MECH 6374 Project

Monte Carlo Applications to Astronomical Bodies

Conductive and Radiative Heat Transfer

December 2018

Yajat Pandya, Brian Atwood, David Axe

Abstract:

Radiation in lieu of astronomical bodies has been a subject relying upon a lot of experimental datasets. Our primary motive for the project was to validate few of these data with the help of insight gained from the Radiative topics covered during the MECH 6374 course. The findings not only validate the historical data, but also provide a brief insight on futuristic conditions and how these methods can be possibly used to predict radiation on bodies which are too distant from Earth to measure from.

A series of novel astronomical problems are proposed within the realm of radiative solar exchange. Using the solar system as a medium, astronomical concepts and historical datasets are used with various Monte Carlo simulations in order to gain insight on view factors, heat exchange, and dynamics of moving astronomical bodies. All of these simulations are facilitated with the help of NASA Jet Propulsion Lab's Solar System Dynamics group's HORIZONS system, which allows researchers to gather temporal, spatial, and topographical data about bodies in the solar system [1]. Simulations include the following configurations:

1. Earth's average temperature variation due to its elliptical orbit around the Sun
2. Surface temperature of the Halley's Comet during its relatively elliptical trajectory
3. Temperature variation of an Interstellar Probe performing a "flyby" maneuver near other planet(s)
4. Studying the long term effects of the Sun's growing size and temperature on other planet(s)

Insight into the various radiative exchange phenomena in our solar system was obtained via a robust Monte Carlo method [2].

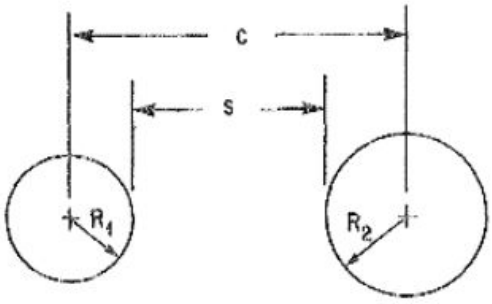
View Factor:

View factor is paramount in determining radiative exchange through various bodies. A simple view factor can be calculated by:

$$F_{i\text{-}j} = \frac{1}{A_i}\int\left(\int \frac{\cos\theta_i \cos\theta_j}{\pi S_{i\text{-}j}^2} dA_j\right) dA_i$$

Where dA_i and dA_j are the differential elements for two surfaces and S_{ij} is the length between these two. θ_i and θ_j give the angles between the connecting line and the vectors to which are normal on each surface.

For the majority of simulations using planetary bodies, the view factor would be for spherical space. The view factor could be seen as:



$$F_{12} = [1 - \sqrt{1 - (R_2/c)^2}][1 - \sqrt{1 - (R_1/c)^2}](c/R_1)^2$$

Where for a sphere of radius R_1 to a large of radius R_2, and given a distance of 'c' between spherical centers. Using the mathematical foundation for view factor, we derived the formula for a general case for 2 spheres as shown above. This was verified in [10].

Temperature calculation:

Earth's temperature must be obtained through a series of relations due to the Sun's radiation. Initially, it is known that the power output from the sun is:

$$P_s = \sigma T_s^4 (4\pi R_s^2)$$

And radiative power received at earth:

$$P_e = \alpha * P_s * Solid\ Angle = \alpha P_s(\pi R_e^2/(4\pi d^2))$$

While radiated power due to earth's temperature

$$P_e = \sigma T_e^4 (4\pi R_e^2)$$

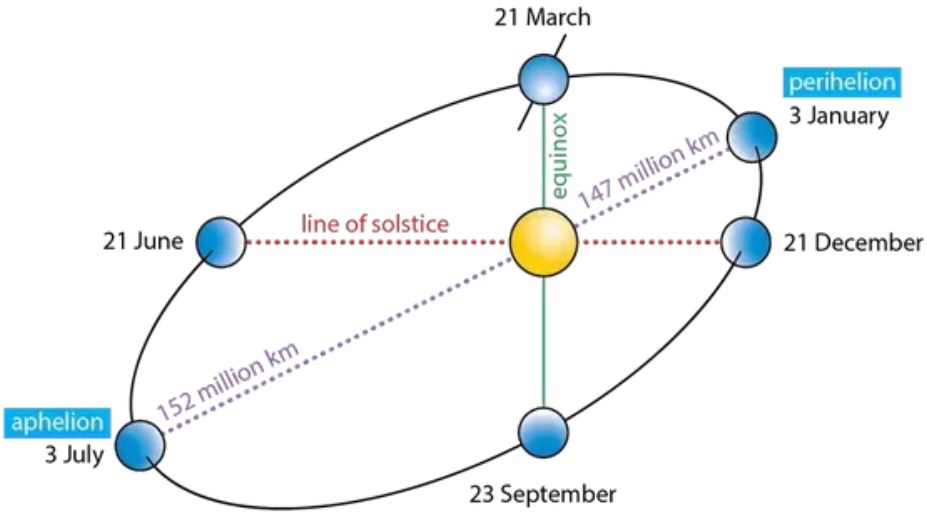
Thus, while understanding that the radiative power received at the earth is equal to the radiated power by earth's own temperature, a relationship can be obtained to determine earth's temperature:

$$T_e = [T_s^4 R_s^2/(4\alpha d^2)]^{.25}$$

Earth:

The orbit of the Earth around the Sun is elliptical, but only just so. Throughout the year, as the earth revolves around the sun, its distance from the sun changes due to the eccentricity of its orbit. Although its eccentricity is low (only 0.0167), its temperature still varies somewhat significantly over the course of the yearly cycle. The focus of our experiments regarding Earth is

to use the Monte-Carlo method to generate view factors that can be used to accurately estimate these temperature variations throughout its orbit.



Given that the distance of a point on the ellipse from one of the foci:

$$r = a(1 - e^2)/(1 + e\cos\theta)$$

Where a is the semi-major axis, a relationship can be formed between the varying distance, d, the maximum distance, a, and the ratio dependent upon the day of the year. This is given by:

$$d = a(1 - e^2)/(1 + e\cos(2\pi * (day - 4)/365))$$

We calculate the View Factor also from our robust Monte-Carlo simulation where we generate a large number of rays from the Sun in all directions. This required us to relate the Monte Carlo View Factor, with the instantaneous temperature of the earth.

$$T_e = [T_s{}^4 R_s{}^2/(4\alpha d^2)]^{.25}$$

Here, we substitute the value of d^2 with the VF:

$$d^2 = (R_s{}^2(VF)^2 - R_E{}^2)^2/(4VF(VF - 1)\,(R_s{}^2(VF) - R_E{}^2))$$

This way, we get the Earth's surface temperature as a function of View Factor obtained by our Monte-Carlo code.

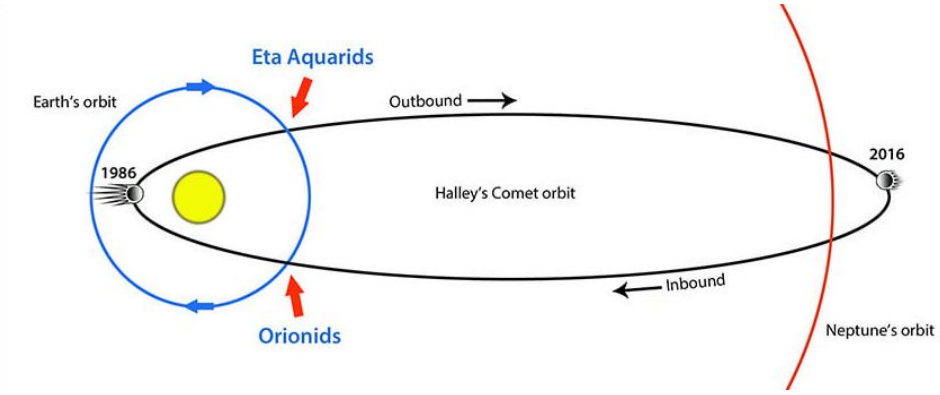
Interstellar probe:

The second experiment involved simulating the sending of a probe on a fly-by of each planet in the solar system to evaluate the effects of planetary radiation on a nearby body. An arbitrary starting point near the planet of interest was chosen, and from that point the probe was sent along a vector path that took it past the planet. Although the original aim of this experiment was to

conduct a fly-by on each planet individually, two things were discovered that made this unfeasible. The first was that a realistically sized probe would have a view factor that was too small for our computational limitations. The second was that flying by each planet would be unnecessary because the overall change in the view factors would follow a similarly shaped curve regardless of the planet of interest. Using this new information, the experiment was changed to make the best of the situation. Instead of eight planetary flybys, one planet, Mercury, was selected for study for its high temperatures. Five flybys were initiated in which the only changing parameter was the radius of the probe of interest. Although significant, effects of the sun were neglected because the aim of this experiment was to determine only the individual contribution of the planet to the temperature variation of the probe.

Halley's comet:

For the third experiment, a simulation was conducted to measure incident radiation on Halley's comet as it travels through its very large elliptical orbit. The trajectory of the comet was coded in a similar manner to the trajectory of a planetary body. The effects of the radiation in one revolution around the sun was analyzed and compared to existing data.



Halley's elliptical path [8].

As with the probe experiment, this experiment was unable to be completed as designed because the limits of available computational power were exceeded by the small view factor of Halley's comet from the Sun. The experiment was modified to simply estimate the temperature of the comet using numerical methods.

Solar Life-cycle:

The final experiment focused on the long-term transient physical properties of our Sun. Over a period of billions of years, the Sun's radius and temperature will vary with time. Data from Ribas [4] explicitly defines these temporal relationships. This model will be used to study the effects of this increasing radiative surface area and temperature on the same probes and comet used in the

first and second simulation. Due to the time required to run accurate Monte-Carlo solutions for the Sun-Earth view factor, the original experiment design was again modified to account for computational limitations. Rather than simulating the temporal effects of the Sun against Earth's real orbit, Earth was statically positioned closer to the Sun to reduce computational time. This provided a data curve that can be scaled to estimate effects on bodies farther away, but implementation of these scaling methods is beyond the scope of this paper.

Results:

For each of the experiments in this report, a simulation was conducted using a novel FORTRAN 90 code on the *Navier* computer cluster, which is hosted on University of Texas at Dallas's *Ganymede* supercomputer. Depending on the requirements of the individual simulations, anywhere between 16 and 96 processors were used for computation. MATLAB was used for post-processing to create the figures available in the proceeding sections. Both the simulation and post-processing codes are available for reference in the Appendix.

Earth Experiment:

The following equation was used to generate an analytical solution for the view factor from the Sun to the Earth.

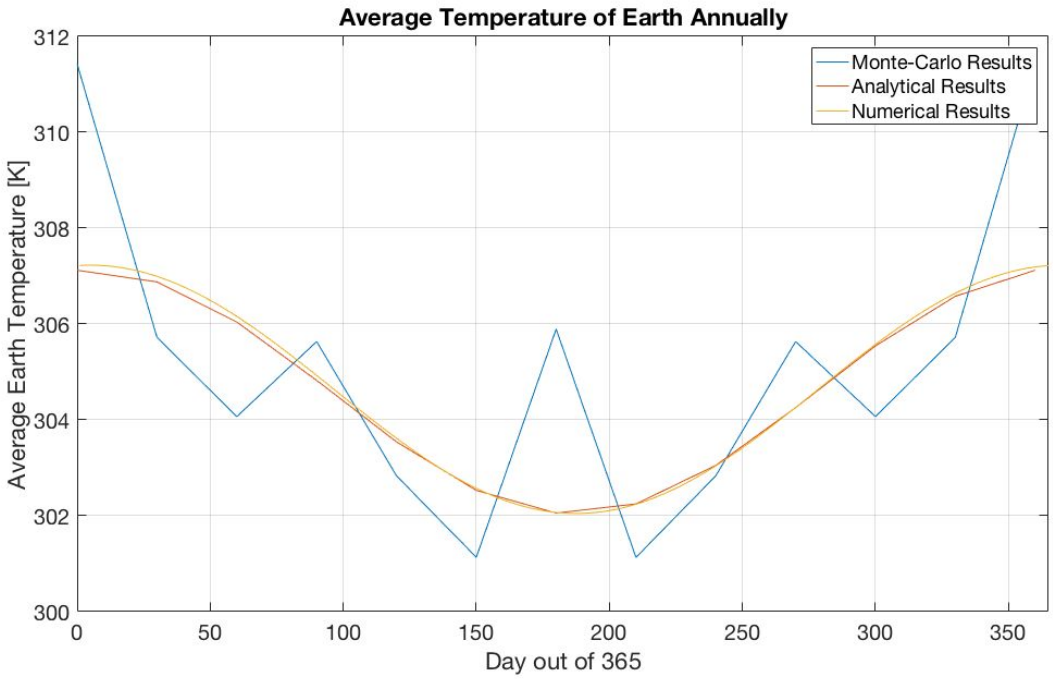
$$F_{12} = [1 - \sqrt{1 - (R_2/c)^2}][1 - \sqrt{1 - (R_1/c)^2}](c/R_1)^2$$

Monte-Carlo simulations were run at 12 regularly spaced intervals along earth's orbit that corresponded to the 4th day of each month. For each simulation, 9.6e11 rays were fired from the Sun, but only on the hemisphere of the Sun that faced Earth. There are three reasons that 9.6e11 rays were selected to be fired. Firstly, the view factor from Sun to Earth was on the order of 1e-10, which meant that a minimum of 1e10 rays would need to be fired to consistently generate even one hit. Secondly, an upper limit on the order of 1e12 rays was determined because each of the 12 months with this number of rays would take approximately 3.3 hours, or 40 hours in total simulation time - even with 96 processors. For each additional order of magnitude of rays to be fired, the time required for the computation rose by a factor of ten. So, to fire 1e13 rays instead of 1e12, it would take approximately 400 hours of computational time. Finally, in order to optimally parallelize the simulation, it is important to evenly distribute the work among the processors. A compromise was reached wherein 9.6e11 was roughly on the order of magnitude of 1e12, while also being evenly divisible by 96.

Numerical results for this experiment were generated from the following equations:

$$d = a\frac{1-e^2}{1+ecos(2\pi\frac{day-4}{365})} \qquad Te = \frac{T_s^4 R_s^2}{4\alpha d^2}^{0.25}$$

Where d is the distance from Sun to Earth at any point in its orbit, e is the eccentricity of the earth's orbit, α is the absorptivity of the earth (1-albedo), Te is the temperature of the Earth, and Ts and Rs are temperature and radius of the sun, respectively. Numerical, analytical, and Monte-Carlo results from this experiment can be seen on the plot below.

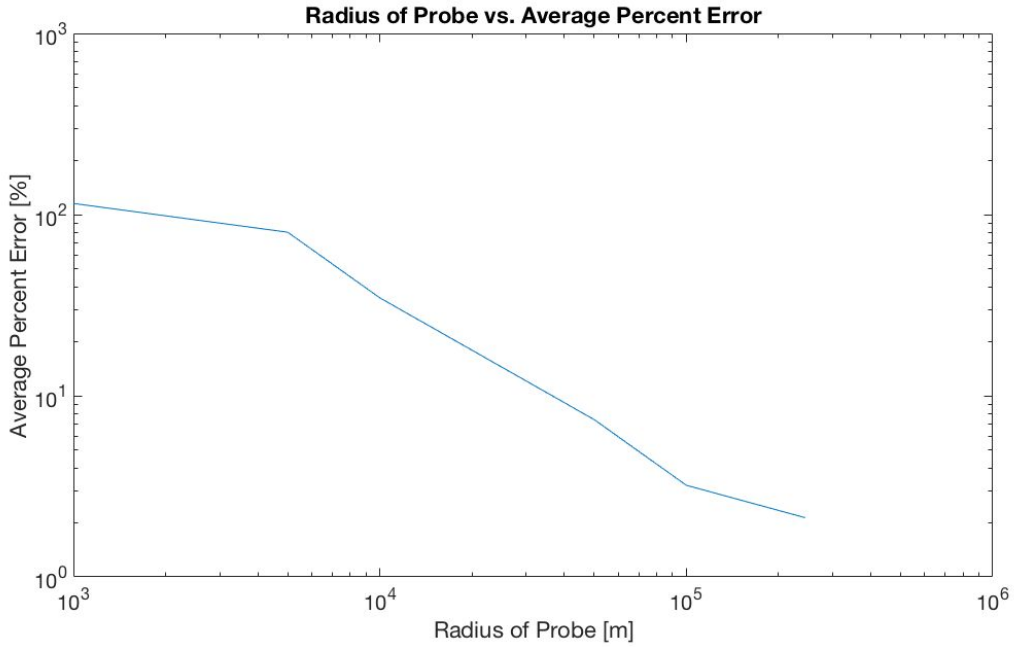


It can be seen that the numerical results and the analytical results (from the analytical view factor calculations) agree with each other nicely. Additionally, the Monte-Carlo results follow these two curves, albeit with a percent error on the order of 10%. This intuitively makes sense because the Sun to Earth view factor is on the order of 1e-10, and roughly 1e12 rays are being fired from the Sun. This gives a theoretical percent error on the order of 10% for any view factor generated with these parameters. The complex thermal exchange mechanisms that occur inside of the atmosphere, such as the ocean currents and the greenhouse effect, were not considered in our simulation. These may explain why the results show temperatures that are slightly higher than real values measured for Earth.

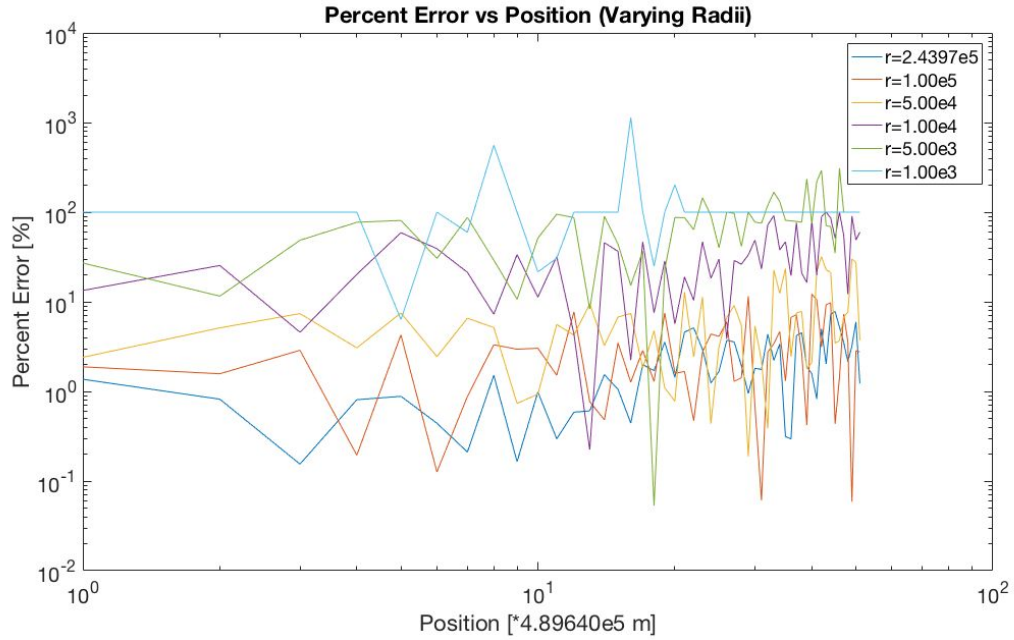
Interstellar Probe Experiment:

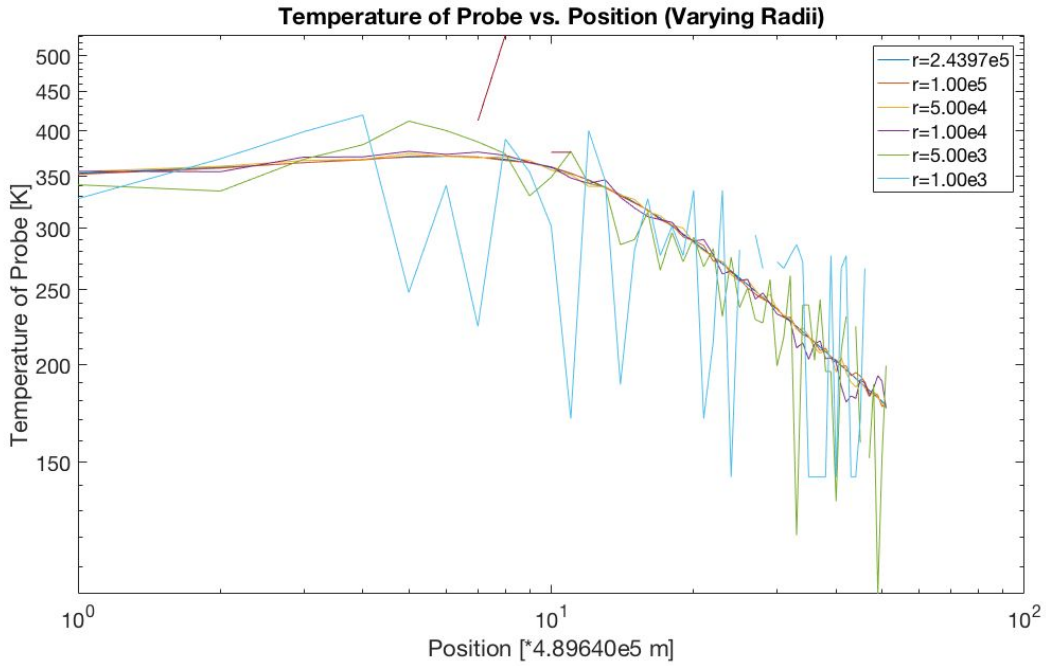
For the probe experiment, the setup consisted of a sphere the size of Mercury (r=2.4397e6 m), isolated in space. A spherical probe was sent on a fixed trajectory of length 2.445e7 m that

passed the planet. Gravity was ignored, so that the trajectory was straight and had one unique minimum distance from Mercury. The trajectory was subdivided into 50 equal length segments (5.18e6 m), and a view factor was generated via a Monte-Carlo simulation at each point. This process of flying by the planet was repeated a total of six times, with each of the six repetitions having a unique radius of 2.4397e5 m, 1.0e5 m, 5.0e4 m, 1.0e4 m, 5.0e3 m, and 1.0e3 m, respectively. The original intention of this experiment was to fly a realistically-sized probe past each of the eight planets of the solar system to measure the planet's individual contributions to a spacecraft's temperature variation. However, it was soon determined that the view factor of a probe of radius 3m was much too small (especially compared to larger planets such as the gas giants) to calculate via Monte-Carlo in the timeframe of this project. Instead, the experiment was changed to show multiple flybys on the same path, so that the reader could see how a decreasing view factor with a static number of simulated rays affected the percent error of a the results. Below, a plot shows the relationship between the percent radius of the probe and percent error. As the radius of the craft decreases, percent error increases. At a radius on the order of one kilometer, the percent error becomes unacceptable.



Below, two plots show the changing view factors as the radius decreases, and the changing temperature as the radius decreases.

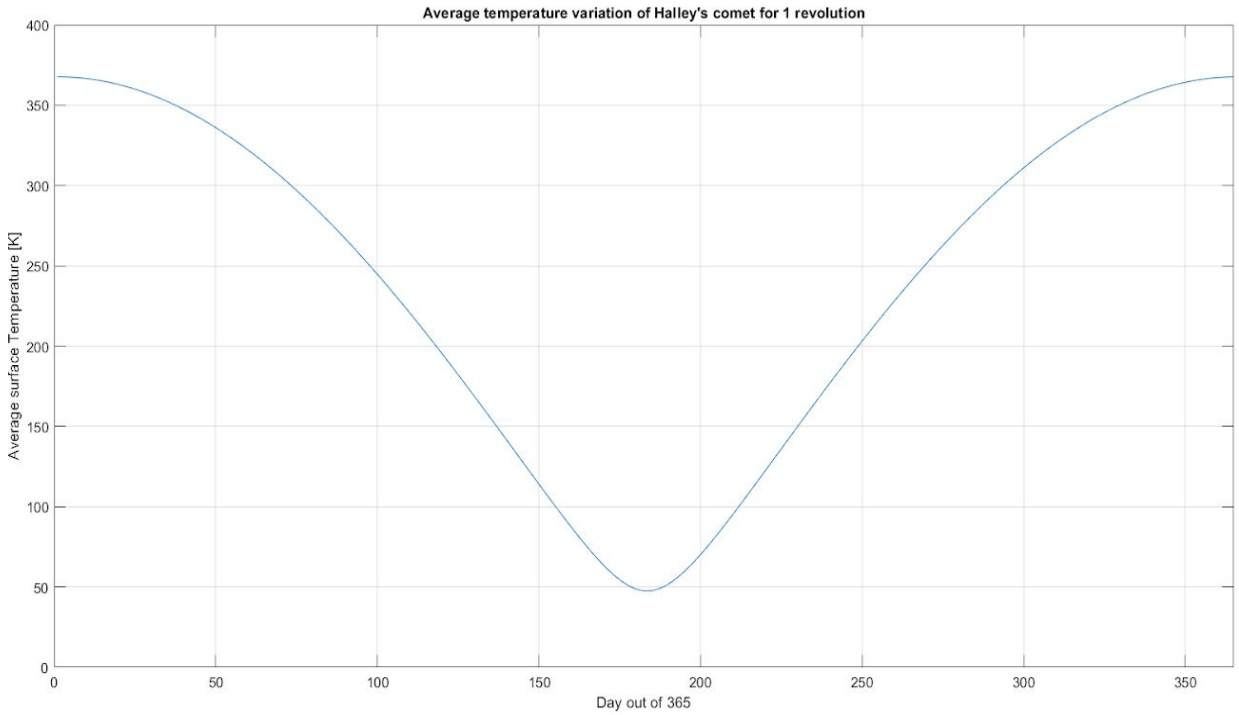




As illustrated by these two plots, the final two radii (r=5.0e3 and r=1.0e3) provide completely unreliable results with the number of rays that were launched. Due to the computational time of this experiment, more rays could not be added within the time schedule of the project. Larger versions of the proceeding plots, as well as individual plots showing finer detail of each probe's simulation, are available for reference in the Appendix.

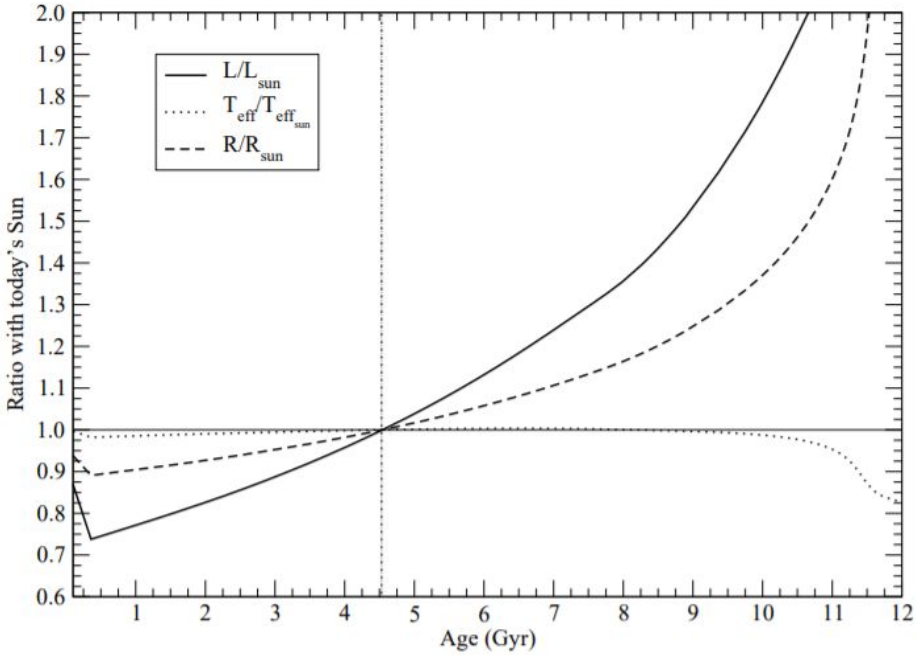
Halley's Comet Experiment:

Due to the high eccentricity (e=0.96714) of Halley's Comet's orbit, the view factor was too small (on the order of 1e-20) to calculate using the Monte-Carlo method due to the limitation of computational power. However, a numerical solution for the comet's curve was computed using methods explained above. The comet's 75 year orbit was scaled down to 365 "comet days", so that it could be accurately compared to the temperature profile of the earth. Using an albedo value of 0.04, the temperature profile of the comet seems to be fairly accurate when compared to measurements taken by the Vega craft that landed on it in 1987. They found that the temperature was 350K when the comet was at its closest point of orbit to the Sun. The numerical results below confirm this.

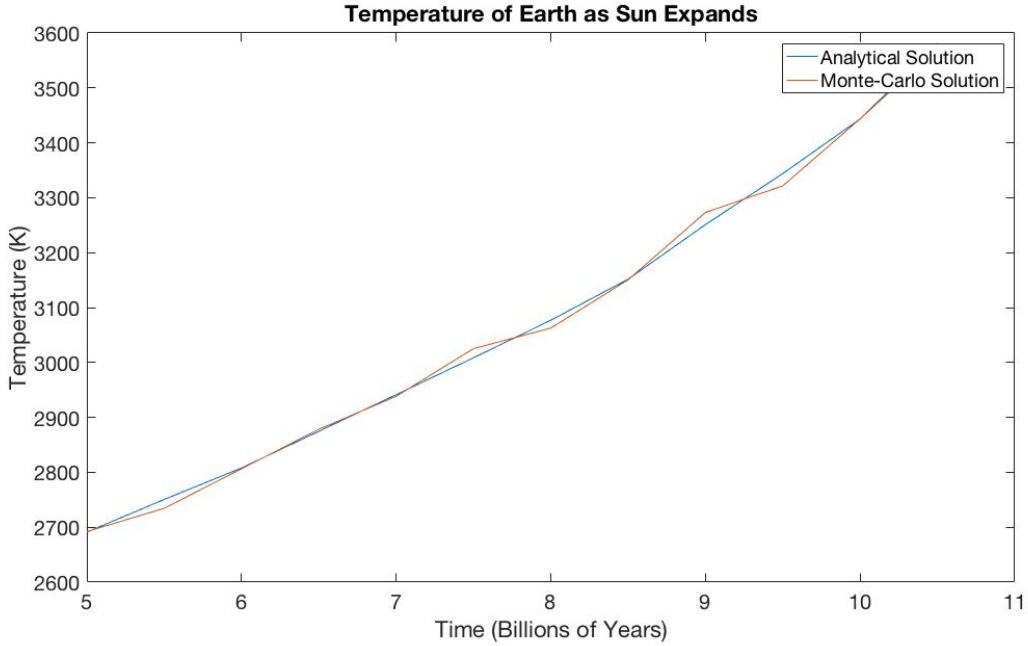


Solar Life Cycle Experiment:

The final experiment conducted in this project was the analysis of the aging Sun's effect on Earth's temperature. As mentioned previously, the experiment was modified such that Earth was modeled as a sphere of radius 6.371e6 m that was positioned at a fixed point located 2.0e9 m from the Sun (Sun's r=6.955e8 m). Based on the plot below, the Sun's size and temperature were interpolated at intervals of 0.5 Billion years, starting with 5.0 Billion years.



Thus, 12 different simulations were run, each with unique values of the Sun's radius and temperature. Below are the results, which show a general trend of exponentially increasing temperature with the increasing radius.



As the time approaches 11 billion years, the Sun's temperature decreases significantly. Interestingly, the temperature of the Earth at this time increases despite this fact. It would seem that the increasing radius of the Sun is outpacing its decreasing temperature, so that it is still outputting more energy - at this particular time. Over time of course, due to the fourth-power

nature of the temperature in the Stefan-Boltzmann law, temperature would outscale the second-power nature of the Sun's increasing area.

Conclusion:

In conclusion, these experiments found that some severe computational limits exist that prohibit the accurate calculation of Monte-Carlo view factors of the solar system bodies due to their low ratio of radius to distance-from-Sun. However, when the Monte-Carlo view factor calculation was within computational limits, good agreement was seen between the theoretical and empirical values and the results of the simulations. Perhaps in the years to come, increases in computing power and available supercomputing resources will allow this method to be used for any body in the solar system.

References:

1. Modest, Michael F. *Radiative heat transfer*. Academic press, 2013.
2. http://webserver.dmt.upm.es/~isidoro/tc3/Radiation%20View%20factors.pdf
3. Ribas, I. (2009). The Sun and stars as the primary energy input in planetary atmospheres. *Proceedings of the International Astronomical Union, 5*(S264), 3-18. doi:10.1017/S174392130999229
4. Nicholas Metropolis & S. Ulam (1949) *The Monte Carlo Method*, Journal of the American Statistical Association, 44:247, 335-341, DOI: 10.1080/01621459.1949.10483310
5. NASA Solar system fact sheet - https://solarsystem.nasa.gov/planets/
6. https://nssdc.gsfc.nasa.gov/planetary/factsheet/cometfact.html
7. JPL-NASA HORIZONS Solar system data - https://ssd.jpl.nasa.gov/?horizons#telnet
8. Earth Albedo - https://earthobservatory.nasa.gov/images/84499/measuring-earths-albedo
9. *Approximate Radiation Shape Factors between Two Spheres*, J. D. Felske, J. Heat Transfer 100(3), 547-548 (Aug 01, 1978) (2 pages) doi:10.1115/1.3450848
10. R. R. Britt (29 November 2001). *"Comet Borrelly Puzzle: Darkest Object in the Solar System"*. Space.com. Retrieved 16 December 2008.
11. https://www.universetoday.com/135322/comet-halley-plays-bit-part-weekend-eta-aquarid-meteor-shower/

APPENDIX:

Video links:
Probe: https://drive.google.com/open?id=1qFVqaW3A1tOoZSfMPgjNlX9lWBFGuqBB
Earth: https://drive.google.com/open?id=17oOdTx7qAa7lALH214TVxEmdw0gLtpjl
Comet: https://drive.google.com/open?id=1Skf2An8HzDX-xOem9kSS2TDLe0AyLmyj

Section A: Simulation code (FORTRAN 90)
Aging Sun Simulation Code:

```
      module global
      integer :: numprocs,ierr,my_node
      integer*8 :: n2do
      real*8,parameter :: r2=6.3781e6 !earth radius
      real*8,parameter :: ecc=0.0167,d=2.0e9
      real*8,dimension(12) :: r1_arr
      real*8,parameter :: c_pi=2.*asin(1.),sig=5.67e-8
      real*8,dimension(3) :: src,norm,t1,t2,u
      end

      program main
      use global
      use mpi
      implicit none

      real*8,dimension(3) :: uy,uz,res
      real*8 :: i,nrays,sf,sf_an,sf_an1,num,st,et,tick,k,r1
      real*8 :: theta,phi,nhits,dist,t_nhits,e,f,g,h,t,E1,ep1
      nhits=0
      t_nhits=0
      nrays = 9600000000
      uy=(/0.,1.,0./)
      uz=(/0.,0.,1./)
      r1_arr(1:12)=(/7.233283e8,7.5939307e8,7.859240e8,8.220905e8,
     &           8.624299e8,9.0141604e8,9.4939684e8,9.980540e8,
     &           1.064127e9,1.144111e9,1.238004e9,1.352763e9/)

      call random_seed()

c     Parallel initialization
      call MPI_INIT(ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,my_node,ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs,ierr)
      st=MPI_Wtime()
      n2do=nrays/numprocs
      do k=1,12
        r1=r1_arr(k)
        nhits=0
        t_nhits=0
      do i=((my_node*n2do)+1),(n2do*(my_node+1))

c     Generate random source on surface
        call random_number(num)
        phi=2*c_pi*num
        call random_number(num)
        theta=acos(1-num)
        src=(/r1*sin(theta)*cos(phi),r1*sin(theta)*sin(phi),
     &        d-(r1*cos(theta))/)
c     Generate normal from surface at src
        norm=(/(src(1)/r1),(src(2)/r1),((src(3)-d)/r1)/)

c     Creating vectors tangent to surface
        call cross_product(norm,uz,t1)
        t1=t1/NORM2(t1)
        call cross_product(norm,t1,t2)
        t2=t2/NORM2(t2)
```

```
c       Creating random direction vector from surface
          call random_number(num)
          theta=asin(sqrt(num))
          call random_number(num)
          phi=2*c_pi*num
          u=(cos(theta)*norm)+sin(theta)*((cos(phi)*t1)+(sin(phi)*t2))

c       Checking for intersection with sphere
          call cross_product(src,u,res)
          dist=NORM2(res)/NORM2(u)

c       Checking for hit
          if(dist.LE.r2)then
            if(u(3).LT.0.)then
              t_nhits=t_nhits+1
            endif
          endif
        enddo !end nrays loop

        call MPI_ALLREDUCE(t_nhits,nhits,1,MPI_REAL8,MPI_SUM,
     &                     MPI_COMM_WORLD,ierr)

        if(my_node.eq.0)then
          sf=nhits/(nrays*2)
          sf_an=(1-sqrt(1-(r2/d)**2))*(1-sqrt(1-(r1/d)**2))/
     &          ((r1/d)**2)
          sf_an1=(1-sqrt(1-(r2/d)**2))*0.5
          write(6,*)'d value: ',k
          write(6,*)'nhits: ',nhits
          write(6,*)'d: ',d
          write(6,*)'View Factor: ',sf
          write(6,*)'Analytical V.F. ',sf_an
          write(6,*)'Percent Error: ',(sf-sf_an)*100/sf_an
          write(6,*)'sf_an1:',sf_an1
          write(6,*)'***************************'
        endif
        enddo !end dist loop

        et=MPI_Wtime()
        if(my_node.eq.0)then
          write(6,*)'Time Elapsed: ',(et-st),'seconds'
          e=floor((et-st)/3600)
          f=(((et-st)/3600)-e)*60
          g=floor(f)
          h=(f-g)*60
          write(6,*)int(e),'H',int(g),'M',int(h),'S'
        endif
        call MPI_FINALIZE(ierr)
        end program

c ======================================================================
c ======================================================================

        subroutine cross_product(a,b,c)
        real*8,dimension(3),intent(in) :: a,b
        real*8,dimension(3),intent(out) :: c

        c(1)=(a(2)*b(3))-(a(3)*b(2))
        c(2)=(a(3)*b(1))-(a(1)*b(3))
        c(3)=(a(1)*b(2))-(a(2)*b(1))

        return
        end subroutine

c ======================================================================
c ======================================================================
```

Earth Orbit Simulation Code:

```
      module global
      integer :: numprocs,ierr,my_node
      integer*8 :: n2do
      real*8,parameter :: r1=6.957e8,r2=6.3781e6,au=149597870700
      real*8,parameter :: ecc=0.0167
      real*8,parameter :: c_pi=2.*asin(1.),sig=5.67e-8
      real*8,dimension(3) :: src,norm,t1,t2,u
      end

      program main
      use global
      use mpi
      implicit none

      real*8,dimension(3) :: uy,uz,res
      real*8,dimension(4) :: d_arr
      real*8 :: i,nrays,sf,sf_an,sf_an1,num,st,et,tick,k
      real*8 :: theta,phi,nhits,dist,t_nhits,e,f,g,h,t,d,E1,ep1
      nhits=0
      t_nhits=0
      nrays = 960000000000
      uy=(/0.,1.,0./)
      uz=(/0.,0.,1./)
      E1=sig*(5777**4)*4*c_pi*(r1**2)
      ep1=E1/nrays
      call random_seed()

c     Parallel initialization
      call MPI_INIT(ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,my_node,ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs,ierr)
      st=MPI_Wtime()
      n2do=nrays/numprocs
      do k=0,11
        nhits=0
        t_nhits=0
        t=30*k
        d=au*(((1-(ecc**2)))/(1+(ecc*cos(2*c_pi*((t)-4)/365))))
      do i=((my_node*n2do)+1),(n2do*(my_node+1))

c     Generate random source on surface
        call random_number(num)
        phi=2*c_pi*num
        call random_number(num)
        theta=acos(1-num)
        src=(/r1*sin(theta)*cos(phi),r1*sin(theta)*sin(phi),
     &        d-(r1*cos(theta))/)
c     Generate normal from surface at src
        norm=(/(src(1)/r1),(src(2)/r1),((src(3)-d)/r1)/)

c     Creating vectors tangent to surface
        call cross_product(norm,uz,t1)
        t1=t1/NORM2(t1)
        call cross_product(norm,t1,t2)
        t2=t2/NORM2(t2)
```

```
c       Creating random direction vector from surface
          call random_number(num)
          theta=asin(sqrt(num))
          call random_number(num)
          phi=2*c_pi*num
          u=(cos(theta)*norm)+sin(theta)*((cos(phi)*t1)+(sin(phi)*t2))

c       Checking for intersection with sphere
          call cross_product(src,u,res)
          dist=NORM2(res)/NORM2(u)

c       Checking for hit
          if(dist.LE.r2)then
            if(u(3).LT.0.)then
              t_nhits=t_nhits+1
            endif
          endif
        enddo !end nrays loop

        call MPI_ALLREDUCE(t_nhits,nhits,1,MPI_REAL8,MPI_SUM,
     &                     MPI_COMM_WORLD,ierr)

        if(my_node.eq.0)then
          sf=nhits/(nrays*2)
          sf_an=(1-sqrt(1-(r2/d)**2))*(1-sqrt(1-(r1/d)**2))/((r1/d)**2)
          sf_an1=(1-sqrt(1-(r2/d)**2))*0.5
          write(6,*)'Month: ',k+1
          write(6,*)'nhits: ',nhits
          write(6,*)'d: ',d
          write(6,*)'View Factor: ',sf
          write(6,*)'Analytical V.F. ',sf_an
          write(6,*)'Percent Error: ',(sf-sf_an)*100/sf_an
          write(6,*)'sf_an1:',sf_an1
          write(6,*)'**************************'
        endif
        enddo !end month loop

        et=MPI_Wtime()
        if(my_node.eq.0)then
c          sf=nhits/(nrays*2)
c          sf_an=(1-sqrt(1-(r2/d)**2))*(1-sqrt(1-(r1/d)**2))/((r1/d)**2)
c          sf_an1=(1-sqrt(1-(r2/d)**2))*0.5
c          write(6,*)'nhits: ',nhits
c          write(6,*)'View Factor: ',sf
c          write(6,*)'Analytical V.F. ',sf_an
c          write(6,*)'Percent Error: ',(sf-sf_an)*100/sf_an
c          write(6,*)'sf_an1:',sf_an1
          write(6,*)'Time Elapsed: ',(et-st),'seconds'
          e=floor((et-st)/3600)
          f=(((et-st)/3600)-e)*60
          g=floor(f)
          h=(f-g)*60
          write(6,*)int(e),'H',int(g),'M',int(h),'S'
        endif
        call MPI_FINALIZE(ierr)
        end program
```

```
c ======================================================================
c ======================================================================

      subroutine cross_product(a,b,c)
      real*8,dimension(3),intent(in) :: a,b
      real*8,dimension(3),intent(out) :: c

      c(1)=(a(2)*b(3))-(a(3)*b(2))
      c(2)=(a(3)*b(1))-(a(1)*b(3))
      c(3)=(a(1)*b(2))-(a(2)*b(1))

      return
      end subroutine

c ======================================================================
c ======================================================================
```

Probe Flyby Simulation Code:

```
      module global
      integer :: numprocs,ierr,my_node
      integer*8 :: n2do
      real*8,parameter :: r1=2.4397e6,r2=1.00e3!,d=5.18367e6
      real*8,parameter :: c_pi=2.*asin(1.),l_len=2.4482e7
      real*8,parameter :: d_m=5.18367e6,x_dm=2.4397e6
      real*8,dimension(3) :: src,norm,t1,t2,u
      end


      program main
      use global
      use mpi
      implicit none

      real*8,dimension(3) :: uy,uz,res
      real*8 :: i,nrays,sf,sf_an,sf_an1,num,j,dx,x1,x2
      real*8 :: theta,phi,nhits,dist,t_nhits,d

      nhits=0
      t_nhits=0
      nrays = 96000000
      uy=(/0.,1.,0./)
      uz=(/0.,0.,1./)
      call random_seed()

c     Parallel initialization
      call MPI_INIT(ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,my_node,ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs,ierr)
      n2do=nrays/numprocs

      do j=0,50
        t_nhits=0
        nhits=0
        dx=l_len/50
        x1=j*dx
        x2=abs(x1-x_dm)
        d=sqrt((d_m**2)+(x2**2))
      do i=((my_node*n2do)+1),(n2do*(my_node+1))
```

```
c       Generate random source on surface
          call random_number(num)
          phi=2*c_pi*num
          call random_number(num)
          theta=acos(1-num)
          src=(/r1*sin(theta)*cos(phi),r1*sin(theta)*sin(phi),
     &          d-(r1*cos(theta))/)
c       Generate normal from surface at src
          norm=(/(src(1)/r1),(src(2)/r1),((src(3)-d)/r1)/)

c       Creating vectors tangent to surface
          call cross_product(norm,uz,t1)
          t1=t1/NORM2(t1)
          call cross_product(norm,t1,t2)
          t2=t2/NORM2(t2)

c       Creating random direction vector from surface
          call random_number(num)
          theta=asin(sqrt(num))
          call random_number(num)
          phi=2*c_pi*num
          u=(cos(theta)*norm)+sin(theta)*((cos(phi)*t1)+(sin(phi)*t2))

c       Checking for intersection with sphere
          call cross_product(src,u,res)
          dist=NORM2(res)/NORM2(u)

c       Checking for hit
          if(dist.LE.r2)then
            if(u(3).LT.0.)then
              t_nhits=t_nhits+1
            endif
          endif
        enddo   !end i loop

       call MPI_ALLREDUCE(t_nhits,nhits,1,MPI_REAL8,MPI_SUM,
     &                    MPI_COMM_WORLD,ierr)
        if(my_node.eq.0)then
          sf=nhits/(nrays*2)
          sf_an=(1-sqrt(1-(r2/d)**2))*(1-sqrt(1-(r1/d)**2))/((r1/d)**2)
          sf_an1=(1-sqrt(1-(r2/d)**2))*0.5
          write(6,*)'*********',j,'*********'
          write(6,*)'nhits: ',nhits
          write(6,*)'View Factor: ',sf
          write(6,*)'Analytical V.F. ',sf_an
          write(6,*)'Percent Error: ',(sf-sf_an)*100/sf_an
          write(6,*)'sf_an1:',sf_an1
        endif
        enddo   !end j loop
        call MPI_FINALIZE(ierr)
        end program
```

Section B: Post-processing code (MATLAB)

Probe Flyby Post-processing Code:

```
clear all
clc

r_merc=2.4397e6;
r_2A=2.4397e5;r_2B=1.0e5;r_2C=5.0e4;r_2D=1.0e4;r_2E=5.0e3;r_2F=1.0e3;
len=2.4482e7;
dx=len/50;
for l=1:51
    d(l)=dx*(l-1);
end
Ts = 700; % K
A=load('2_4397e5.txt');
B=load('1_00e5.txt');
C=load('5_00e4.txt');
D=load('1_00e4.txt');
E=load('5_00e3.txt');
F=load('1_00e3.txt');

for j=1:51
    A1(j,1)=A(j,1); %sf_an
    A2(j,1)=A(j,2); %sf
    B1(j,1)=B(j,1); %sf_an
    B2(j,1)=B(j,2); %sf
    C1(j,1)=C(j,1); %sf_an
    C2(j,1)=C(j,2); %sf
    D1(j,1)=D(j,1); %sf_an
    D2(j,1)=D(j,2); %sf
    E1(j,1)=E(j,1); %sf_an
    E2(j,1)=E(j,2); %sf
    F1(j,1)=F(j,1); %sf_an
    F2(j,1)=F(j,2); %sf
end

% Percent Error Calcs
TPE1=0;TPE2=0;TPE3=0;TPE4=0;TPE5=0;TPE6=0;
for k=1:51
    PE1(k)=(abs(A2(k)-A1(k))/A1(k))*100;
    TPE1=TPE1+PE1(k);
    PE2(k)=(abs(B2(k)-B1(k))/B1(k))*100;
    TPE2=TPE2+PE2(k);
    PE3(k)=(abs(C2(k)-C1(k))/C1(k))*100;
    TPE3=TPE3+PE3(k);
    PE4(k)=(abs(D2(k)-D1(k))/D1(k))*100;
    TPE4=TPE4+PE4(k);
    PE5(k)=(abs(E2(k)-E1(k))/E1(k))*100;
    TPE5=TPE5+PE5(k);
    PE6(k)=(abs(F2(k)-F1(k))/F1(k))*100;
    TPE6=TPE6+PE6(k);
end

```

```
50
51 -  for n=1:51
52 -      bA1=(((r_merc^2)*(A1(n)^2)-r_2A^2)^2)/(4*A1(n)*(A1(n)-1)*((A1(n)*r_merc^2)-r_2A^2));
53 -      bA2=(((r_merc^2)*(A2(n)^2)-r_2A^2)^2)/(4*A2(n)*(A2(n)-1)*((A2(n)*r_merc^2)-r_2A^2));
54 -      bB1=(((r_merc^2)*(B1(n)^2)-r_2B^2)^2)/(4*B1(n)*(B1(n)-1)*((B1(n)*r_merc^2)-r_2B^2));
55 -      bB2=(((r_merc^2)*(B2(n)^2)-r_2B^2)^2)/(4*B2(n)*(B2(n)-1)*((B2(n)*r_merc^2)-r_2B^2));
56 -      bC1=(((r_merc^2)*(C1(n)^2)-r_2C^2)^2)/(4*C1(n)*(C1(n)-1)*((C1(n)*r_merc^2)-r_2C^2));
57 -      bC2=(((r_merc^2)*(C2(n)^2)-r_2C^2)^2)/(4*C2(n)*(C2(n)-1)*((C2(n)*r_merc^2)-r_2C^2));
58 -      bD1=(((r_merc^2)*(D1(n)^2)-r_2D^2)^2)/(4*D1(n)*(D1(n)-1)*((D1(n)*r_merc^2)-r_2D^2));
59 -      bD2=(((r_merc^2)*(D2(n)^2)-r_2D^2)^2)/(4*D2(n)*(D2(n)-1)*((D2(n)*r_merc^2)-r_2D^2));
60 -      bE1=(((r_merc^2)*(E1(n)^2)-r_2E^2)^2)/(4*E1(n)*(E1(n)-1)*((E1(n)*r_merc^2)-r_2E^2));
61 -      bE2=(((r_merc^2)*(E2(n)^2)-r_2E^2)^2)/(4*E2(n)*(E2(n)-1)*((E2(n)*r_merc^2)-r_2E^2));
62 -      bF1=(((r_merc^2)*(F1(n)^2)-r_2F^2)^2)/(4*F1(n)*(F1(n)-1)*((F1(n)*r_merc^2)-r_2F^2));
63 -      bF2=(((r_merc^2)*(F2(n)^2)-r_2F^2)^2)/(4*F2(n)*(F2(n)-1)*((F2(n)*r_merc^2)-r_2F^2));
64
65 -      TP1(n)=((Ts^4)*(r_merc^2)/(bA1*2.8))^(0.25);
66 -      TPA(n)=((Ts^4)*(r_merc^2)/(bA2*2.8))^(0.25);
67 -      TPB(n)=((Ts^4)*(r_merc^2)/(bB2*2.8))^(0.25);
68 -      TPC(n)=((Ts^4)*(r_merc^2)/(bC2*2.8))^(0.25);
69 -      TPD(n)=((Ts^4)*(r_merc^2)/(bD2*2.8))^(0.25);
70 -      TPE(n)=((Ts^4)*(r_merc^2)/(bE2*2.8))^(0.25);
71 -      TPF(n)=((Ts^4)*(r_merc^2)/(bF2*2.8))^(0.25);
72 -  end
73
74 -  APE1=TPE1/55;
75 -  APE2=TPE2/55;
76 -  APE3=TPE3/55;
77 -  APE4=TPE4/55;
78 -  APE5=TPE5/55;
79 -  APE6=TPE6/55;
80 -  APE=[APE6,APE5,APE4,APE3,APE2,APE1];
81 -  RAD=[1.00e3,5.00e3,1.00e4,5.00e4,1.00e5,2.4397e5];
```

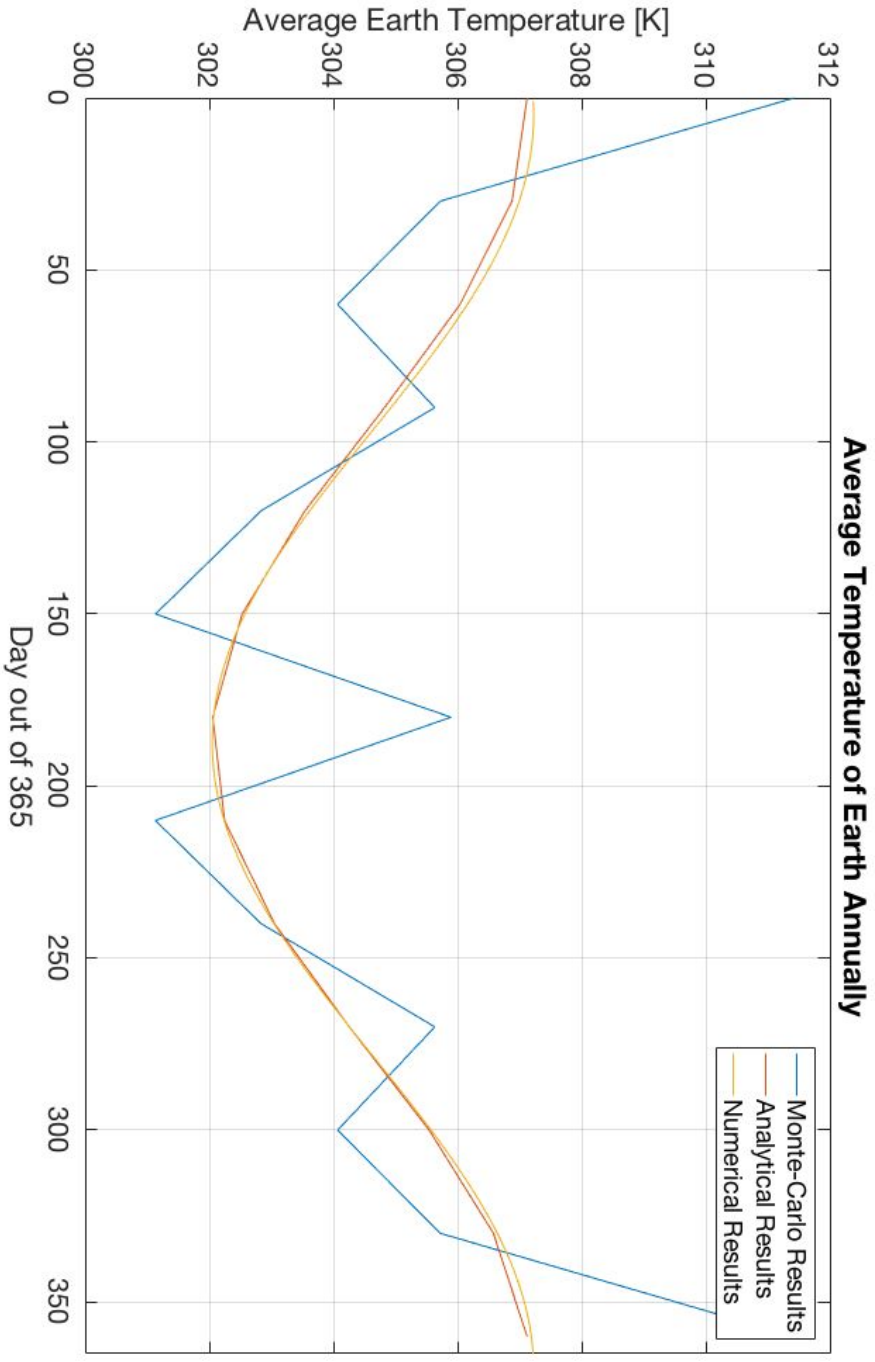
Earth Orbit Post-processing Code:

```
clc
clear all

sigma=5.67*10^-8;Ts=5778;rs=695700000;
% dse=147597870000;
% sc=sigma*(Ts^4)*rs*rs/(dse*dse);
% Te=(0.7*sc/(4*sigma))^0.25;

au=149597870700;
es=0.017;
for t=0:1:365
d(t+1)=au*(1-es*es)/(1+es*cos(2*pi*(t-4)/365));
sc(t+1)=sigma*(Ts^4)*rs*rs/(d(t+1)*d(t+1));
Te(t+1)=(sc(t+1)/(0.7*4*sigma))^0.25;
end

Ts=5777; % K
r_2=6.3781e6;rs=695700000;
d=149597870700;


VF_arr=[4.968750000000000E-010,4.614583333333333E-010,4.515600000E-10,4.609300000E-10,4.442700000E-10
    4.343700000E-10,4.625000000E-10,4.343700000E-10,4.442700000E-10,4.609300000E-10,4.515600000E-10
    4.614583333333333E-010,4.968750000000000E-010];es=0.0167;
VF_arr1=[4.699680832774140E-010,4.684850863131192E-010,4.633924923458846E-010,4.560627752688937E-010
    4.484364047044858E-010,4.424877668856961E-010,4.397245591079688E-010,4.408372917938752E-010
    4.455486839035201E-010,4.526732832402209E-010,4.603844210260546E-010,4.666598038388410E-010
    4.699680832774140E-010];
for i=1:13
    x(i)=((i-1)*30);
    dd(i)=d*(1-es*es)/(1+es*cos(2*pi*(30*i-4)/365));
    xx=(rs/dd(i))^2;
    k=1-sqrt(1-(r_2/dd(i))^2);
    xxx=(((rs^2)*(VF_arr1(i)^2)-r_2^2)^2)/(4*VF_arr1(i)*(VF_arr1(i)-1)*((VF_arr1(i)*rs^2)-r_2^2));
    xxx1=(((rs^2)*(VF_arr(i)^2)-r_2^2)^2)/(4*VF_arr(i)*(VF_arr(i)-1)*((VF_arr(i)*rs^2)-r_2^2));
    Te2(i)=((Ts^4)*(rs^2)/(xxx*2.8))^0.25;
    Te1(i)=((Ts^4)*(rs^2)/(xxx1*2.8))^0.25;
    Tee(i)=((Ts^4)*xx/2.8)^0.25;
end
```

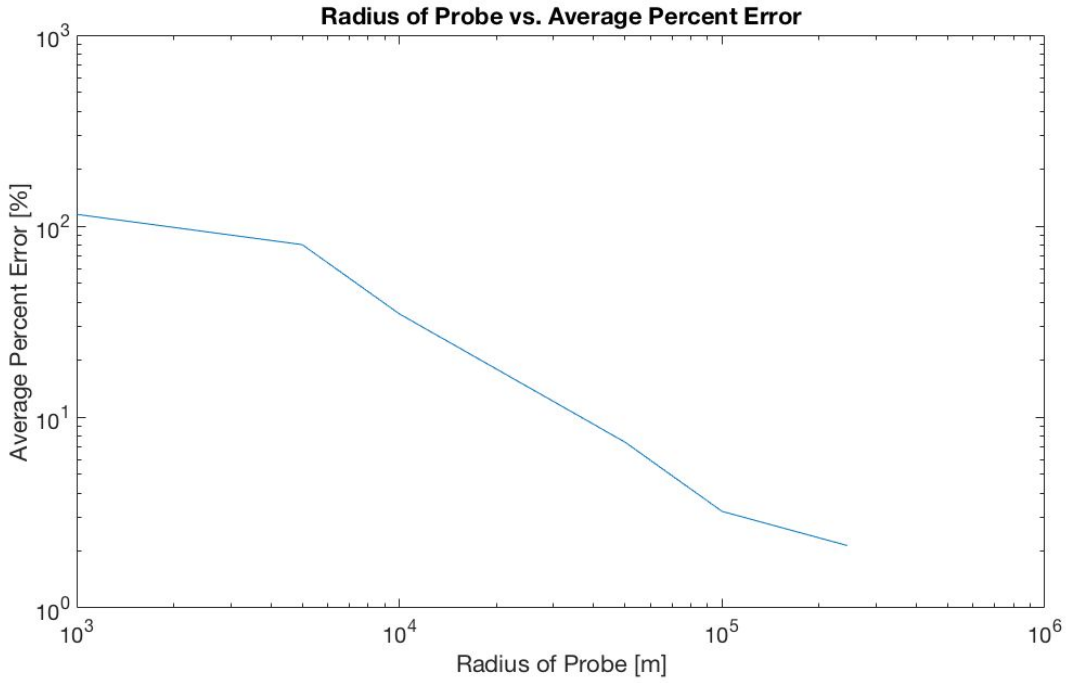
Expanding Sun Post-processing Code:

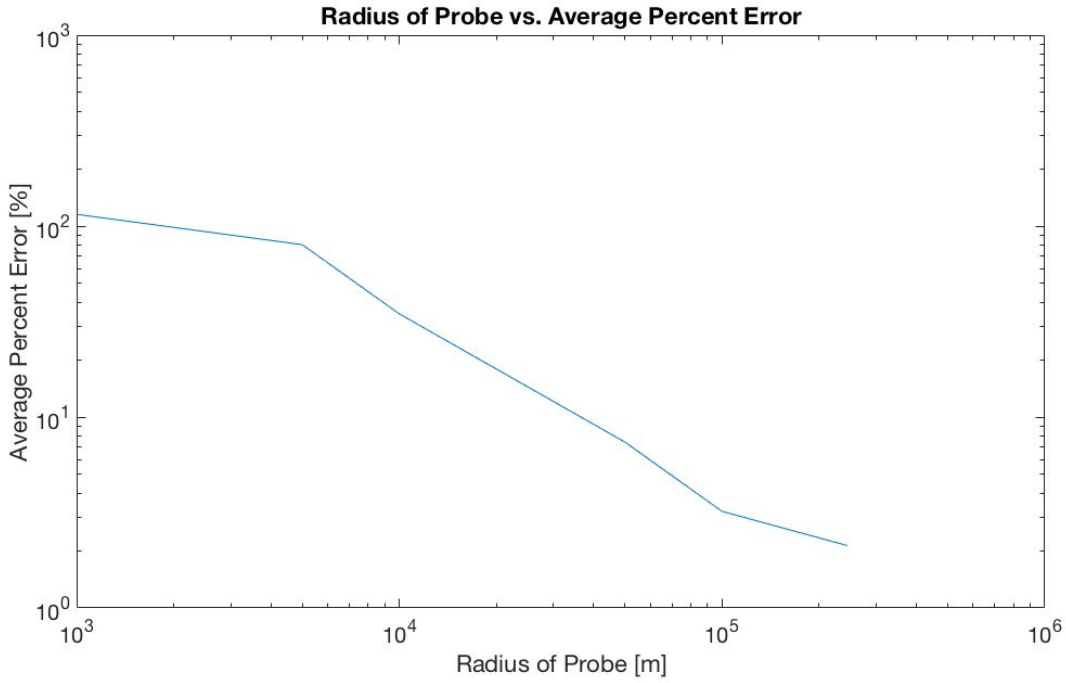
```
clear all
clc

r_2=6.3781e6;
d_arr=[2e9,5e9,1e10,5e10,5.791e10];
Title_arr=['Distance 1e9','Distance 5e9','Distance 1e10','Distance 5e10','Distance 5.791e10'];
T_arr=[5788.55,5794.33,5794.33,5800.10,5794.33,5788.55,5777,5771.223,5765.44,5719.23,5661.46,5632.575];
r_arr=[7.233283e8,7.539307e8,7.859240e8,8.220905e8,8.624299e8,9.041604e8,9.493684e8,9.980540e8
    1.064127e9,1.14411e9,1.238004e9,1.352763e9];
VF_arr=[2.631584457802976E-006,2.641423078702776E-006,2.649069000640606E-006,2.660071737412217E-006
    2.673168588474969E-006,2.686696064220784E-006,2.704591066597232E-006,2.724241845136661E-006
    2.753571568920506E-006,2.793644223695799E-006,2.848136576787190E-006,2.928236203086338E-006];
VF_arr1=[2.635625000000000E-006,2.577760416666667E-006,2.643125000000000E-006,2.674427083333333E-006
    2.661354166666667E-006,2.750572916666667E-006,2.651406250000000E-006,2.720104166666667E-006
    2.836250000000000E-006,2.711562500000000E-006,2.848802083333333E-006,2.891718750000000E-006];

for i=1:12
    rs=r_arr(i);
    Ts=T_arr(i);
    x(i)=(i/2)+4.5;
    b(i)= (((rs^2)*(VF_arr(i)^2)-r_2^2)^2)/(4*VF_arr(i)*(VF_arr(i)-1)*((VF_arr(i)*rs^2)-r_2^2));
    b1(i)=(((rs^2)*(VF_arr1(i)^2)-r_2^2)^2)/(4*VF_arr1(i)*(VF_arr1(i)-1)*((VF_arr1(i)*rs^2)-r_2^2));
    Te(i)=((Ts^4)*(rs^2)/(b(i)*2.8))^0.25;
    Te1(i)=((Ts^4)*(rs^2)/(b1(i)*2.8))^0.25;
end
```

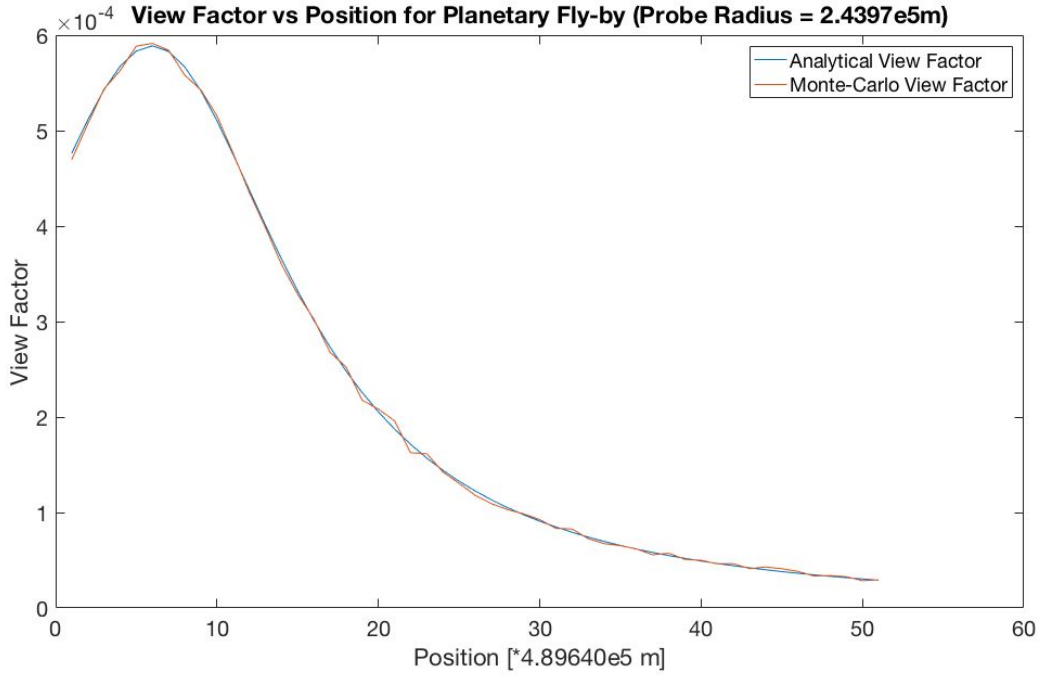
Section C: Complete List of Figures
Earth Orbit Experimental Results:

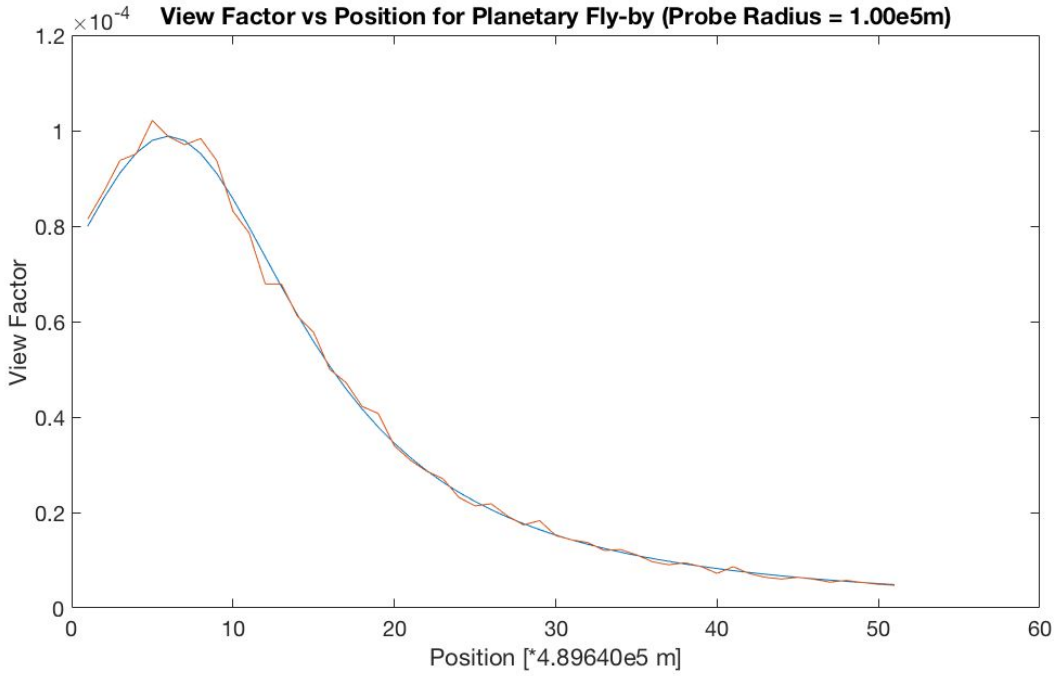


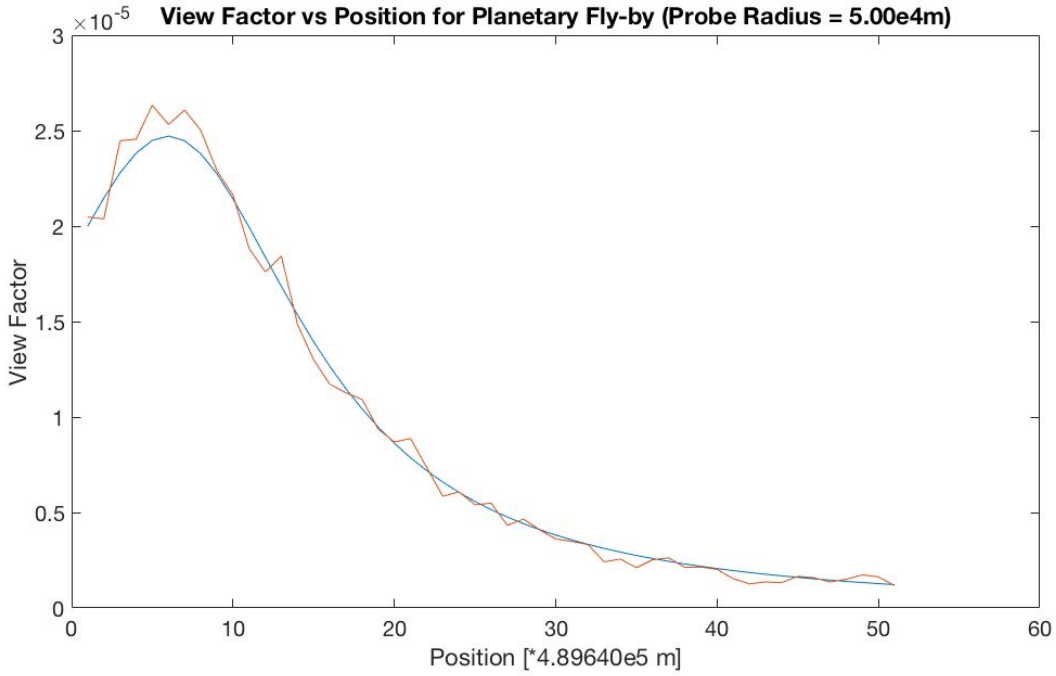
Probe Flyby Experimental Results:

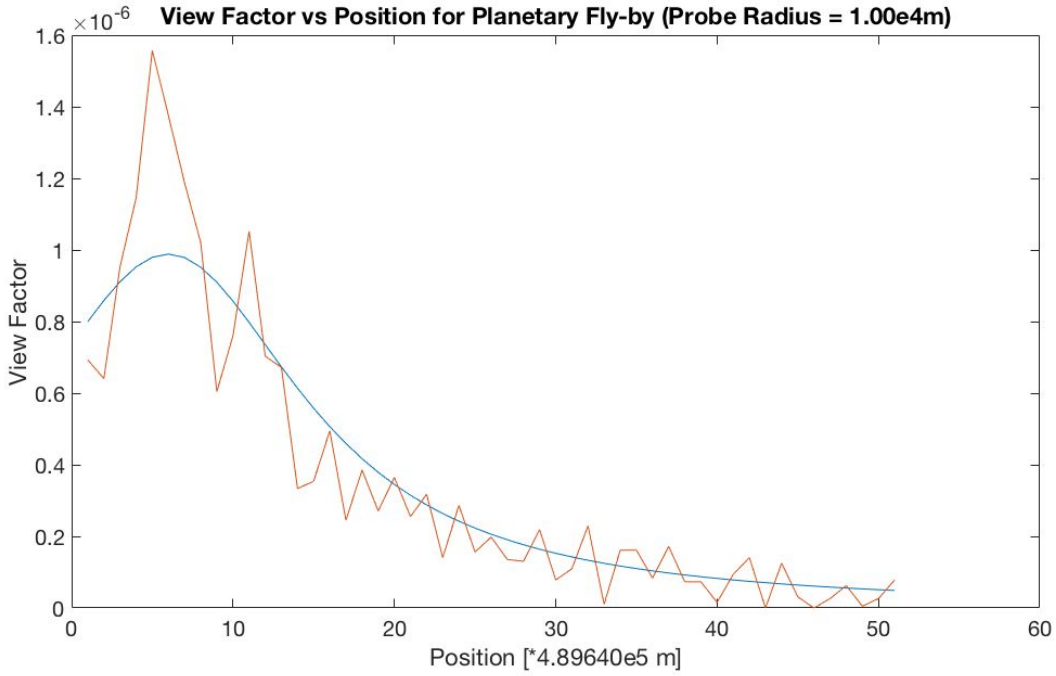


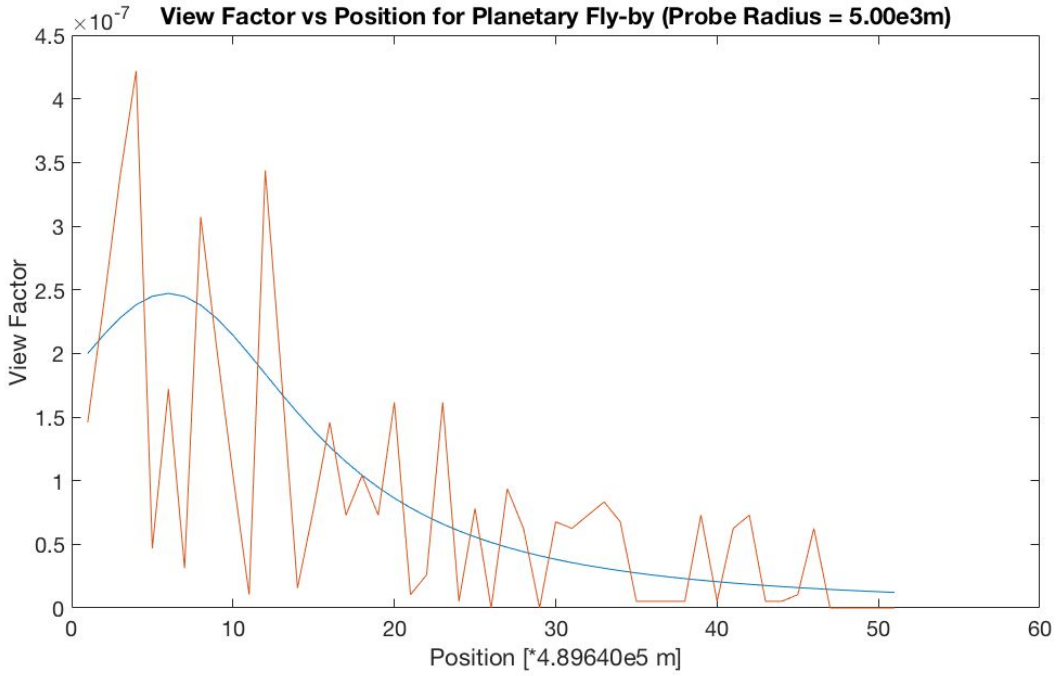


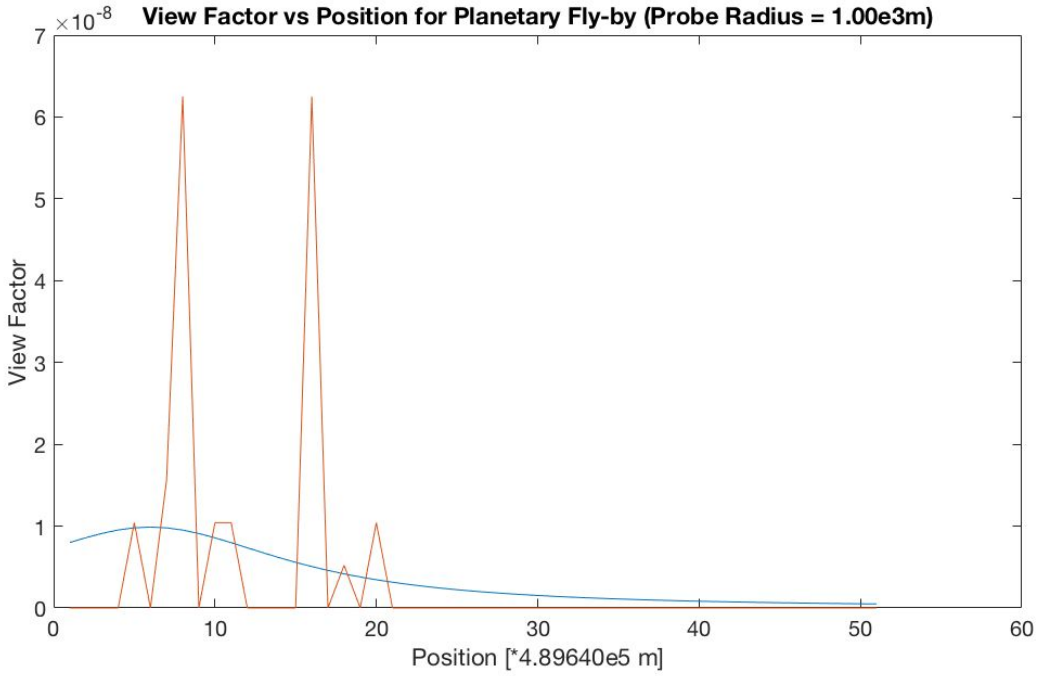

View Factor vs Position for Planetary Fly-by (Probe Radius = 2.4397e5m)
×10^-4
Analytical View Factor
Monte-Carlo View Factor
View Factor
Position [*4.89640e5 m]
0
1
2
3
4
5
6
0
10
20
30
40
50
60

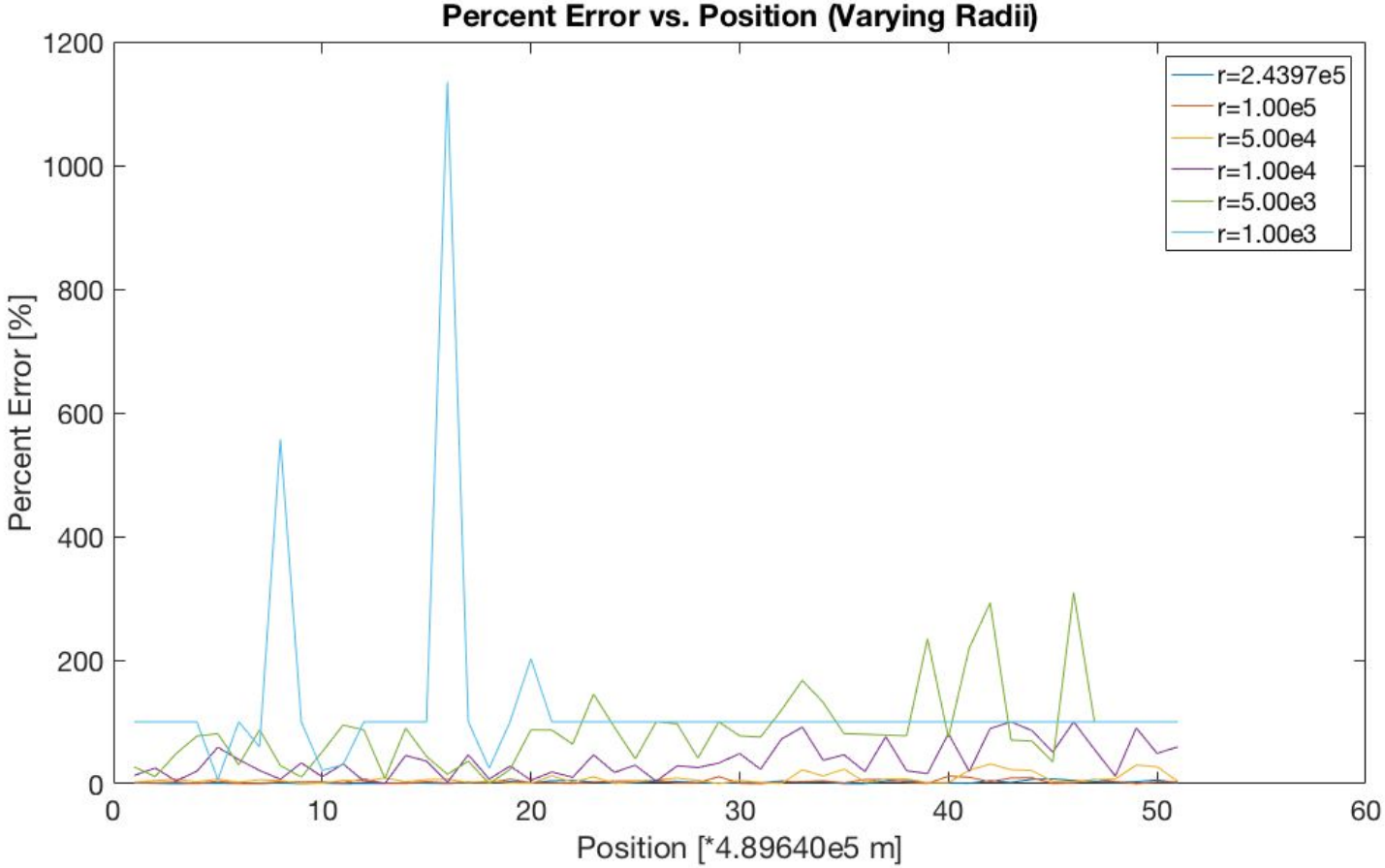

View Factor vs Position for Planetary Fly-by (Probe Radius = 1.00e5m)
×10^-4
1.2
1
0.8
0.6
0.4
0.2
0
View Factor
0
10
20
30
40
50
60
Position [*4.89640e5 m]

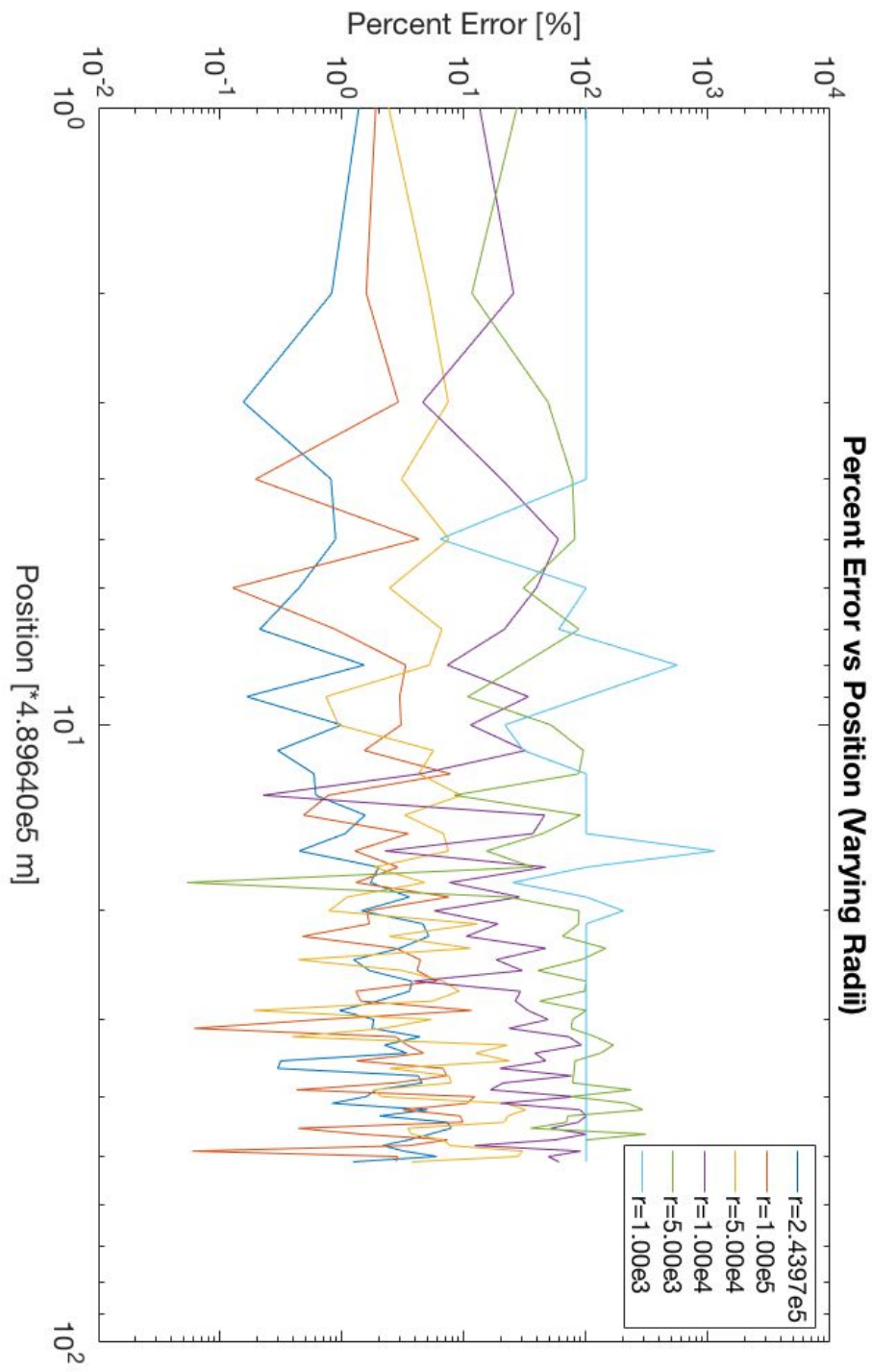

View Factor vs Position for Planetary Fly-by (Probe Radius = 5.00e4m)
×10^-5
3
2.5
2
1.5
1
0.5
0
View Factor
0
10
20
30
40
50
60
Position [*4.89640e5 m]

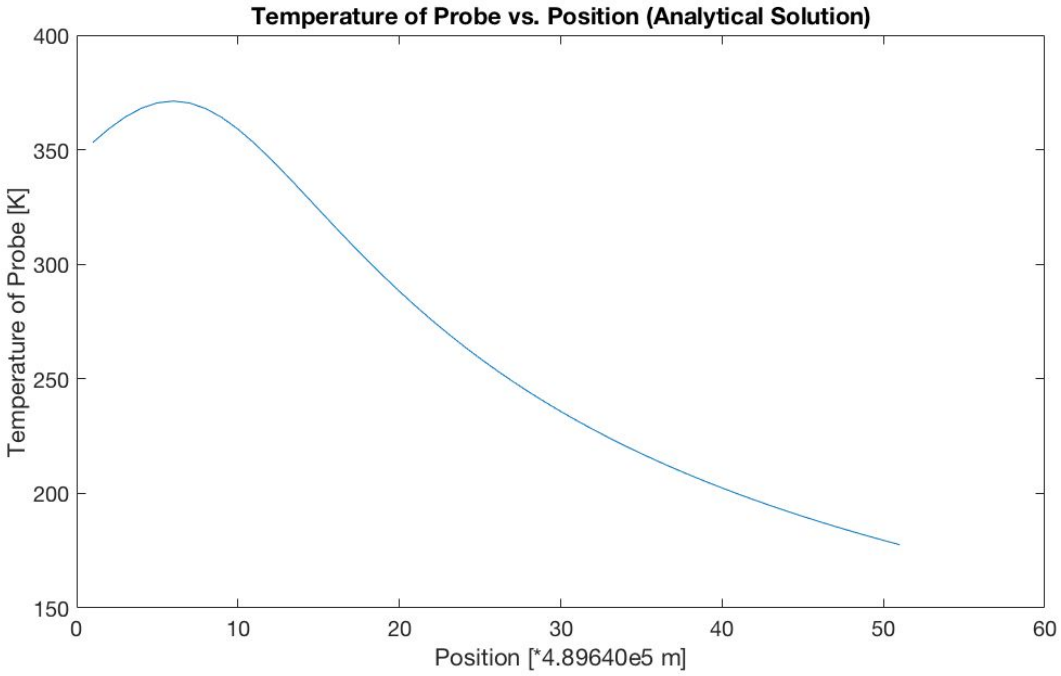

View Factor vs Position for Planetary Fly-by (Probe Radius = 1.00e4m)
×10^-6
View Factor
Position [*4.89640e5 m]
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
10
20
30
40
50
60

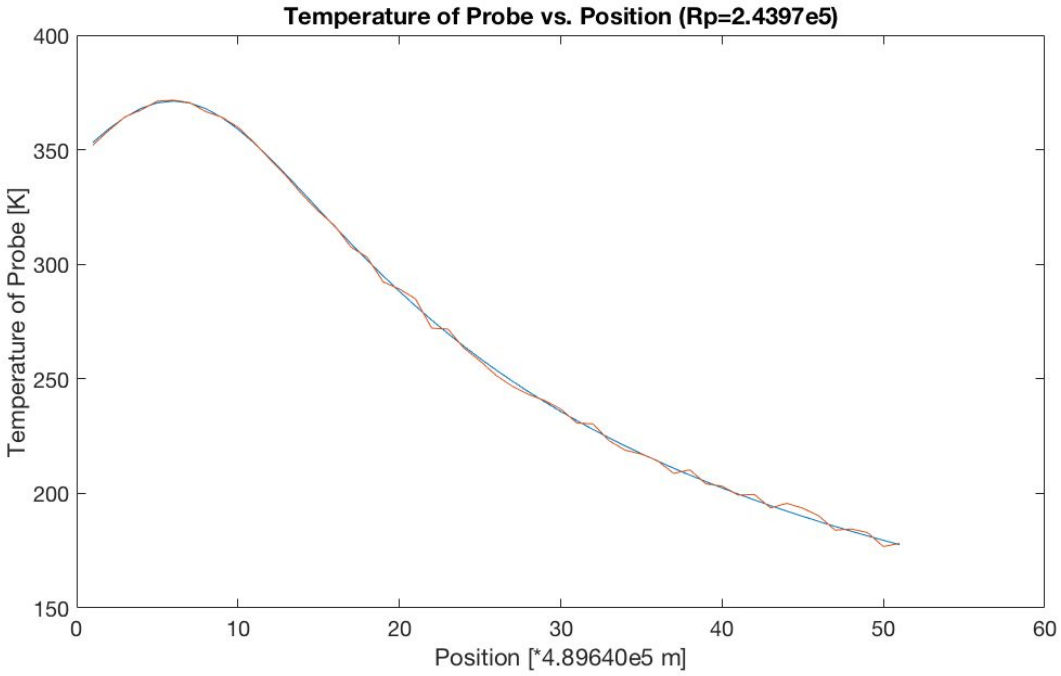

View Factor vs Position for Planetary Fly-by (Probe Radius = 5.00e3m)
×10^-7
View Factor
Position [*4.89640e5 m]
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
10
20
30
40
50
60

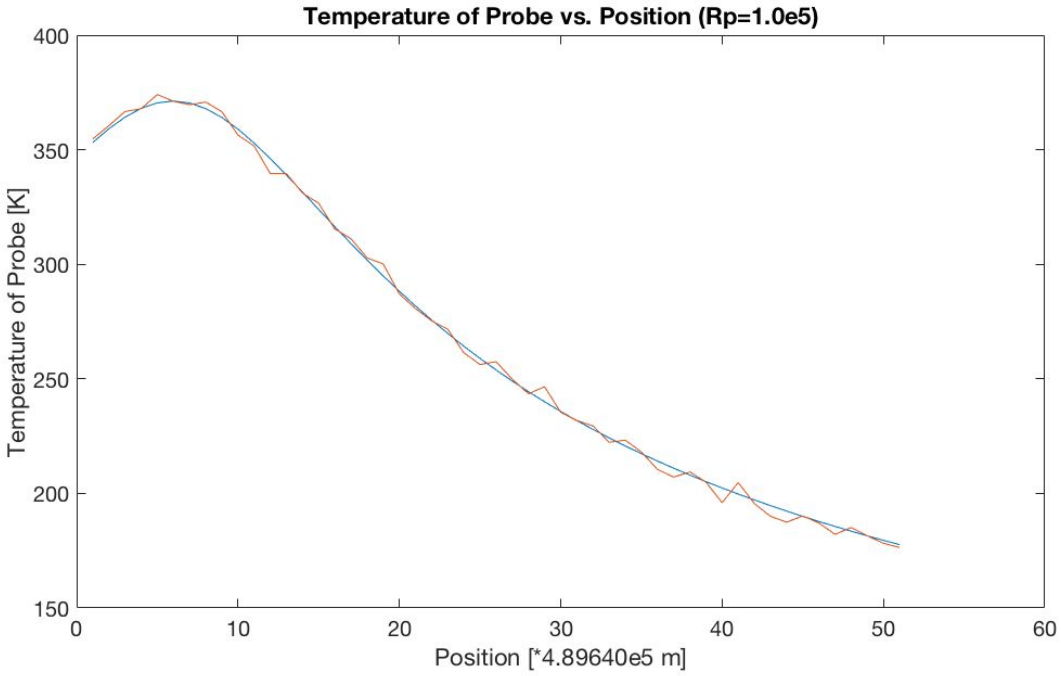

View Factor vs Position for Planetary Fly-by (Probe Radius = 1.00e3m)
×10^-8
View Factor
Position [*4.89640e5 m]
0
1
2
3
4
5
6
7
0
10
20
30
40
50
60

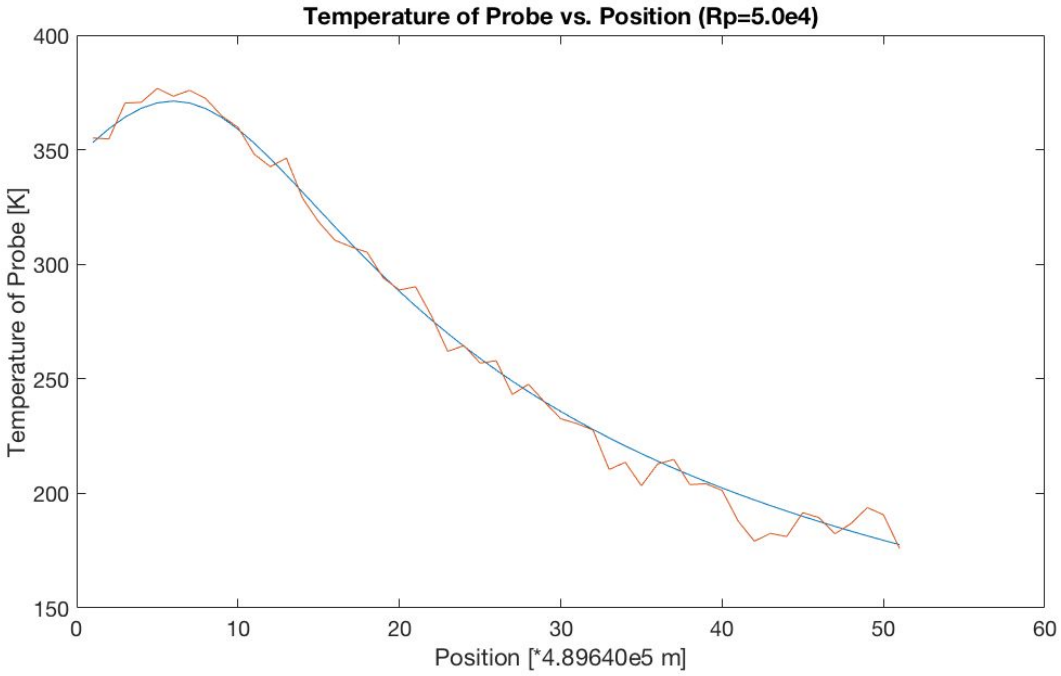

Percent Error vs. Position (Varying Radii)
Percent Error [%]
Position [*4.89640e5 m]
r=2.4397e5
r=1.00e5
r=5.00e4
r=1.00e4
r=5.00e3
r=1.00e3
0
200
400
600
800
1000
1200
0
10
20
30
40
50
60

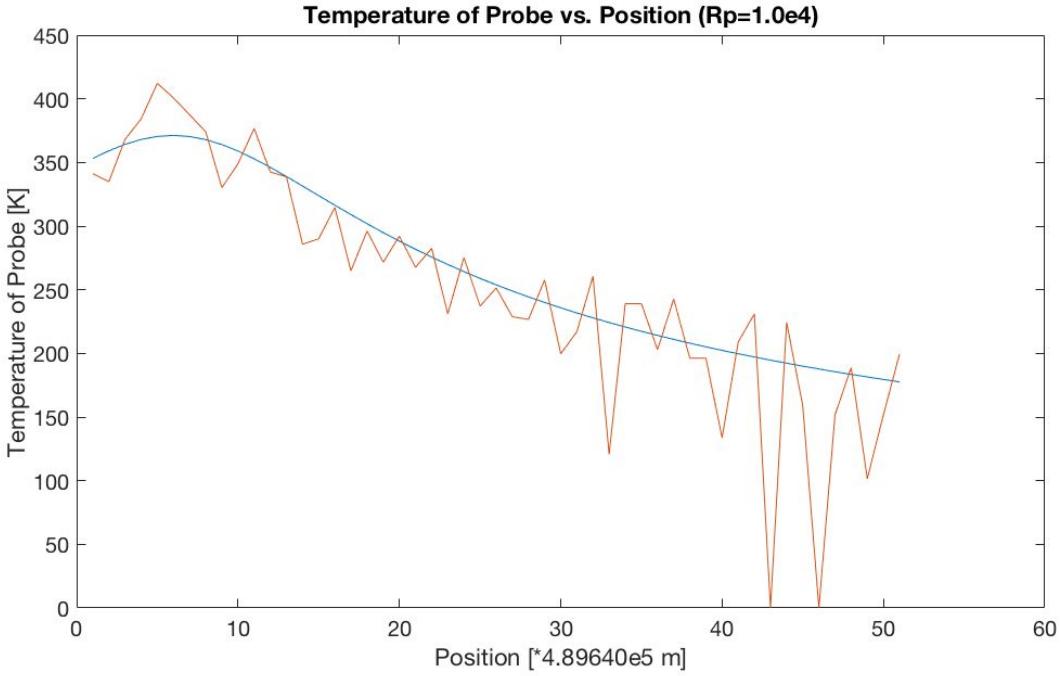

Percent Error vs Position (Varying Radii)
Percent Error [%]
Position [*4.89640e5 m]
r=2.4397e5
r=1.00e5
r=5.00e4
r=1.00e4
r=5.00e3
r=1.00e3

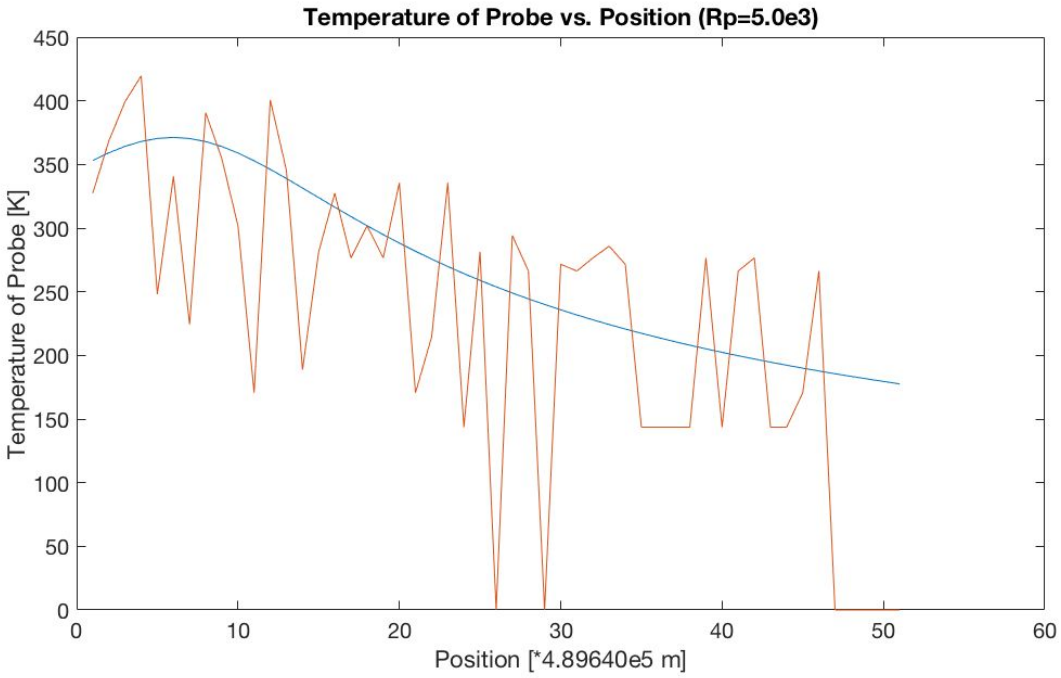

Temperature of Probe vs. Position (Analytical Solution)
Temperature of Probe [K]
Position [*4.89640e5 m]
400
350
300
250
200
150
0
10
20
30
40
50
60

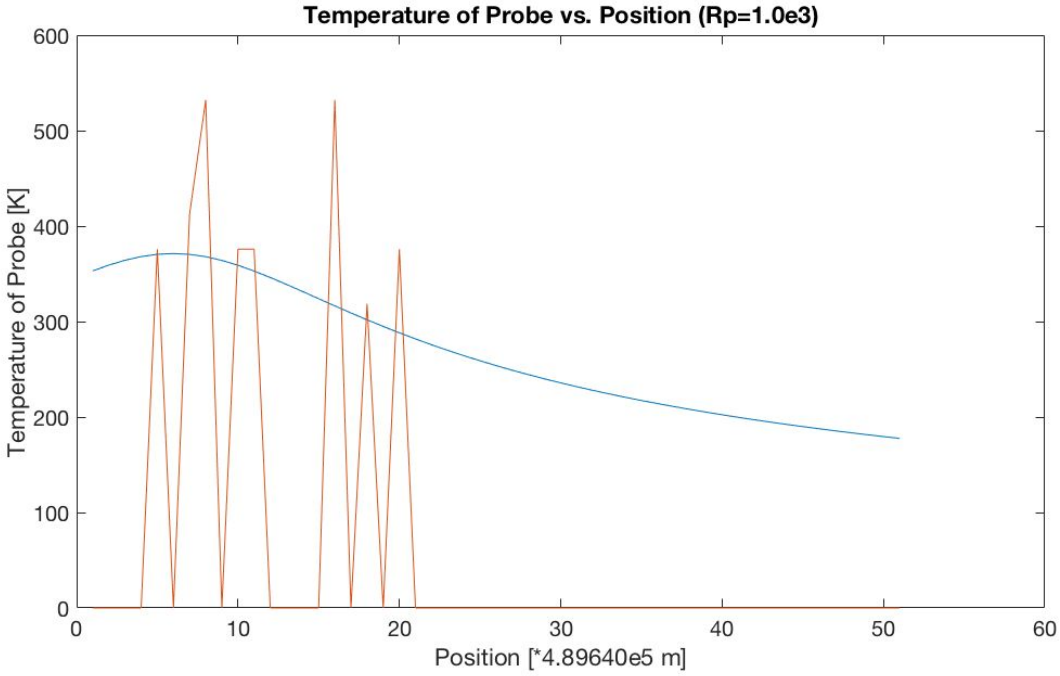

Temperature of Probe vs. Position (Rp=2.4397e5)
Temperature of Probe [K]
Position [*4.89640e5 m]
400
350
300
250
200
150
0
10
20
30
40
50
60

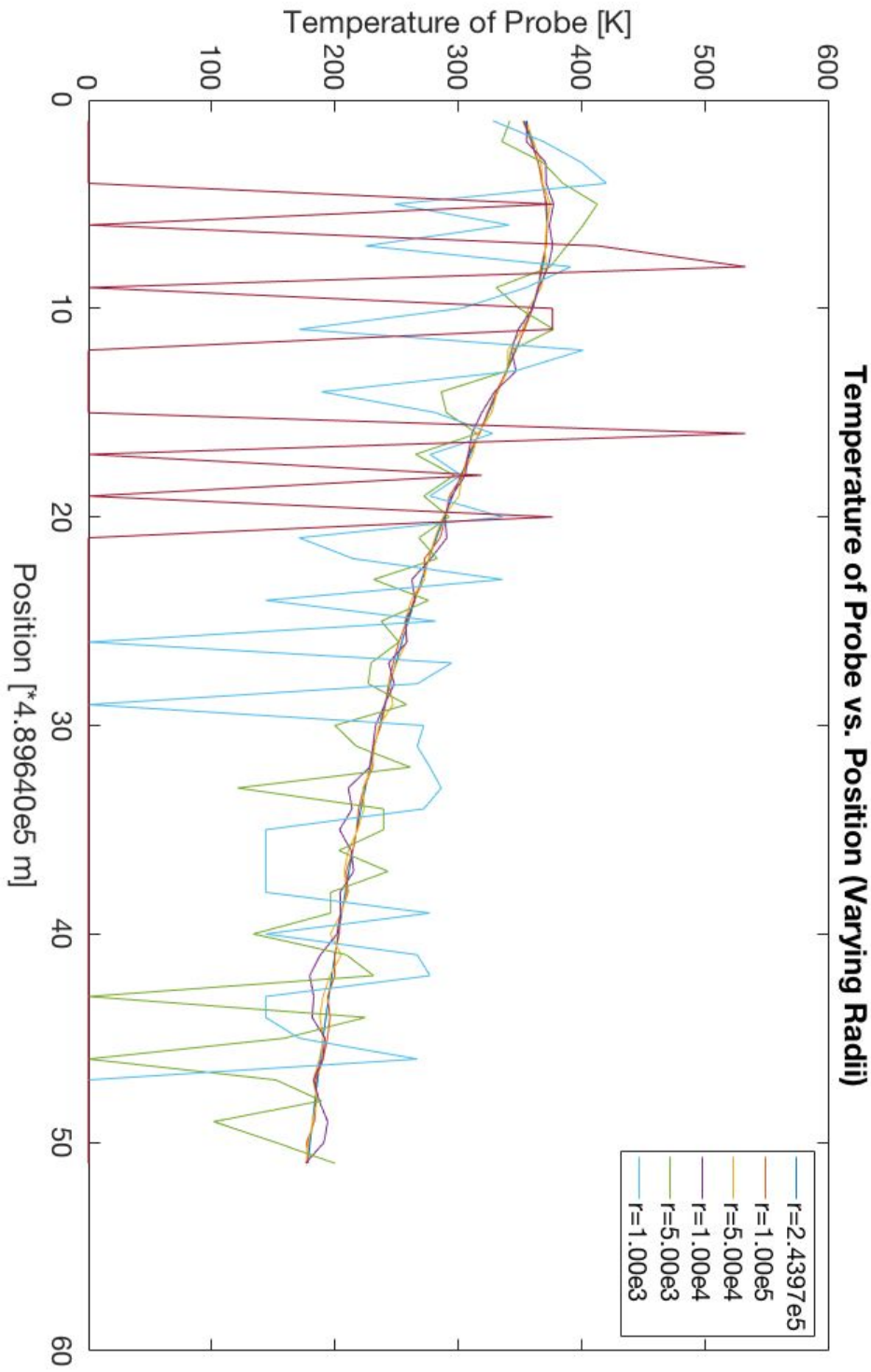

Temperature of Probe vs. Position (Rp=1.0e5)
Temperature of Probe [K]
Position [*4.89640e5 m]
400
350
300
250
200
150
0
10
20
30
40
50
60

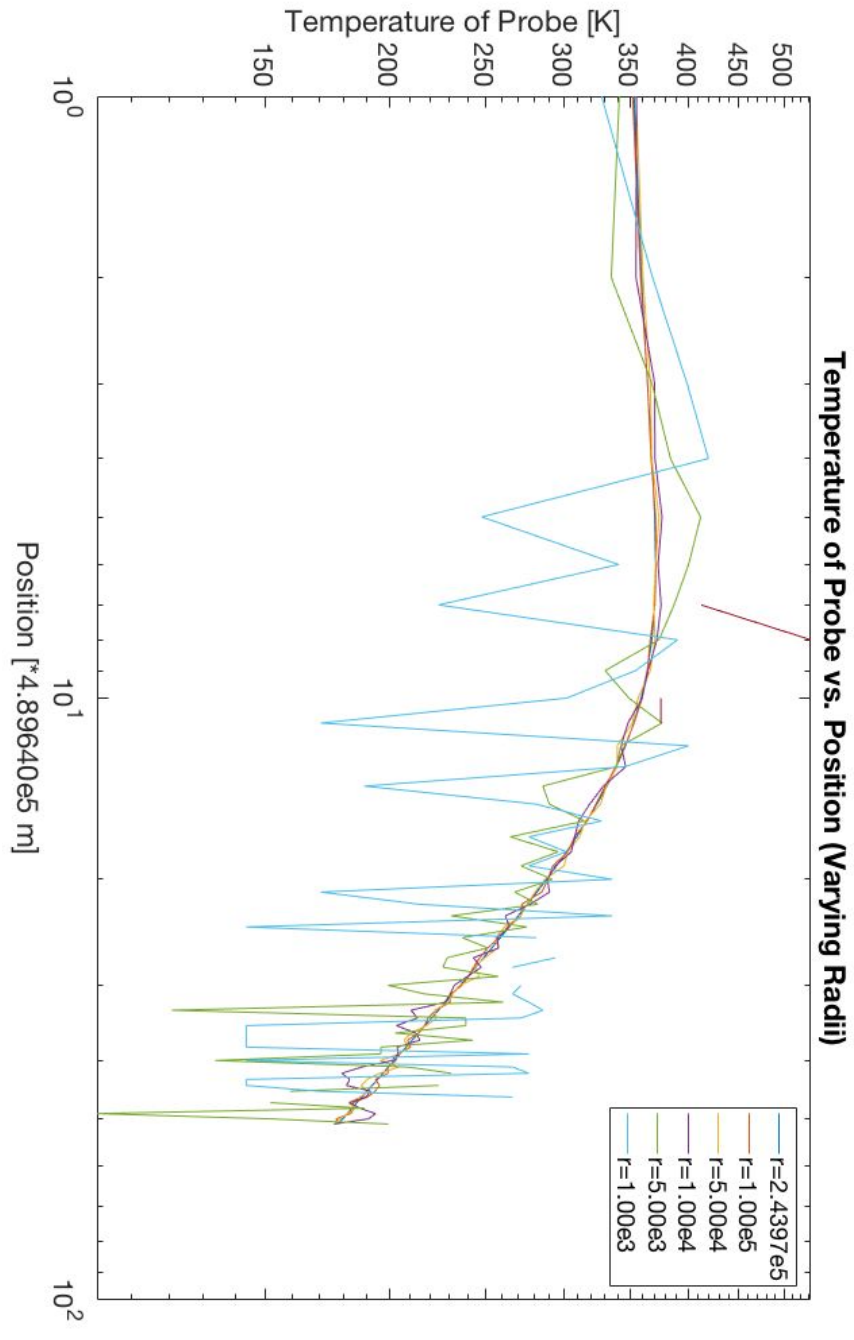

Temperature of Probe vs. Position (Rp=5.0e4)
Temperature of Probe [K]
Position [*4.89640e5 m]
400
350
300
250
200
150
0
10
20
30
40
50
60


Temperature of Probe vs. Position (Rp=1.0e4)
Temperature of Probe [K]
Position [*4.89640e5 m]
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60

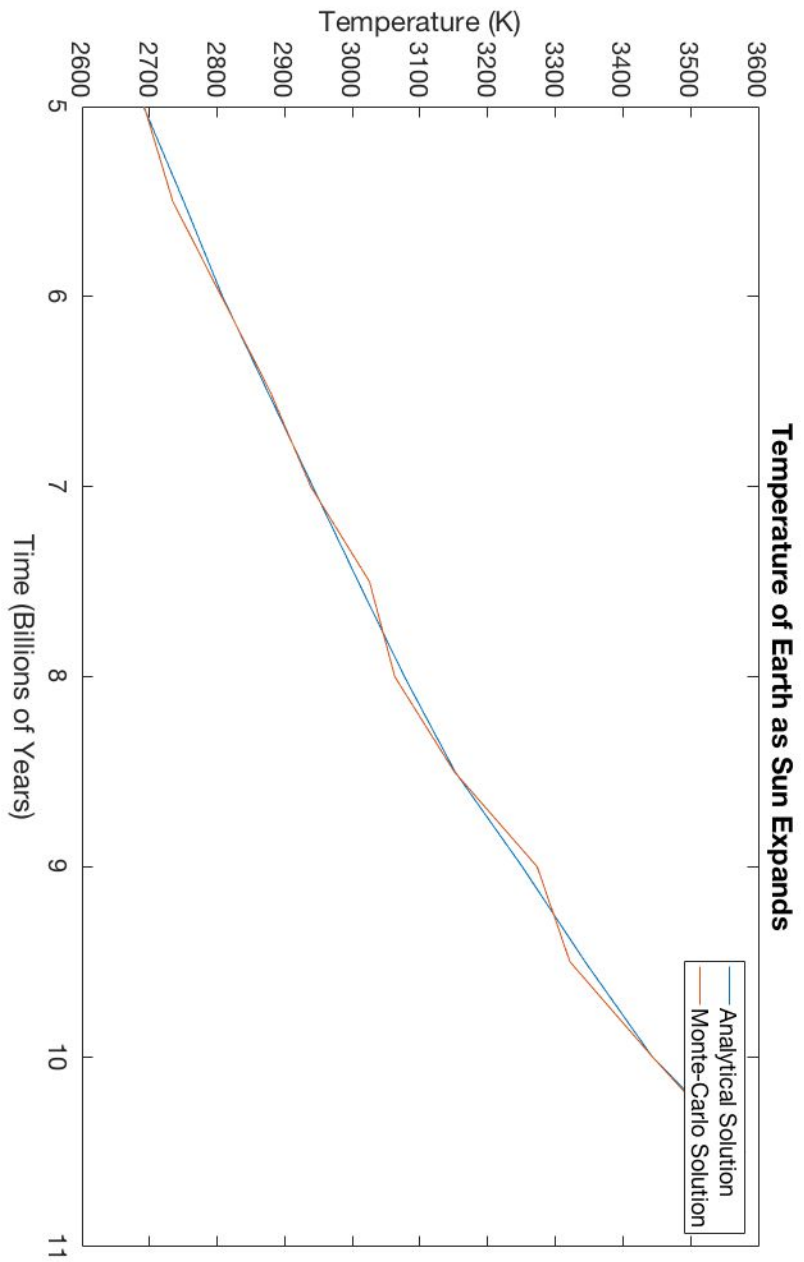

Temperature of Probe vs. Position (Rp=5.0e3)
Temperature of Probe [K]
Position [*4.89640e5 m]
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60


Temperature of Probe vs. Position (Rp=1.0e3)
Temperature of Probe [K]
Position [*4.89640e5 m]
0
100
200
300
400
500
600
0
10
20
30
40
50
60


Temperature of Probe vs. Position (Varying Radii)
Temperature of Probe [K]
Position [*4.89640e5 m]
0
100
200
300
400
500
600
0
10
20
30
40
50
60
r=2.4397e5
r=1.00e5
r=5.00e4
r=1.00e4
r=5.00e3
r=1.00e3


Temperature of Probe vs. Position (Varying Radii)
Temperature of Probe [K]
Position [*4.89640e5 m]
r=2.4397e5
r=1.00e5
r=5.00e4
r=1.00e4
r=5.00e3
r=1.00e3

Expanding Sun Experimental Results:



Halley's Comet Numerical Results: