Time-Dependent Behavior of Heat in a Cubic Satellite

Jonathan Bunton

December 4, 2017

Abstract

Before launching objects into low orbit space, it is prevalent to study their expected behavior in given conditions. One such behavior is the expected temperature variations at the time of launch, moving foward in time. To determine this behavior, we can utilize the fact that in general, the temperature of a material over time $T(x, y, z, t)$ is governed by the heat equation. [5] This partial differential equation is given by:

$$\nabla^2 T = \frac{1}{\alpha}\frac{dT}{dt}$$

To begin to solve this equation, we require proper boundary conditions. In this particular case, we consider a tidally locked 0.1 meter cubic mass of polystyrene foam, coated with polysilicon, initially at 30 K. Centered in the polystyrene foam is a 0.4 meter cubic aluminum mass which stays at a constant 100 K. This system serves as our "satellite." To calculate the change in temperature over time, we consider two relevant methods of heat transfer: conduction and radiation. All outside faces of the satellite are able to emit radiation, and we assume the face nearest the sun absorbs thermal energy, with both absorption and emission values being governed by the Stefan-Boltzmann law. [6] The conduction of heat is considered at all points on the satellite, governed by the Laplacian operator in the heat equation.

To numerically calculate these values, we discretize our space by dividing the satellite into a 3D mesh. To find values for $\nabla^2 T$, we utilize a central difference method to approximate the second derivatives at each point. We consider this and the diffusivity constant, α, which gives a material's tendency to conduct heat, alongside the radiation terms on the satellite faces to find a value for $\frac{dT}{dt}$.

Once this value is determined at a particular time, we need only numerically integrate to find the values at points further in time. In this project, we use fourth-order Runge-Kutta to perform accurate integration of these values in time.[4] The resulting integration yields the temperature behavior at all points over a set number of timesteps.

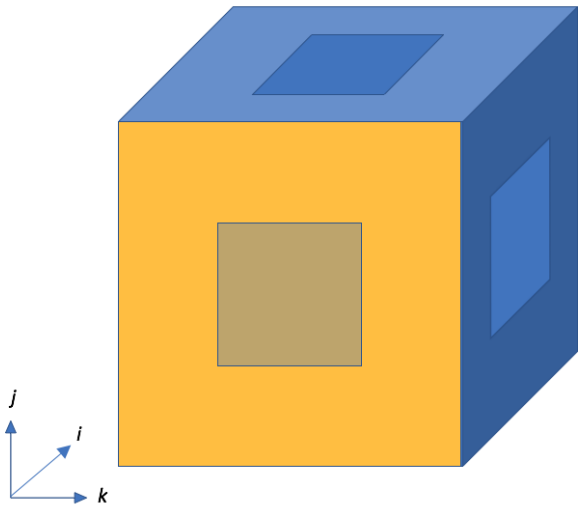


Figure 1: An image indicating the orientation of our cubic satellite in space, alongside the position of the interior cube, marked on the outer faces.

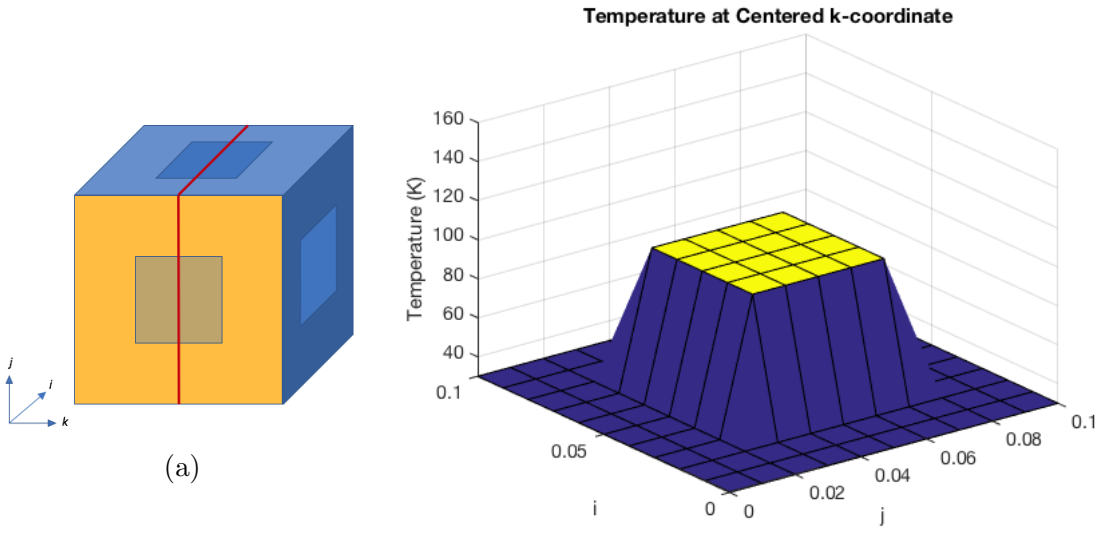
The results differ from our expected steady-state behavior outlined in prior work [2], but this work neglected the diffusivity and specific heats of each material that govern the satellite components' susceptibility to changes in temperature. By scaling up these constants from their true values, we can see effects occur much faster and make accurate comments on the temperature trends without wasting computing power. These trends indicate decaying exponential functions govern the temperature along the faces of the box. In addition, the interior 100 K aluminum box acts initially as a heat source to the 30 K foam, but as time goes on, the box acts more as a heat sink for the warming foam outside.

Results

We have oriented the satellite with the axes i, j, k as shown in figure 1. The face marked with yellow, $i = 1$, is facing the sun. This orientation is consistent through all plots.

Temperature Distribution Results

Because the results are effectively five-dimensional (x, y, z in space, with a temperature and time at each point), the results are plotted as planes through the satellite cube at particular times. All plots are done in this manner, with an image indicating the particular plane cut through the satellite. For reference, the initial conditions placed on the system are shown in fig. 2.



(a)

(b)

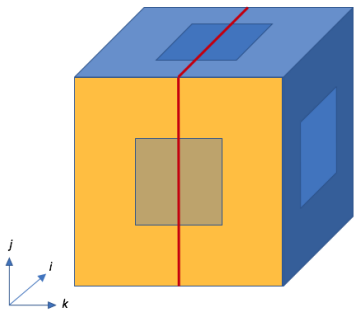
Figure 2: A plot of the initial temperature distribution through a cross-section at the center of the k-axis.

Using the constants for the specific heat and diffusivity of the various materials within the satellite, a cut through all i, j, and k planes are identical and shown after approximately 15 minutes in fig. 3. Most noticeable is the lack of heat build up on the face nearest the sun, where one would expect a more prominent effect. Because polysilicon has a very high specific heat and is very dense, it does not absorb the radiated heat from the sun effectively. [1] In addition, polystyrene's incredibly low diffusivity constant indicates the material does not conduct heat very quickly, which is also reflected by the amount of time required to reach the distribution shown. [3]

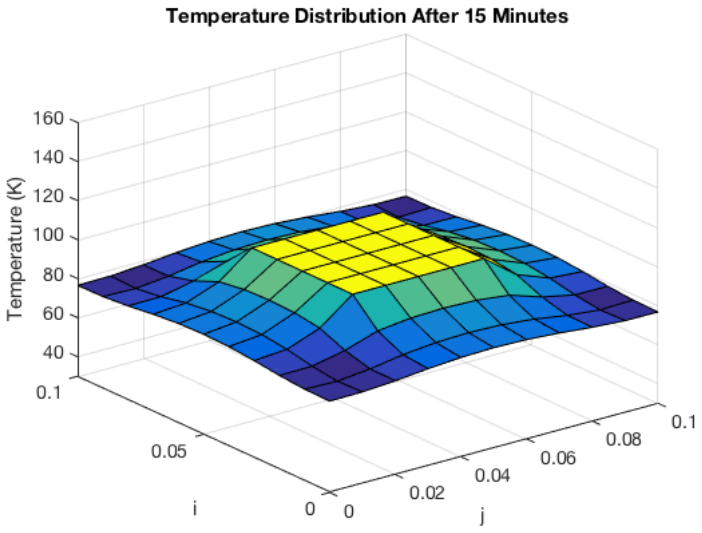
Moving further forward in time leads to a nearly flat line temperature distribution at 100 K. This makes sense in the context, as the only release of heat is via radiation from the polysilicon layers which do so poorly. The poor radiation keeps the heat generated by the internal box contained in the system as it slowly raises (through the poor diffusivity in the polystyrene) in temperature to almost level with the 100 K, as shown in fig. 4, which represents the distribution after approximately 30 minutes.

Previous work suggested a different shape for eventual temperature distribution, but this work failed to consider the type of material at hand. If we instead set this simulation's material-based constants (density, diffusivity, specific heat) to 1, it would model a system more closely parallel to the one analyzed in previous work. The result is shown in fig. 5.

In the i-axis cross section shown in fig. 6 the pronounced dip in the center of each face

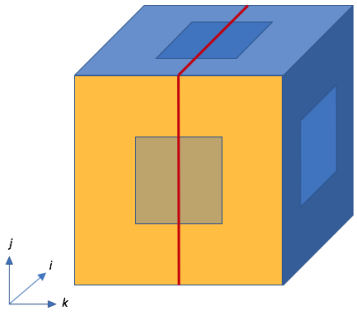


(a)

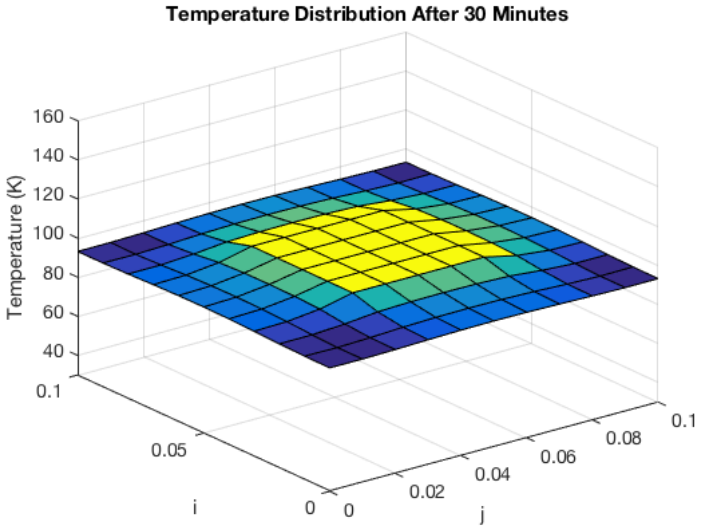


(b)

Figure 3: A plot of temperature through a cross-section at the center of the k-axis. This distribution is shared across the other two cross-sections, showing an exponential decay away from the 100 K satellite center. In this case, the center acts as a source of heat, and the heat absorbed from the sun is negligible.

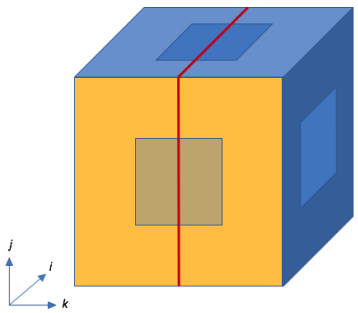


(a)

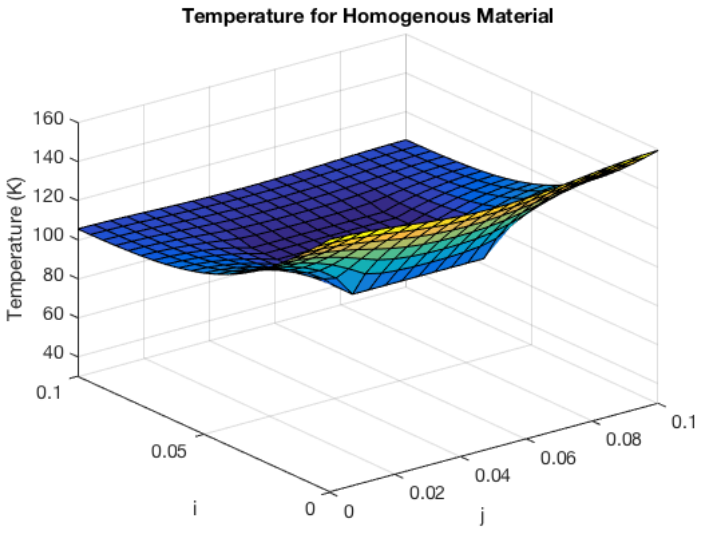


(b)

Figure 4: A plot of temperature through a cross-section at the center of the k-axis. This distribution is shared across the other two cross-sections, showing a very slight exponential decay from the center, where the faces radiate away heat.

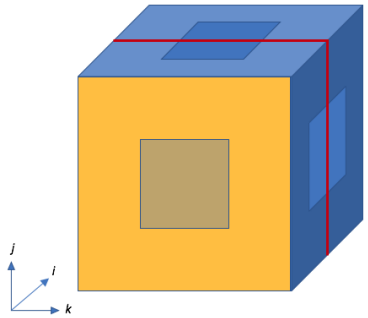


(a)

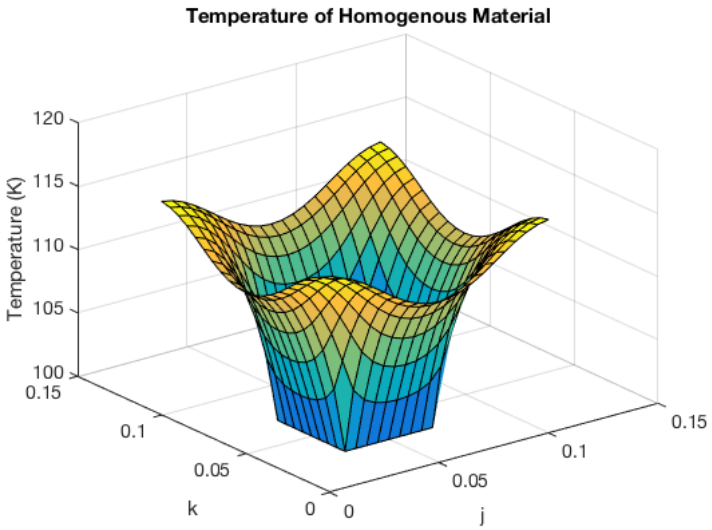


(b)

Figure 5: A plot of temperature through a cross-section at the center of the k-axis in a homogenous material. The result is a slight exponential decay moving away from the sun, with the aluminum box acting effectively as a heat sink.



(a)



(b)

Figure 6: A plot of temperature through a cross-section at the center of the i-axis in a homogenous material. Clear exponential decays show the eventual role of the center aluminum as a heat sink, rather than source.

is clear. This effect occurs because center face points are in contact with more points in the satellite than the edges or corners, meaning in the central difference method they are averaged with fewer internal values. Because in our simulation we considered space a vacuum with nothing to conduct thermal energy from, our central difference method effectively averages with less internal points, causing these points to hold heat more than others.

Discussion

Ultimately, these results point to a very thermally stable satellite if crafted from the materials laid out in our assumptions. The long term behavior shows that the primary source of heat in the system is the center aluminum box, meaning if it truly does operate as assumed at a constant temperature of 100 K, the remainder of the box will slowly rise to match the 100 K. 100 K is relatively low in temperature, which means the box is safe in low orbit.

Compared to previous simulations, this result is strikingly different. This mostly comes into effect as a result of the simplifying assumptions made in previous work, where the type of material in question had no bearing on the long term behavior, which is not entirely true. As proof of this as the reason for discrepancy, a quick simulation of the same satellite but instead cast as a homogenous material with all characteristic constants set to one yields results closer to previous simulations.

References

[1] American Elements, 1093 Broxton Avenue, Los Angeles, CA. *Polycrystalline Silicon*, 2017. Technical data sheet page for polycrystalline silicon sales in ingot or powder form.

[2] Jonathan Bunton. Solving for the steady-state solution of the heat equation in a cubic satellite. Previous Submission, October 2017.

[3] W Glenz. Rigid polystyrene foam (eps, xps). 100:73–77, 01 2010.

[4] Martin W. Kutta. Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Zeitschrift für Methematik und Physik*, 1901.

[5] T.N. Narasimhan. Fourier's heat conduction equation: history, influence, and connections. *Reviews of Geophysics*, February 1999.

[6] Kim Sharp and Franz Matschinsky. Translation of ludwig boltzmanns paper on the relationship between the second fundamental theorem of the mechanical theory of heat and probability calculations regarding the conditions for thermal equilibrium sitzungberichte der kaiserlichen akademie der wissenschaften. mathematisch-naturwissen classe. abt. ii, lxxvi 1877, pp 373-435 (wien. ber. 1877, 76:373-435). reprinted in wiss. abhandlungen, vol. ii, reprint 42, p. 164-223, barth, leipzig, 1909. *Entropy*, 17(4):1971–2009, 2015.

Appendix

Attached is the source code for this project, written in Fortran.

```
program theateq
use derivatemp
use constants
implicit none
real*8, allocatable, dimension(:,:,:,:) :: T
real*8, allocatable, dimension(:,:,:) :: dT
real*8 :: tstep = 1E-7, dx
integer, parameter :: timesteps = 100000
integer :: ilow, ihigh, i, j, k, l, n = 2, resolution = 100
integer :: speed = 9001


n = 10*n + 1
ilow = int(0.3*n+0.7)
ihigh = int(0.7*n+0.3)
dx = boxwidth/float(n-1)
print*, 'How much computing POWER?'
read*, speed
if (speed .le. 9000) then
allocate(T(n,n,n,2),dT(n,n,n))
T(:,:,:,1) = 30
T(ilow:ihigh,ilow:ihigh,ilow:ihigh,1) = 100
do i = 1,timesteps-1
    !K1
    call tderiv(T(:,:,:,1), dT)
    T(:,:,:,2) = T(:,:,:,1) + (tstep/6.0)*dT
    !K2
    call tderiv(T(:,:,:,1) + (tstep/2.0)*dT, dT)
    T(:,:,:,2) = T(:,:,:,2) + (tstep/3.0)*dT
    !K3
    call tderiv(T(:,:,:,1) + (tstep/2.0)*dT, dT)
    T(:,:,:,2) = T(:,:,:,2) + (tstep/3.0)*dT
    !K4
    call tderiv(T(:,:,:,1) + (tstep)*dT, dT)
```

```
    T(:,:,:,2) = T(:,:,:,2) + (tstep/6.0)*dT
    T(:,:,:,1) = T(:,:,:,2)
end do
open(unit=100, file='ans.txt')
print*,'Writing data...'
do i = 1,n
    do j = 1,n
        do k = 1,n
        write(100,*) i, j, k, 1, T(i,j,k,2)
        end do
    end do
end do
close(100)



else
allocate(T(n,n,n,timesteps), dT(n,n,n))
T(:,:,:,1) = 30
T(ilow:ihigh,ilow:ihigh,ilow:ihigh,1) = 100
do i = 1,timesteps-1
    !K1
    call tderiv(T(:,:,:,i), dT)
    T(:,:,:,i+1) = T(:,:,:,i) + (tstep/6.0)*dT
    !K2
    call tderiv(T(:,:,:,i) + (tstep/2.0)*dT, dT)
    T(:,:,:,i+1) = T(:,:,:,i+1) + (tstep/3.0)*dT
    !K3
    call tderiv(T(:,:,:,i) + (tstep/2.0)*dT, dT)
    T(:,:,:,i+1) = T(:,:,:,i+1) + (tstep/3.0)*dT
    !K4
    call tderiv(T(:,:,:,i) + (tstep)*dT, dT)
    T(:,:,:,i+1) = T(:,:,:,i+1) + (tstep/6.0)*dT
end do
print*,'Writing data...'

open(unit=100, file = 'ans.txt')
```

```
do l = 1, int(timesteps/resolution)
   do i = 1,n
      do j = 1,n
         do k = 1,n
         write(100,*) i, j, k, l,  T(i,j,k,resolution*l)
         end do
      end do
   end do
end do
close(100)
end if

deallocate(dT,T)

open(unit=200,file = 'info.txt')
write(200,*) n
write(200,*) timesteps
write(200,*) boxwidth/float(n-1)
write(200,*) resolution
close(200)

end program theateq
```

```
module derivatemp
use constants
implicit none
contains

subroutine tderiv(T, dT)
real*8, dimension(:,:,:) :: T, dT
real*8 :: dx
integer :: n, i, j, k, ilow, ihigh
dT = 0
n = size(T,1)
dx = boxwidth/dble(n-1)
ilow = int(0.3*n+0.7)
ihigh = int(0.7*n+0.3)
```

```
!Corners
!1st index = 1 edges absorb heat from sun
dT(1,1,1) = sigma*((Tsun**4)/(csil*denssil*dx))*(Rsun/Rorbit)**2 -
     sigma*(T(1,1,1)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(1,1,2)+T(1,2,1)+T(2,1,1)-3.0*T
              (1,1,1))

dT(1,1,n) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)**2 -
     sigma*(T(1,1,n)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(1,1,n-1)+T(1,2,n)+T(2,1,n)-3.0*T
              (1,1,n))

dT(1,n,n) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)**2 -
     sigma*(T(1,n,n)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(1,n,n-1)+T(1,n-1,n)+T(2,1,n)-3.0*T
              (1,n,n))

dT(1,n,1) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)**2 -
     sigma*(T(1,n,1)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(1,n,2)+T(1,n-1,1)+T(2,n,1)-3.0*T(1,
              n,1))



dT(n,n,n) = -sigma*(T(n,n,n)**4)/(csil*denssil*dx)*(dx**2) &
       + alphasil/(dx**2)*(T(n,n,n-1)+T(n,n-1,n)+T(n-1,n,n)-3.0*T(n
           ,n,n))
dT(n,n,1) = -sigma*(T(n,n,1)**4)/(csil*denssil*dx)*(dx**2) &
       + alphasil/(dx**2)*(T(n,n,2)+T(n,n-1,1)+T(n-1,n,1)-3.0*T(n,n
           ,1))
dT(n,1,1) = -sigma*(T(n,1,1)**4)/(csil*denssil*dx)*(dx**2) &
       + alphasil/(dx**2)*(T(n,1,2)+T(n,2,1)+T(n-1,1,1)-3.0*T(n
           ,1,1))
dT(n,1,n) = -sigma*(T(n,1,n)**4)/(csil*denssil*dx)*(dx**2) &
       + alphasil/(dx**2)*(T(n,1,n-1)+T(n,2,n)+T(n-1,1,n)-3.0*T(n
           ,1,n))
```

```
!Edges
do i = 2, n-1
   !1st index = 1 edges absorb heat from sun
   dT(1,1,i) =  sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)
      **2 - sigma*(T(1,1,i)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(1,1,i+1)+T(1,1,i-1)+T(1,2,i)+T(2,1,
            i)-4.0*T(1,1,i))

   dT(1,n,i) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)
      **2 - sigma*(T(1,n,i)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(1,n,i+1)+T(1,n,i-1)+T(1,n-1,i)+T(2,
            n,i)-4.0*T(1,n,i))

   dT(1,i,n) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)
      **2 - sigma*(T(1,i,n)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(1,i+1,n)+T(1,i-1,n)+T(1,i,n-1)+T(2,
            i,n)-4.0*T(1,i,n))

   dT(1,i,1) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)
      **2 - sigma*(T(1,i,1)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(1,i+1,1)+T(1,i-1,1)+T(2,i,1)+T(1,i
            ,2)-4.0*T(1,i,1))



   dT(n,1,i) = -sigma*(T(n,1,i)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(n,1,i-1)+T(n,1,i+1)+T(n,2,i)+T(n
            -1,1,i)-4.0*T(n,1,i))
   dT(n,n,i) = -sigma*(T(n,n,i)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(n,n,i-1)+T(n,n,i+1)+T(n,n-1,i)+T(n
            -1,n,i)-4.0*T(n,n,i))
   dT(n,i,1) = -sigma*(T(n,i,1)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(n,i,2)+T(n,i+1,1)+T(n,i-1,1)+T(n-1,
            i,1)-4.0*T(n,i,1))
   dT(n,i,n) = -sigma*(T(n,i,n)**4)/(csil*denssil*dx) &
         + alphasil/(dx**2)*(T(n,i,n-1)+T(n,i+1,n)+T(n,i-1,n)+T(n
```

```
            -1,i,n)-4.0*T(n,i,n))
    dT(i,1,1) = -sigma*(T(i,1,1)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,1,2)+T(i,2,1)+T(i+1,1,1)+T(i
              -1,1,1)-4.0*T(i,1,1))
    dT(i,n,1) = -sigma*(T(i,n,1)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,n,2)+T(i,n-1,1)+T(i+1,n,1)+T(i-1,
              n,1)-4.0*T(i,n,1))
    dT(i,1,n) = -sigma*(T(i,1,n)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,1,n-1)+T(i,2,n)+T(i+1,1,n)+T(i
              -1,1,n)-4.0*T(i,1,n))
    dT(i,n,n) = -sigma*(T(i,n,n)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,n,n-1)+T(i,n-1,n)+T(i+1,n,n)+T(i
              -1,n,n)-4.0*T(i,n,n))

end do


! Faces
do i = 2, n-1
    do j = 2, n-1
    dT(1,i,j) = sigma*((Tsun**4)/(csil*denssil*dx))*(rsun/rorbit)
       **2 - sigma*(T(1,i,j)**4)/(denssil*dx) &
          + alphasil/(dx**2)*(T(1,i,j+1)+T(1,i,j-1)+T(1,i+1,j)+T(1,
              i-1,j)+T(2,i,j)-5.0*T(1,i,j))

    dT(i,1,j) = -sigma*(T(i,1,j)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,1,j+1)+T(i,1,j-1)+T(i+1,1,j)+T(i
              -1,1,j)+T(i,2,j)-5.0*T(i,1,j))
    dT(i,j,1) = -sigma*(T(i,j,1)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,j+1,1)+T(i,j-1,1)+T(i+1,j,1)+T(i
              -1,j,1)+T(i,j,2)-5.0*T(i,j,1))
    dT(n,i,j) = -sigma*(T(n,i,j)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(n,i,j+1)+T(n,i,j-1)+T(n,i+1,j)+T(n,
              i-1,j)+T(n-1,i,j)-5.0*T(n,i,j))
    dT(i,n,j) = -sigma*(T(i,n,j)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,n,j+1)+T(i,n,j-1)+T(i+1,n,j)+T(i
```

```
                -1,n,j)+T(i,n-1,j)-5.0*T(i,n,j))
    dT(i,j,n) = -sigma*(T(i,j,n)**4)/(csil*denssil*dx) &
          + alphasil/(dx**2)*(T(i,j-1,n)+T(i,j+1,n)+T(i+1,j,n)+T(i
              -1,j,n)+T(i,j,n-1)-5.0*T(i,j,n))
    end do
end do

do i = 2,n-1
    do j = 2,n-1
        do k = 2,n-1
        dT(i,j,k) = alphasty/(dx**2)*(T(i,j,k+1)+T(i,j,k-1)+T(i,j+1,
            k)+T(i,j-1,k)+T(i-1,j,k)+T(i+1,j,k)-6.0*T(i,j,k))
        end do
    end do
end do

dT(ilow:ihigh,ilow:ihigh,ilow:ihigh) = 0

end subroutine
end module
```

MATLAB Plotting Script

```
A = importdata('ans.txt');
info = importdata('info.txt');
n = info(1);
tsteps = info(2);
dx = info(3);
div = info(4);
%T = zeros(n,n,n,tsteps/div);
T = zeros(n,n,n,1);

count = 1;
for time = 1:tsteps/div
    for i = 1:n
        for j = 1:n
```

```
                for k = 1:n
                    T(A(count,1),A(count,2),A(count,3),A(count,4)) = A
                        (count,5);
                    count = count+1;
                end
            end
        end
    end

limits = [0;0.1;0;0.1;30;160];
[x y] = meshgrid(0:dx:0.1+dx);
mid = ceil(n/2);
for i = 1:tsteps/div
surf(x,y,squeeze(T(:,:,mid,i)));
%surf(x,y,squeeze(T(:,mid,:,1)));
axis(limits)
drawnow
end
xlabel('j');
ylabel('i');
zlabel('Temperature (K)');
title('Temperature at Centered k-coordinate');
```