Regosketch 2 : Simulate Heat Transfer around a Power Cable buried in LunarRegolith

Mick Goulish
31 March, 2023

Overview

I have been planning to bury my power cables 2 meters deep in lunar regolith. I have them figured out to only be dissipating 1 Watt of heat per linear meter of cable. *Surely,* thought I, *that can't be a problem!*

Well. If this simulation is anywhere near reality, then yes. It *can* be a problem. Big problem.

I model the heat transfer in 1-cm concentric shells outward from the cable. In the absence of any kind of fluid, I don't see any reason why heat would rise, so it just spreads outward in concentric cylindrical shells.

Eventually, those expanding cylindrical shells would intersect the surface and start radiating heat out into space, but I didn't take it that far. It became clear that it was becoming a problem well before we reach the 2-meter high surface.

Doing the Simulation

The basic idea is this:

- Take a 1-meter length of aluminum cable that is buried at an arbitrary depth in lunar regolith.
- Add 1 Joule per second (1 Watt) of energy to it. This is the amount of heat that I calculated (using a calculator here: https://www.electricalvolt.com/2022/04/cable-power-loss-calculator-formula-calculation/) would be lost per meter in my cable during power transmission.
- Use specific heat of aluminum to calculate how the cable's heat increases. (From starting temp of 230 K, about -46 F.)
- Do timesteps of 15 minutes, because 1-second timesteps would take forever to calculate. I ran the simulation for 1 year's worth of timesteps.
- At every timestep, use lunar regolith thermal conductivity to calculate how much heat is accepted by the 1-cm cylindrical shell around the cable.
- Use lunar regolith specific heat to calculate how much this new energy increases the temperature of the cylindrical shell around the cable.

- Keep doing that same calculation for concentric 1-cm shells outward from the cable.
- Quit doing a given timestep when the heat difference between the current shell and the next falls to less than 0.1 K. In succeeding timesteps the energy will build up until it crosses that threshold, and then it will be propagated. (There must be some threshold like this, or we would continue calculating tiny heat transfers outward forever in the first timestep.)

Assumptions

I barely know what I am doing here, and I had to make a lot of assumptions. Like:

- Thermal conductivity is given in units per meter, but I want to do 1-cm shells around my cable. I think that thinner layers of material conduct *more* heat, not less. So I multiplied those numbers by 100.
- The amount of heat lost must (I think) be proportional to the surface area that is losing the heat. So I assumed that the heat conductance numbers were per square meter, and scaled it by the surface area of each concentric shell.

If these assumptions are wrong, it will be easy to correct them in the code. By far the largest part of the work was not in the actual application of heat-transfer theory, but in all the other plumbing around it.

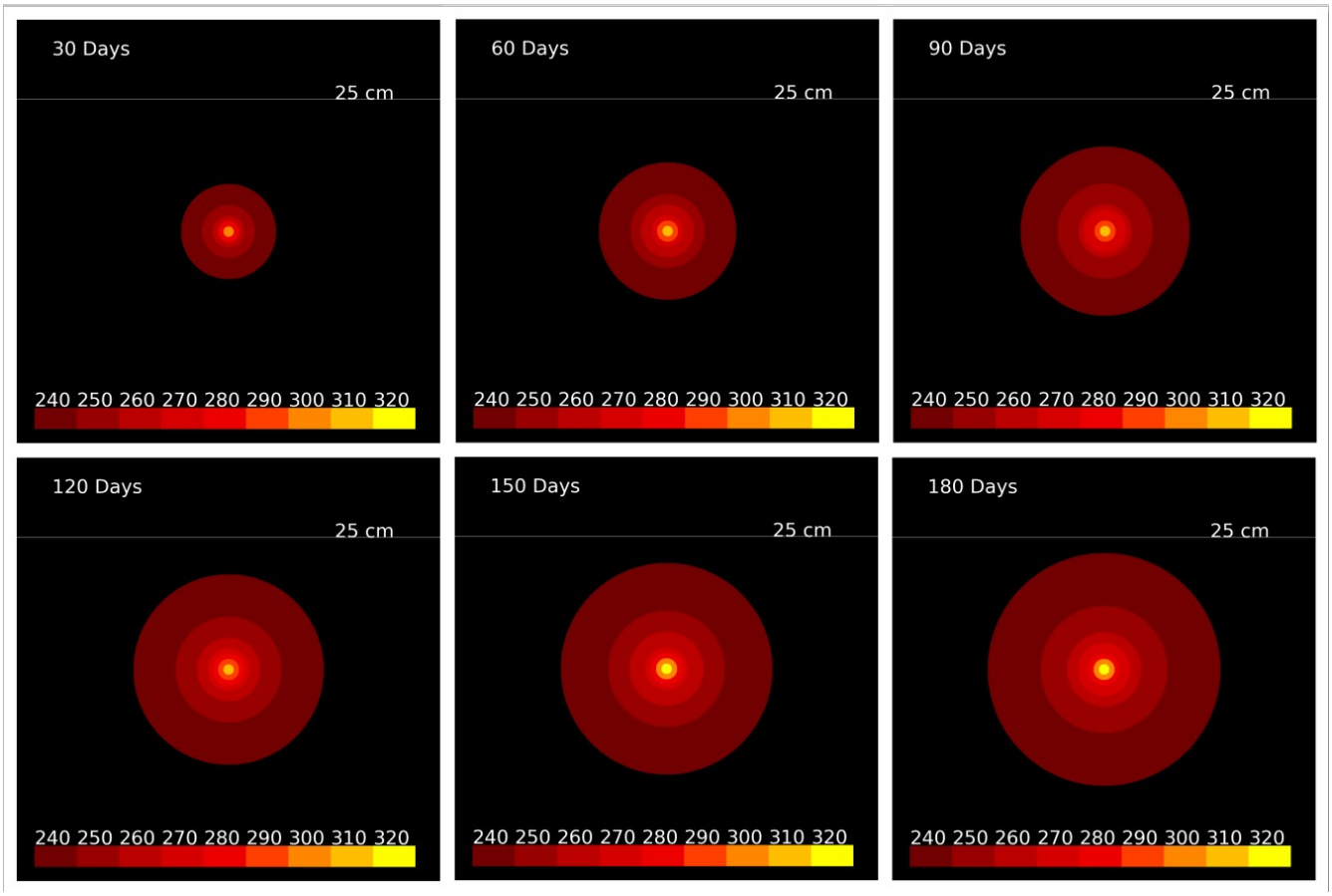
Results

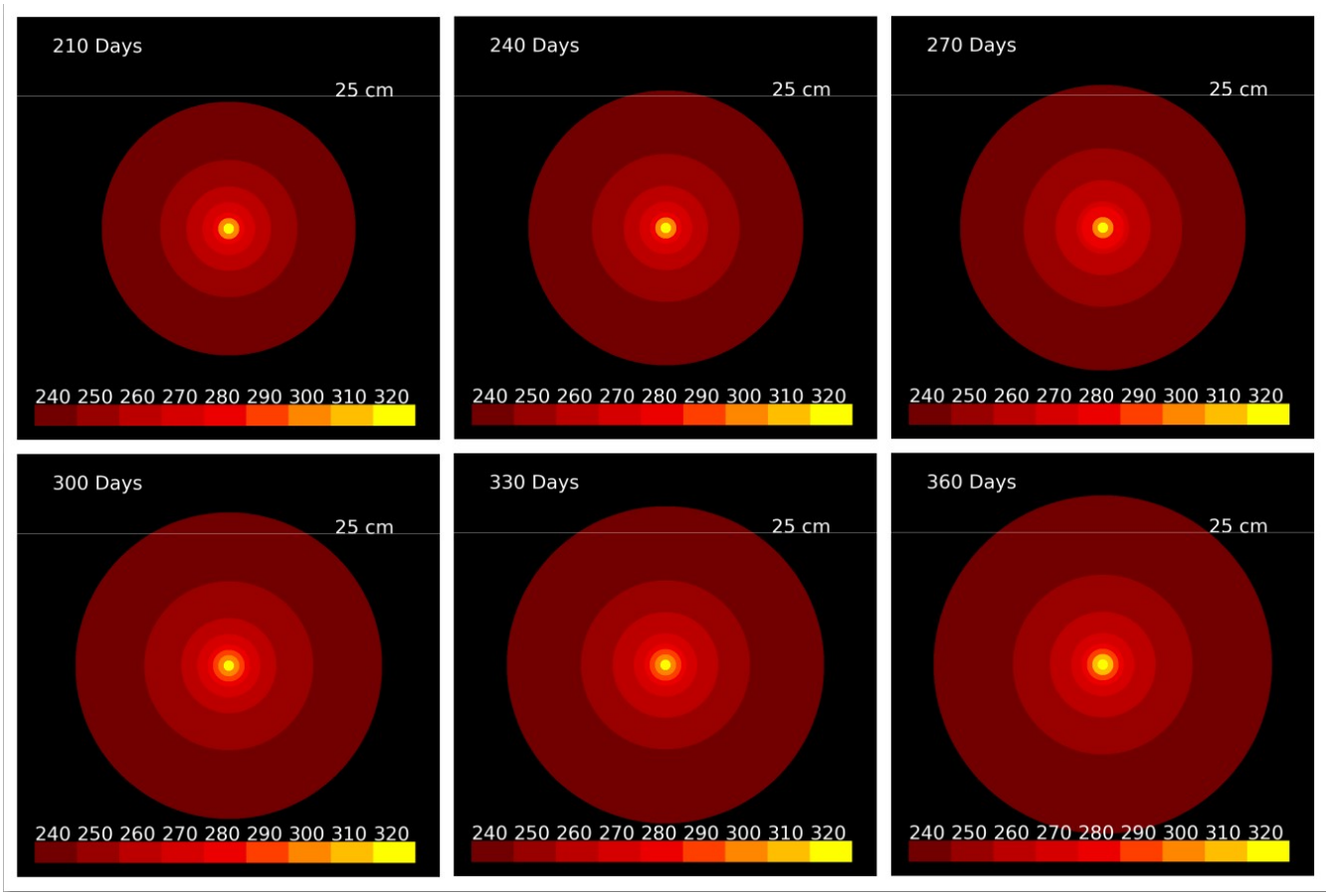
If this is anything near correct, then my plan for a 2-meter-deep cable is doomed.

One Watt per meter is indeed a small amount of power, so things don't heat up quickly. But lunar regolith is such a good heat insulator that after a whole year the heated cylinders have not yet reached even halfway to the surface, and the cable temperature has climbed from 230 to 330 K, or from about negative 46 to +130 Fahrenheit. At this point we should be taking into account how the wire resistivity changes with increasing heat. It's not good.

So – I have to change my ideas about how to run the cable. They either need to be on the surface – and somehow shielded from surface temperature changes – or buried much more shallowly, which means they can't be under my roadways.

Here are the renderings:


30 Days
25 cm
240 250 260 270 280 290 300 310 320
60 Days
25 cm
240 250 260 270 280 290 300 310 320
90 Days
25 cm
240 250 260 270 280 290 300 310 320
120 Days
25 cm
240 250 260 270 280 290 300 310 320
150 Days
25 cm
240 250 260 270 280 290 300 310 320
180 Days
25 cm
240 250 260 270 280 290 300 310 320



How Do I Run It?

I'm including the code at the end of this paper, but it is also available at:

https://github.com/mgoulish/lunar-power-cable

To get it, do this: git clone https://github.com/mgoulish/lunar-power-cable

Then go into the directory it made, make a directory called './frames'make sure the Python file is executable, and execute it.

The frames subdirectory is where the program will put the images.

The Code

Here is the actual code, in case this is the only way you can get it.

```
#! /usr/bin/env python3

import math
import sys
from PIL import Image, ImageDraw, ImageFont, ImageOps
import matplotlib.pyplot as mpl
import numpy as np
from skimage.draw import (line, polygon, disk,
                          circle_perimeter,
                          rectangle,
                          ellipse, ellipse_perimeter,
                          bezier_curve)



#-------------------------------------------------
# Draw the concentric shell cross-sections at a
# particular time step.
#-------------------------------------------------
def draw ( n_time_step ) :
  # Define colors for the temperature breaks,
  # in Kelvins
  K_320   = (1.00, 1.00, 0.00)
  K_310   = (1.00, 0.75, 0.00)
  K_300   = (1.00, 0.53, 0.00)
  K_290   = (1.00, 0.25, 0.00)
  K_280   = (0.93, 0.00, 0.00)
  K_270   = (0.84, 0.00, 0.00)
  K_260   = (0.74, 0.00, 0.00)
  K_250   = (0.60, 0.00, 0.00)
  K_240   = (0.45, 0.00, 0.00)
  AMBIENT = (0.00, 0.00, 0.00)   # 230


  # Make the image.
  image_size = 1200
  center = image_size / 2
  img = np.zeros((image_size, image_size, 3), dtype=np.double)
  print ( f"drawing image for timestep {n_time_step}" )
  pixels_per_cm = 15

  # Iterate through all the concentric shells of regolith
  # around the power cable, choose the color for their temp
  # and draw them.
  # Note that we are iterating backwards here, from largest
  # to smallest shell, so that the smaller circles will get
  # drawn on top of the larger ones.
  for s in range(299, -1, -1) :
    temp = shell_temps[s]
    if temp <= ambient_temperature :
```

```
    continue
  if temp   >= 320 :
    color = K_320
  elif temp >= 310 :
    color = K_310
  elif temp >= 300 :
    color = K_300
  elif temp >= 290 :
    color = K_290
  elif temp >= 280 :
    color = K_280

  elif temp >= 270 :
    color = K_270
  elif temp >= 260 :
    color = K_260
  elif temp >= 250 :
    color = K_250
  elif temp >= 240 :
    color = K_240
  else :
    color = AMBIENT
  # 's' is the shell number, which is the same as its radius,
  # in centimeters.
  r = (s+1) * pixels_per_cm
  rr, cc = disk((center, center), r, shape=img.shape)
  img[rr, cc, :] = color

# Draw the 25 cm line ----------------------------
# i.e. 25 cm above the cable
y = int(center) - 15 * 25
rr, cc = line ( y, 0, y, image_size-1)
img[rr, cc] = ( 1.0, 1.0, 1.0 )
# Clear matplotlib figure from previous image.
# Or we will get new text superimposed on old text.
mpl.clf()
mpl.figure(figsize=(4,4))

# Draw the temperature key -----------------------------
rect_y      = 1100
rect_x      = 50
rect_width  = 120
rect_height = 60
temp = 240
for color in (K_240,K_250,K_260,K_270,K_280,K_290,K_300,K_310,K_320):
  rect_start  = (rect_y, rect_x)
  rect_extent = (rect_height, rect_width)
  rr, cc = rectangle(rect_start, extent=rect_extent, shape=img.shape)
  img[rr, cc] = color
  mpl.text ( rect_x, rect_y - 5, str(temp), color=(1,1,1) )
  rect_x += rect_width
  temp += 10

# Let's display number of days.
seconds_per_day = 86400
timesteps_per_day = seconds_per_day / time_step
days = int(n_time_step / timesteps_per_day)
label = str(days) + " Days"
mpl.text ( 100, 100, label, color=(1,1,1) )
```

```
  # Label the 25 cm line
  mpl.text ( 900, y, "25 cm", color=(1,1,1) )

  # Save out the image
  filename = './frames/img_' + "{:05d}".format(n_time_step) + '.png'
  mpl.imshow ( img )
  mpl.axis ( 'off' )
  mpl.savefig ( filename, bbox_inches='tight', dpi=400 )



#============================================
# Main
#============================================


#--------------------------------------------
# Initial Conditions
#--------------------------------------------
sim_duration        = 36000 # In time-steps
time_step           = 900  # Size of each time step in seconds.
# This is a little over 1 year

regolith_density    = 1.8   # g/cm3
reg_thermal_conduct = 0.85  # Watts / ( cm * K)
reg_spec_heat       = 1.512 # J/(g * K)
aluminum_density    = 2.7   # g/cm3
cable_length        = 100   # cm
cable_diameter      = 2     # cm. This is a *big* cable.
cable_mass          = math.pi * (cable_diameter/2)**2 * cable_length * \
                      aluminum_density
ambient_temperature = 230   # 230 K ~= -46 F
heat_dissipation    = 1     # Watt, or Joule per second, in the cable
al_spec_heat        = 0.903 # Joules / (gram * K)
original_cable_temp = ambient_temperature
added_cable_joules  = 0
min_temp_delta      = 0.1

#--------------------------------------------
# find cable surface area
#--------------------------------------------
cable_area = cable_diameter * math.pi * 100
cable_area /= 10000  # Convert to square meters.
#print ( f"cable area is {cable_area} m^2" )


#-------------------------------------------------
# We will look at 1-cm-thick shells going outward
# from the cable. The cable is 2 m deep so let's
# look at 300 of them. (The top 100 shells will be
# truncated by the surface.)
# These arrays will track -- or at least store --
# all the quantities I care about for each shell.
#-------------------------------------------------
n_shells = 300
shell_temps  = [ambient_temperature] * n_shells
shell_energy = [0] * n_shells
shell_areas  = [0] * n_shells
```

```
shell_masses = [0] * n_shells


#-----------------------------------------------
# Let's pre-calc the masses and areas of all
# the shells. Not shell 0, though. That's the cable.
#-----------------------------------------------
shell_masses[0] = cable_mass
shell_areas [0] = cable_area
shell_energy[0] = cable_mass * ambient_temperature * al_spec_heat
for r in range(1, 300) : # r stands for radius, in cm
  shell_area      = math.pi * 2 * r * 100
  shell_areas[r]  = shell_area / 10000 # In m2
  shell_volume    = shell_area # It's 1 cm thick.
  shell_mass      = shell_volume * regolith_density # In grams
  shell_masses[r] = shell_mass
  shell_energy[r] = shell_masses[r] * ambient_temperature * reg_spec_heat


#-----------------------------------------------
# Once per step for the durtion of the sim,
# propagate heat energy from the cable outward.
#-----------------------------------------------
for step in range(0, sim_duration+1) :
  if 0 == (step % 1000) :
    print ( f"step {step}" )
  for s in range(0, 299) :
    # heat diff between this shell and next one out
    temp_diff = shell_temps[s] - shell_temps[s+1]
    if temp_diff < min_temp_delta :
      break
    # Thermal conductivity of regolith at depth,
    # which I take to mean thermal conductivity of
    # regolith compressed a little, is:
    #   8.5e–3 W m–1 K–1 at a depth of 1 m
    # So for a centimeter instead of a meter, that is:
    #   8.5e-1 W / ( cm * K )
    # It changes inversely with length. I think.
    # And I think that means per meter squared.
    # So first find how much energy we are losing
    # per square meter because of the temp diff.
    # The TC is Watts per K, so scale for diff
    heat_loss = reg_thermal_conduct * temp_diff
    # And this is per square meter of surface,
    # so now scale by this shell's surface area.
    heat_loss *= shell_areas[s]
    # And this heat loss is in Watts, so -- since
    # we are running this sim in N-second time steps --
    # we multiply by N to get Joules.
    heat_loss *= time_step
    # So -- remove this quantity of heat energy, in Joules,
    # from this shell and give it to the next one!
    shell_energy[s]   -= heat_loss
    shell_energy[s+1] += heat_loss

    # Calc new temps for these two shells
    # This shell
    joules_per_gram = shell_energy[s] / shell_masses[s]
    kelvins = joules_per_gram / reg_spec_heat
```

```
  shell_temps[s] = kelvins
  # The next shell
  joules_per_gram = shell_energy[s+1] / shell_masses[s+1]
  kelvins = joules_per_gram / reg_spec_heat
  shell_temps[s+1] = kelvins
  # end of heat propagation loop --------------

#-----------------------------------------------
# Now that we have propagated as much energy
# as we can outward, add new energy to the
# cable.   Shell 0 is the cable.
#-----------------------------------------------
shell_energy[0] += heat_dissipation * time_step
joules_per_gram = shell_energy[0] / shell_masses[0]
kelvins = joules_per_gram / al_spec_heat
shell_temps[0] = kelvins


#-------------------------------------------------
# Make a drawing every 2880 timesteps.
# 2880 * 15 minutes is 30 days.
#-------------------------------------------------
if step > 0 and 0 == (step % 2880) :
  draw ( step )
# end of sim loop ----------------------
```