exist for or apply to the Moon. The analysis will consider the following aspects:

- Accelerations due to each radiation pressure component (direct solar, albedo, thermal) in radial, cross-track and along-track directions with extended model (cf. [20, Fig. 3])
- Dependence of accelerations on position in orbit and time (cf. [20, Fig. 7]), correlate with relative sun position and albedo map
- Sensitivity analysis for albedo and target reflection/absorption coefficients (since these parametrizations are often inaccurate, investigating influence of their errors is important)
- Effect of different levels of detail of radiation pressure models on accelerations (cf. [20, Fig. 8]) and Keplerian orbit elements (e.g., how does addition of albedo radiation change semi-major axis?), moving from baseline model towards extended model
  - Baseline model: only direct solar radiation source, cannonball target, no occultation
  - For source, add albedo and thermal radiation (vary paneling resolution, constant and spherical harmonics albedo, constant or varying thermal radiation from Equations (3) and (4), with/without instantaneous reradiation, dynamic/static paneling)
  - For target, switch to paneled model

- Add multiple occultation
- Compare mean difference and RMS difference w.r.t. baseline in radial, cross-track and along-track directions after propagation arc
- Compare Keplerian orbits w.r.t. baseline after propagation arc
- Measure performance impact of increased level of detail through wall-clock and/or CPU time, and memory footprint

6 Code design

All models presented in Section 2 will be implemented. The following Python-like pseudocode shows the classes and their interactions. The code is not complete but only contains parts relevant for radiation pressure computations.

All radiation source and target calculations are performed in their respective local frames. The `class RadiationPressureAcceleration` handles reference frame transformations between them. This simplifies and decouples source and target code.

```
################################################################################################
#       ENVIRONMENT                                                                            #
################################################################################################
class Body:
   """Models Sun, planets and spacecraft"""
   position: Vector3
   mass: double

   # List of all sources originating from this body
   # For sun: PointRadiationSourceModel for direct solar radiation
   # For planets: PaneledRadiationSourceModel for albedo + thermal radiation
   # For spacecraft: -
   radiationSourceModel: RadiationSourceModel

   # Target model (for bodies undergoing radiation pressure acceleration)
   # For sun: -
   # For planets: -
   # For spacecraft: CannonballRadiationPressureTargetModel or
   #    PaneledRadiationPressureTargetModel
   radiationPressureTargetModel: RadiationPressureTargetModel


class RadiationPressureAcceleration(AccelerationModel3d):
   """
   Radiation pressure acceleration from a single source onto a single target.
   """
   source: Body  # e.g. Sun
   target: Body  # e.g. LRO
   occultationModel: OccultationModel

   def updateMembers(currentTime: double) -> void:
        """"Evaluate radiation pressure acceleration at current time step"""

        # rotate target position to source-fixed frame
        irradianceList = source.radiationSourceModel \
                           .evaluateIrradianceAtPosition(target.position)
        # rotate irradiances to target-fixed frame

        force = Vector3.Zero()
        # Iterate over all source panels and their fluxes
        for sourceIrradiance, sourceCenter in irradianceList: # i=1..m
            sourceToTargetDirection = (target.position - sourceCenter).normalize()
            # rotate sourceToTargetDirection to target-fixed frame
            sourceIrradiance = occultationModel.applyOccultation(sourceIrradiance)
            force += target.evaluateRadiationPressureForce(sourceIrradiance,
                                                           sourceToTargetDirection)
        # rotate force to global frame
        currentAcceleration = force / target.mass


abstract class OccultationModel:
    occultingBodies: list[Body]  # e.g. Earth and Moon

    def applyOccultation(sourceIrradiance: double, occultedBody: Body, targetBody: Body) -> double:
        pass


abstract class ShadowFunctionOccultation:
    def applyOccultation(irradiance: double, occultedBody: Body, targetBody: Body) -> double:
        # Calculate using Montenbruck 2000 or Zhang 2019 equations
        # Compared to current function in Tudat, takes multiple occulting bodies
        shadowFunction = ...
        irradiance *= shadowFunction
        return irradiance


abstract class ReflectionLaw:
    # Models a constant BRDF
    def evaluateReflectedFraction(surfaceNormal: Vector3, incomingDirection: Vector3,
                                  observerDirection: Vector3) -> double:
        # Calculate azimuth/polar angles for incoming and observer directions
        # Evaluate BRDF
        reflectedFraction = ...  # [1 / sr]
        return reflectedFraction

    def evaluateReactionVector(surfaceNormal: Vector3, incomingDirection: Vector3) -> Vector3:
        # integrates Wetterer Eq 2


class LambertianReflectionLaw(ReflectionLaw):
    # Possibly subclass of SpecularDiffuseMixReflectionLaw
    reflectance: double  # identical with albedo

    def evaluateReflectedFraction(surfaceNormal: Vector3, incomingDirection: Vector3,
                                  observerDirection: Vector3) -> double:
        return reflectance / PI

```

```

class SpecularDiffuseMixReflectionLaw(ReflectionLaw):
   absorptivity: double
   specularReflectivity: double
   diffuseReflectivity: double

   def evaluateReactionVector(surfaceNormal: Vector3, incomingDirection: Vector3) -> Vector3:
      # evaluates Wetterer Eq 5


################################################################################################
#       SOURCES                                                                                #
################################################################################################

abstract class RadiationSourceModel:
   source: Body  # The source that this model belongs to
                 # For albedo, this is the reflecting body, not the Sun

   def evaluateIrradianceAtPosition(targetPosition: Vector3) -> list[Vector3]:
      """
      Calculate irradiance at target position, also return source position. Subclasses
      are aware of source geometry. Return a list of tuples of flux and origin to
      support multiple fluxes with different origins for paneled sources.
      """
      pass


#===============================================================================================
#       Point radiation source
#===============================================================================================
class IsotropicPointRadiationSourceModel(RadiationSourceModel):
   """Point source (for Sun)"""
   luminosityModel: LuminosityModel

   def evaluateIrradianceAtPosition(targetPosition: Vector3) -> list[tuple[double, Vector3]]:
      sourcePosition = source.position
      distanceSourceToTarget = targetPosition.norm()
      luminosity = luminosityModel.evaluateLuminosity()
      irradiance = luminosity / (4 * PI * distanceSourceToTarget**2)  # Eq. 1
      return [(irradiance, sourcePosition)]


abstract class LuminosityModel:
   """Gives luminosity for a point source"""

   def evaluateLuminosity() -> double:
      pass


class ConstantLuminosityModel(LuminosityModel):
   """Gives luminosity directly"""
   luminosity: double

   def evaluateLuminosity():
      return luminosity

```

```

class IrradianceBasedLuminosityModel(LuminosityModel):
   """Gives luminosity from irradiance at certain distance (e.g., TSI at 1 AU)"""
   irradianceAtDistance: double  # could also be a time series from TSI observations
   distance: double

   def evaluateLuminosity():
      luminosity = irradianceAtDistance * 4 * PI * distance**2
      return luminosity


#================================================================================
#       Paneled radiation source
#================================================================================
class PaneledRadiationSourceModel(RadiationSourceModel):
   """Paneled sphere (for planet albedo + thermal radiation)"""
   originalSource: Body  # Usually the Sun, from where incoming radiation originates
   occultingBodies: list[Body]  # For Moon as source, only Earth occults

   panels: list[SourcePanel]

   def _generatePanels():
      # Panelize body and evaluate albedo for panels. For static paneling
      # (independent of spacecraft position), generate once at start of simulation,
      # Query SH albedo model here if available here, or load albedos and
      # emissivities from file
      panels = ...

   def evaluateIrradianceAtPosition(targetPosition: Vector3) -> list[tuple[double, Vector3]]:
      # For dynamic paneling (depending on target position, spherical cap centered
      # at subsatellite point as in Knocke 1988), could regenerate panels here
      # (possibly with caching), or create separate class
      sourceBodyPosition = source.position

      ret = []
      for panel in panels: # i=1..m
         sourcePosition = sourceBodyPosition + panel.relativeCenter

         irradiance = 0
         for radiationModel in panel.radiationModels:
            irradiance += radiationModel.evaluateIrradianceAtPosition(
               panel, targetPosition)

         ret.append((irradiance, sourcePosition))
      return ret


class RadiationSourcePanel:
   area: double
   relativeCenter: Vector3  # Panel center relative to source center
   normal: Vector3  # body-fixed

   radiationModels: list[PanelRadiationModel]


abstract class PanelRadiationModel:
```

```python
    def evaluateIrradianceAtPosition(panel: RadiationSourcePanel, targetPosition: Vector3) \
            -> double:
        pass


class AlbedoPanelRadiationModel(PanelRadiationModel):
    # Usually LambertianReflectionLaw
    reflectionLaw: ReflectionLaw

    def evaluateIrradianceAtPosition(panel: RadiationSourcePanel, targetPosition: Vector3) \
            -> double:
        if not isVisible(panel, targetPosition):
            # Panel hidden at target position
            return 0

        # for received radiation at panel
        shadowFunction = calculateShadowFunction(originalSource, occultingBodies, panel.center)

        reflectedFraction = reflectionLaw.evaluateReflectedFraction(panel.normal,
            originalSourceDirection, targetDirection)
        albedoIrradiance = \
            shadowFunction * ...  # albedo radiation calculation, Eq. 2
        return albedoIrradiance


class DelayedThermalPanelRadiationModel(PanelRadiationModel):
    # Based on Knocke (1988)
    emissivity: double
    temperature: double

    def evaluateIrradianceAtPosition(panel: RadiationSourcePanel, targetPosition: Vector3) \
            -> double:
        thermalIrradiance = emissivity * ...  # thermal radiation calculation, Eq. 3
        return thermalIrradiance


class AngleBasedThermalPanelRadiationModel(PanelRadiationModel):
    # Based on Lemoine (2013)
    emissivity: double

    def evaluateIrradianceAtPosition(panel: RadiationSourcePanel, targetPosition: Vector3) \
            -> double:
        temperature = max(...)
        thermalIrradiance = emissivity * ...  # thermal radiation calculation, Eq. 4
        return thermalIrradiance


class ObservedPanelRadiationModel(PanelRadiationModel):
    """Based on observed fluxes (e.g. from CERES, also requires angular distribution model)"""
    def evaluateIrradianceAtPosition(targetPosition: Vector3):
        observedIrradiance = ...
        return observedIrradiance


################################################################################################
#       TARGETS                                                                                #
```

```
##############################################################################################

abstract class RadiationPressureTargetModel:
   def evaluateRadiationPressureForce(sourceIrradiance: double,
                                   sourceToTargetDirection: Vector3) -> Vector3:
      """
      Calculate radiation pressure force due to a single source panel onto whole target
      """
      pass


class CannonballRadiationPressureTargetModel(RadiationPressureTargetModel):
   area: double
   coefficient: double

   def evaluateRadiationPressureForce(sourceIrradiance: double,
                                   sourceToTargetDirection: Vector3) -> Vector3:
      force = sourceIrradiance * area * coefficient * ...
      return force


class PaneledRadiationPressureTargetModel(RadiationPressureTargetModel):
   panels: List[TargetPanel]

   def evaluateRadiationPressureForce(sourceIrradiance: double,
                                   sourceToTargetDirection: Vector3) -> Vector3:
      force = Vector3.Zero()
      for panel in panels: # j=1..n
         if not isVisible(panel, sourceToTargetDirection):
            # Panel pointing away from source
            break

         reactionDirection = panel.reflectionLaw.evaluateReactionDirection(panel.normal, sourceToTargetDirec
         force += sourceIrradiance * panel.area * reactionDirection * ...
      return force


class TargetPanel:
   area: double
   normal: Vector3  # body-fixed
   center: Vector3  # body-fixed

   reflectionLaw: ReflectionLaw
```

7 Implementation plan

A minimum viable version will be implemented first, including only a point source and a cannonball target (the baseline model). Once this version works and has been verified, the more complex models can follow. All implementations also include unit tests for verification and validation. The implementation plan is as follows:

1. Implement baseline model

   a) Implement `class IsotropicPointRadiationSourceModel` with abstract base class

   b) Implement `class CannonballRadiationPressureTargetModel` with abstract base class

   c) Implement `class RadiationPressureAcceleration` without occultation

   d) Implement LRO simulation (baseline model)

   e) Verify functionality and check if design makes sense

2. Implement `class PaneledRadiationPressureTargetModel`

3. Implement `class PaneledRadiationSourceModel` with static paneling (constant albedo until we get access to DLAM-1)

4. Implement `class OccultationGeometry` for single occulting body and include in `class RadiationPressureAcceleration`

5. Implement LRO simulation (extended model) as described in Section 5

6. Validate complete simulation

7. Implement extra items, if time permits

   a) Implement spherical harmonics lunar albedo model DLAM-1 from Floberghagen *et al.* [11], if we get access

   b) Implement occultation by two bodies from Zhang *et al.* [7]

   c) Implement `class PaneledRadiationSourceModel` with dynamic paneling

   d) Implement self-shadowing from Mazarico *et al.* [18]

   e) Optimize

References

[1] C. J. Wetterer *et al.*, “Refining space object radiation pressure modeling with bidirectional reflectance distribution functions,” *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 185–196, Jan. 2014. DOI: 10.2514/1.60577.

[2] A. Pra *et al.*, “NOMINAL VALUES FOR SELECTED SOLAR AND PLANETARY QUANTITIES: IAU 2015 RESOLUTION b3,” *The Astronomical Journal*, vol. 152, no. 2, p. 41, Aug. 2016. DOI: 10.3847/0004-6256/152/2/41.

[3] M. Wild, D. Folini, C. Schär, N. Loeb, E. G. Dutton, and G. König-Langlo, “The global energy balance from a surface perspective,” *Climate Dynamics*, vol. 40, no. 11-12, pp. 3107–3134, Nov. 2012. DOI: 10.1007/s00382-012-1569-8.

[4] G. Kopp, "Magnitudes and timescales of total solar irradiance variability," *Journal of Space Weather and Space Climate*, vol. 6, A30, 2016. DOI: 10.1051/swsc/2016025.

[5] S. Dewitte and N. Clerbaux, "Measurement of the earth radiation budget at the top of the atmospherea review," *Remote Sensing*, vol. 9, no. 11, p. 1143, Nov. 2017. DOI: 10.3390/rs9111143.

[6] O. Montenbruck and E. Gill, *Satellite Orbits*. Springer Berlin Heidelberg, Dec. 2000, 371 pp. DOI: 10.1007/978-3-642-58351-3.

[7] R. Zhang, R. Tu, P. Zhang, J. Liu, and X. Lu, "Study of satellite shadow function model considering the overlapping parts of earth shadow and moon shadow and its application to GPS satellite orbit determination," *Advances in Space Research*, vol. 63, no. 9, pp. 2912–2929, May 2019. DOI: 10.1016/j.asr.2018.02.002.

[8] P. Knocke, J. Ries, and B. Tapley, "Earth radiation pressure effects on satellites," in *Astrodynamics Conference*, American Institute of Aeronautics and Astronautics, Aug. 1988. DOI: 10.2514/6.1988-4292.

[9] W. Lucht, C. Schaaf, and A. Strahler, "An algorithm for the retrieval of albedo from space using semiempirical BRDF models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 2, pp. 977–998, Mar. 2000. DOI: 10.1109/36.841980.

[10] T. G. Müller, M. Burgdorf, V. Al-Lagoa, S. A. Buehler, and M. Prange, "The moon at thermal infrared wavelengths: A benchmark for asteroid thermal models," *Astronomy &amp Astrophysics*, vol. 650, A38, Jun. 2021. DOI: 10.1051/0004-6361/202039946.

[11] R. Floberghagen, P. Visser, and F. Weischede, "Lunar albedo force modeling and its effect on low lunar orbit and gravity field determination," *Advances in Space Research*, vol. 23, no. 4, pp. 733–738, Jan. 1999. DOI: 10.1016/s0273-1177(99)00155-6.

[12] F. G. Lemoine *et al.*, "Highdegree gravity models from GRAIL primary mission data," *Journal of Geophysical Research: Planets*, vol. 118, no. 8, pp. 1676–1698, Aug. 2013. DOI: 10.1002/jgre.20118.

[13] D. R. Doelling *et al.*, "Advances in geostationary-derived longwave fluxes for the CERES synoptic (SYN1deg) product," *Journal of Atmospheric and Oceanic Technology*, vol. 33, no. 3, pp. 503–521, Mar. 2016. DOI: 10.1175/jtech-d-15-0147.1.

[14] H. H. Kieffer and T. C. Stone, "The spectral irradiance of the moon," *The Astronomical Journal*, vol. 129, no. 6, pp. 2887–2901, Jun. 2005. DOI: 10.1086/430185.

[15] J. Sun and X. Xiong, "Improved lunar irradiance model using multiyear MODIS lunar observations," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 6, pp. 5154–5170, Jun. 2021. DOI: 10.1109/tgrs.2020.3011831.

[16] O. Montenbruck, P. Steigenberger, and U. Hugentobler, "Enhanced solar radiation pressure modeling for galileo satellites," *Journal of Geodesy*, vol. 89, no. 3, pp. 283–297, Nov. 2014. DOI: 10.1007/s00190-014-0774-0.

[17] D. Smith, M. Zuber, F. Lemoine, M. Torrence, and E. Mazarico, "Orbit determination of lro at the moon," in *7^{th} Int. Laser Ranging Service Workshop*, 2008, pp. 13–17.

[18] E. Mazarico, M. T. Zuber, F. G. Lemoine, and D. E. Smith, "Effects of self-shadowing on nonconservative force modeling for mars-orbiting spacecraft," *Journal of Spacecraft and Rockets*, vol. 46, no. 3, pp. 662–669, May 2009. DOI: 10.2514/1.41679.

[19] P. W. Kenneally and H. Schaub, "Fast spacecraft solar radiation pressure modeling by ray tracing on graphics processing unit," *Advances in Space Research*, vol. 65, no. 8, pp. 1951–1964, Apr. 2020. DOI: 10.1016/j.asr.2019.12.028.

[20] K. Vielberg and J. Kusche, “Extended forward and inverse modeling of radiation pressure accelerations for LEO satellites,” *Journal of Geodesy*, vol. 94, no. 4, Mar. 2020. DOI: 10.1007/s00190-020-01368-6.

[21] C. R. Tooley *et al.*, “Lunar reconnaissance orbiter mission and spacecraft design,” *Space Science Reviews*, vol. 150, no. 1-4, pp. 23–62, Jan. 2010. DOI: 10.1007/s11214-009-9624-4.