DRAGONFLY

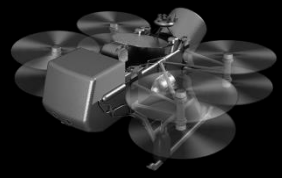
A relocatable lander to explore Titan's prebiotic chemistry and habitability

SPICE Thermal Vector Input for Numerical Heating Calculations

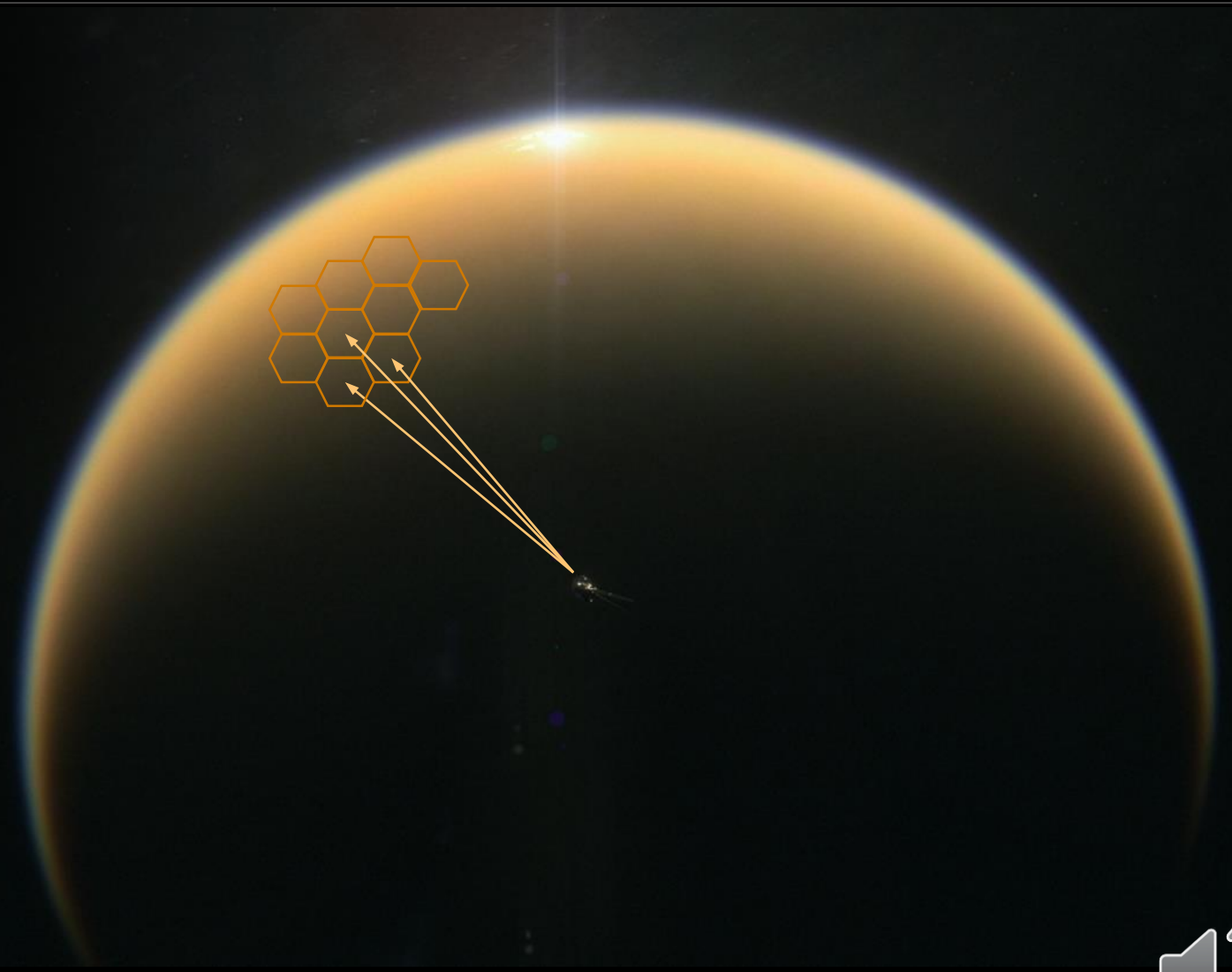
February 29, 2020

Allan Holtzman (Dragonfly Thermal Lead)

Requisite Geometry for Heating Calculations



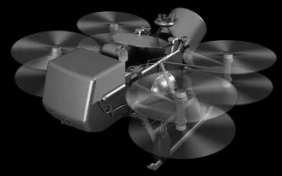
- Heat rate calculations for spacecraft are typically performed using a commercial code, such as Thermal Desktop
  - In-house software can also be used : Numerical Method to Calculate Spacecraft Environmental Heating From Celestial Bodies With Non-uniform Surface Properties and Temperatures*
- Either way, geometry representing the S/C in relation to the planet is required to perform the heating calculations
- One way to get this information is by using SPICE …



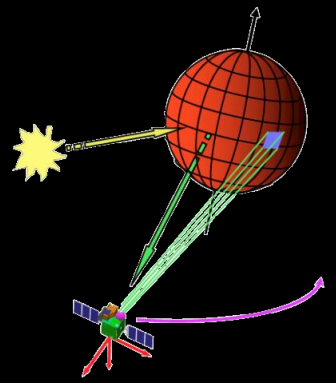
DRAGONFLY

* https://nescacademy.nasa.gov/video/659670f6b7cc47b0921fb7f7e35c94961d/

SPICE Overview



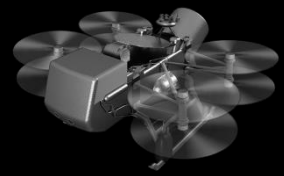
- *No, I'm not referring to Dune*
- SPICE is an information system built by the Navigation and Ancillary Information Facility (NAIF), acting under the direction of NASA's Planetary Science Division
- The acronym refers to logical components in the system (Spacecraft, Planets, Instruments, C-matrix, Events)
- SPICE can be used to extract geometric information for spacecraft and celestial bodies



- Positions and velocities of planets, satellites, comets, asteroids, and spacecraft
- Size, shape and orientation of planets, satellites, comets, and asteroids
- Orientation of a spacecraft and its various moving structures

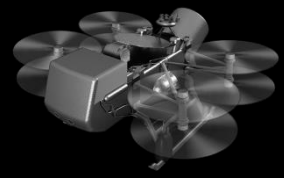
https://naif.jpl.nasa.gov/naif/geometric_parameters.png

Accessing SPICE

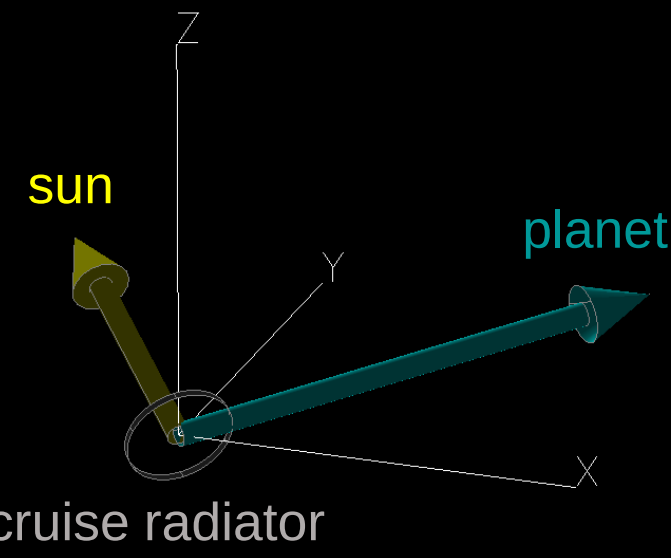


- SPICE functions can be called using C, FORTRAN, Matlab (very popular), and others
  - Third party interfaces exist for other languages, such as Python
  - I have used most of these methods, but my preference has gravitated to C (this presentation assumes a Unix operating system and the C language package installed in /naif/cspice)
- Basic steps for generating geometric information:
  - Install the SPICE libraries on your computer: https://naif.jpl.nasa.gov/naif/toolkit.html
  - Get SPICE kernels from your Mission Design and/or GNC folks (these are standard products that should already exist)
    - Kernel files typically contain location information for the relevant spacecraft and planets over the appropriate time period
    - Spacecraft orientation is generally a separate kernel file provided by GNC or other discipline, which may or may not be needed depending on the nature of your calculations
  - Examine the kernel files to ensure you have the appropriate information
    - Use the /naif/cspice/exe/brief executable to show the summary of a bsp file (shows spacecraft identifier, if it exists)
    - Alternatively, use the SPICE “spkobj” subroutine within the code
  - Use a code or script to access SPICE functions
    - Load the appropriate kernels (order could matter, last data overrides earlier)
    - Extract the geometry you need over the time period of interest
    - To compile on a Unix machine: gcc -I /naif/cspice/include -o yourcode yourcode.c /naif/cspice/lib/cspice.a -lm
  - Use this data as input to your heating calculations

Example: *Dragonfly* Venus Flyby

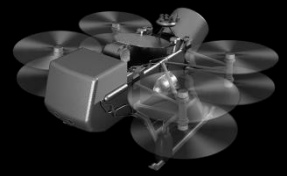


- Problem statement: Determine the approximate temperature increase of the *Dragonfly* spacecraft during a Venus flyby
  - S/C orientation is just spin-stabilized anti-sun, so getting orientation information from SPICE is not required (only location)
  - The *Dragonfly* cruise stage utilizes a fluid loop, which ties most components, including its MMRTG, thermally to the external radiator, making a simple model sufficient for estimating the impact of Venus flyby heating (except for external, lightweight S/C components that are not on the fluid loop, for which a more detailed model is required)
- Solution: Use a simple Thermal Desktop model to solve for the transient temperature response
  - Start with a steady-state no-Venus condition, with the balance of MMRTG heat that makes it into the fluid loop, with S/C spinning anti-sun
  - Apply transient heat loads by setting up a vector input for the “orbit” heating with spin in Thermal Desktop
  - Vector input format is in the S/C body frame, and includes the ratio of the S/C location divided by the planet radius
  - Note the S/C orientation is anti-sun, which prevents any direct solar input to the cruise stage radiator



Vector orbit input:
t, V_{sun}, V_{planet}, r_{ratio}

Example: C Code for SPICE Extraction



```
#include <stdio.h>
#include "SpiceUsr.h"
#include <math.h>
int main () {
  const double CONST_AU = 149597870.61368887; // in km
  const double VENUS_RADIUS = 6051.8; // in km
  double rratio; // ratio of the S/C-planet center distance divided by the planet radius
  SpiceDouble tbeg;
  SpiceDouble tend;
  SpiceChar scid [41] = "-6200005"; // identifier for Dragonfly
  SpiceDouble ven[3]; // Venus vector from point of view of the S/C
  SpiceDouble lt;
  SpiceDouble sun[3]; // Sun vector from point of view of the S/C

  furnsh_c("inputFromMissionDesign.bsp"); // Load all input files from Mission Design, bsp, tls, tpc
  . . .
  str2et_c("15 Apr 2027 00:00:00.000", &tbeg); // Convert time strings to elapsed time
  str2et_c("15 Apr 2027 05:00:00.000", &tend); //  from the J2000 epoch

  printf("t\tsunx\tsuny\tsunz\tplanx\tplany\tplanz\trratio\n");
  SpiceDouble tcur = tbeg;
  while(tcur<tend) {
     spkpos_c("299", tcur, "j2000", "none", scid, ven, &lt); // Venus barycenter is id 299
     spkpos_c("sun",   tcur, "j2000", "none", scid, sun,  &lt);
     sun[0]=sun[0]/CONST_AU; sun[1]=sun[1]/CONST_AU; sun[2]=sun[2]/CONST_AU; // Sun vector is in AU
     rratio = pow(pow(ven[0],2)+pow(ven[1],2)+pow(ven[2],2),0.5)/VENUS_RADIUS;
     printf("%11.1f\t%11.6f\t%11.6f\t%11.6f\t%11.3f\t%11.3f\t%11.3f\t%11.6f\n", tcur-tbeg,sun[0],sun[1],sun[2],ven[0],ven[1],ven[2],rratio);
     tcur=tcur+60.0; // output every minute
  }
  kclear_c ();
  return(0);
}
```

Include files

Parameters

Needed by Thermal Desktop

Get S/C identifier from Mission Design (or examine their files), or use cspice_spkobj to return the identifier

This is where you load the input files, can load separately or specify a metakernel file
Locations of S/C and planets, moons, etc. can be input

Calculations use an elapsed time from an epoch

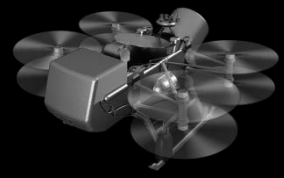
Get the planet vector in the S/C frame (barycenter by “99” suffix)

Get the sun vector in the S/C frame, convert from km to AU (needed by Thermal Desktop)

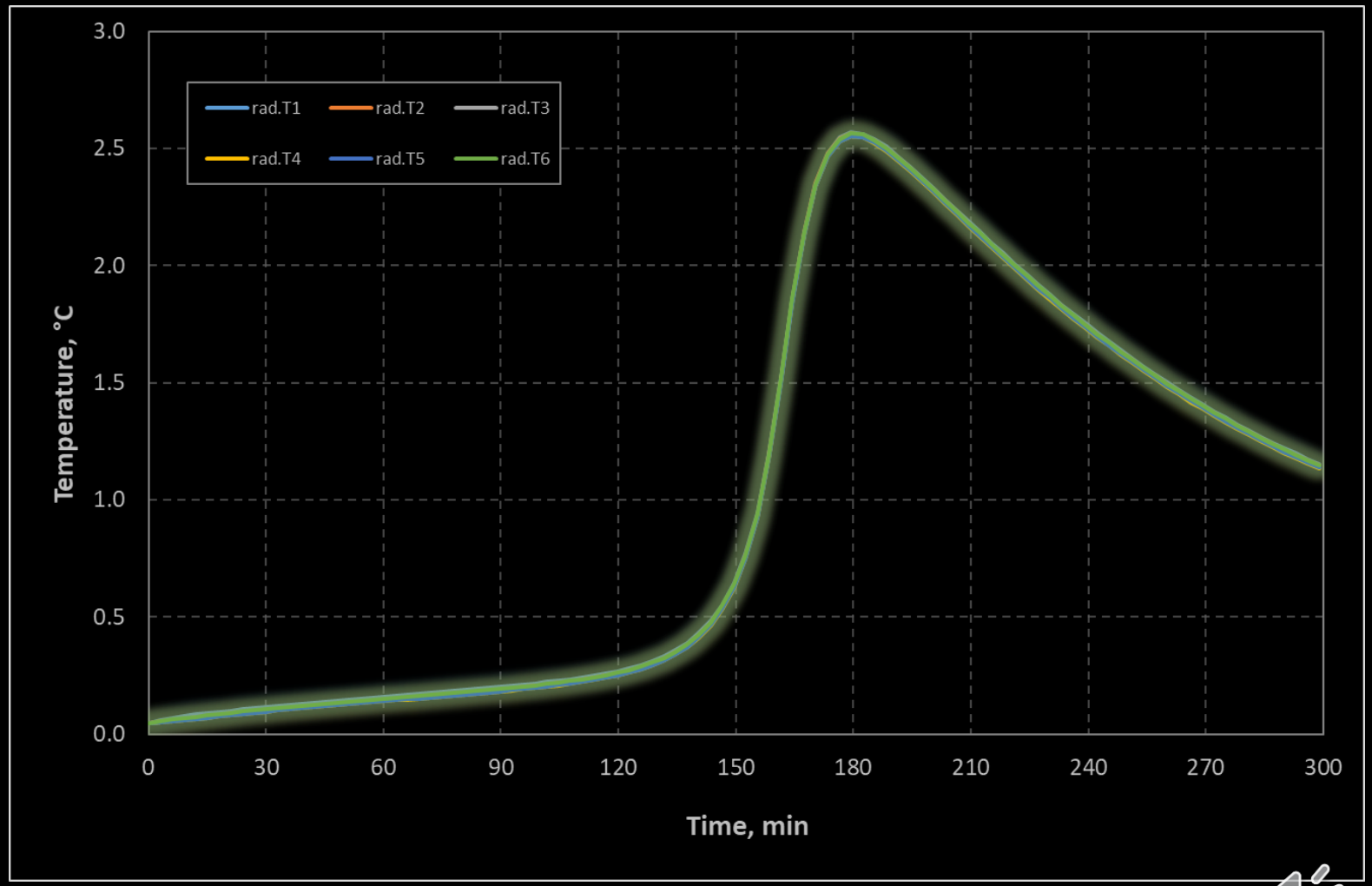
Output the time, V_{sun}, V_{planet}, r_{ratio}

DRAGONFLY

Example: *Dragonfly* Venus Flyby Results

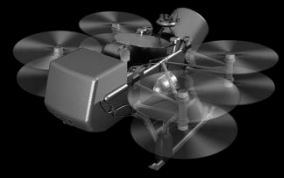


- Using the vector output from the code on the previous slide, temperatures are obtained from a simple Thermal Desktop model of the cruise radiator with an approximate system thermal capacitance
- Steady-state, anti-sun, pre-Venus radiator temperatures are around 0 °C, due to the heat load from the MMRTG
- Peak temperature for the *Dragonfly* radiator due to this Venus flyby is less than 3 °C
  - This is due to the relatively short heating pulse from Venus and high thermal capacitance of the system, thanks to the pumped fluid loop tying all the spacecraft components together



DRAGONFLY

Conclusions



- Using SPICE to extract the basic information needed for a thermal heating calculation is fairly simple
  - As evidenced by the brevity of this presentation, and the example C code
- The SPICE tool is free to use, and if you use gcc, so is the compiler
- Code complexity goes up slightly if you need to account for S/C orientation
  - Need to load an extra kernel file, call one more subroutine, and use the C-matrix
- SPICE kernel files likely already exist for your mission, or can be easily generated
- Output from SPICE can be used for a variety of purposes
  - Drive Thermal Desktop, or in-house software, for heating calculations
  - Process S/C->planet distances to determine where flybys occur, and their nature
  - Determine if a solar array is eclipsed by any solar system body
    - I did this for Europa mission, including Jupiter, Earth, Venus, Jovian moons, etc., predicting full and partial eclipses

The sky is the limit!