

License

Copyright © 2004, Robert Ross & Tony de Souza-Daw. All rights reserved.

Redistribution and use in any forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions must retain the above copyright notice, this list of conditions and the following disclaimer.

- * Neither the name of Robert Ross nor Tony de Souza-Daw nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mathematics Algorithms

ADAO Algorithms

Ozone: DONE(ROBERT 14/9/2004)

Theory: The ozone levels are measured in parts per million (ppm). Also see www.ozonelab.com/articles/007.html

Assumptions:

1. Ozone is completely independent
2. Laboratories does not effect it

Ozone Algorithms:

For each section of the ISS

Ozone[0] = random(0.01 to 0.1) //initialize

Ozone[n] = Ozone[n - 1] + random(-0.03 to 0.03)

Check that Ozone is not negative and not above 0.2ppm otherwise clip it.

Magnetosphere Strength: DONE(ROBERT 14/9/2004)

Theory:

In a LEO the magnetosphere strength ranges from 0.037 to 0.35 Gauss.

Assumptions:

1. The magnetosphere is independent

Magnetosphere Algorithm

Mag[0] = random(0.037 to 0.35) //initialize to a normal value

Mag[n] = Mag[n-1] + random(-0.015 to 0.015)

Check that Mag is not outside the range 0.037 and 0.35 otherwise clip it.

Ionising Radiation Levels: DONE(ROBERT 16/9/2004)

Ionising Algorithm:

Normal Conditions:

$r = \text{random}(0 \text{ to } 9)$

$\text{RADLevels} = 0.65 \pm r$

Solar Flare Conditions: A rise by 0.1 RAD.

$\text{RADLevels} = \text{RADLevels} + (\text{timer})^2 / 144000$

where timer is in seconds

Pseudocode:

Function determineSolarFlare

Invoke: Every 10 minutes

If (flare = false) then

```
{
    If (random(0 to 9) == 0) then
    { //10% chance of occurring every 10 minutes
        flare = true; // 2 minutes before hitting the ISS
    }
}
```

Function RADmonitor

Invoke: Every second

If (flare = true) then

```
{
    //initialise to zero
    flareTimer = flareTimer + 1; //variable that counts until the solar flare hits.
    If (flareTimer > 40)
    { //solar flare has hit (worst is over)
        r = random (0 to 3)/100;
        RADLevels = RADLevels - r; //reduce linearly
        If (RADLevels < 0.65) then
        { // backed to normal conditions, reset solar flares variables
            Flare = false;
            flareTimer = 0;
        }
    }
}
Else
{
    RADLevels = RADLevels + (flareTimer)2/16000;
    // during a solar flares the RAD levels increase by 0.1 quadratically.
    // warn at 0.7 RADs
}
Else
{
    --normal conditions
    r = random (0 to 9)
    RADLevels = 0.650 + r; //does not need to depend on previous value, increment
    are too small.
}
```

Weather

Theory:

This screen is mainly to watch for cyclones. Cyclones are dependent on several variables, although the wind velocity and the humidity have a significant impact. All other variables such as storm activities are ignored. High values (greater than 85%) of humidity can indicate cyclone conditions. The wind velocity will reflect the current situation. High winds (greater than 90kmh^{-1}) will indicate a cyclone. Landing the space shuttle in cyclone conditions region would not be desirable.

Algorithm: Wind Velocity DONE(ROBERT 14/9/2004)

Initialise

drift = 0

windVelocity = random number (between 0 and 15)

Invoke: Every Minute

windVelocity = windVelocity +

NormalizeRandomGenerator($\mu = 1 + \text{drift}$, $\sigma = 1$)

drift = drift + random number (between -0.2 and 0.2)

Algorithm: Humidity DONE(ROBERT 18/9/2004)

Initialise

For each tropical region

Humidity = 70%

Invoke: Every minute

For each tropical region

If (temperature > 26) then

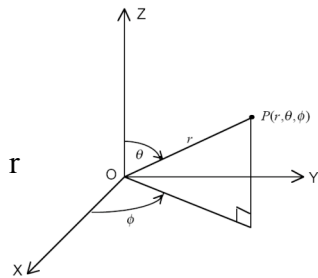
humidity = humidity + NormalRandomGenerator($\mu = 0$, $\sigma = 5$)

end if

Ocean Temperatures DONE (ROBERT: 15/92004)

Theory:

The ocean temperature depends on the longitude θ and latitude ϕ location in question. This determines seasons, night and day temperatures.



θ – determines the longitude effect

Φ – determines the latitude effect

r – radius of the earth (6378km)

Initial temperatures sets the base temperature

Assumptions:

1. Summer in the northern Hemisphere
2. When the Mission Clock starts, it is sunrise in the base location.

Algorithm:

A sinusoidal increase/decrease periodically to map the temperatures around the world. There are 1440 minutes in a 24hours period.

$r = \text{random}(0 \text{ to } 3);$

$\text{temperature} = \text{temperature} + \text{int}(2r\sin(2\pi t/1440 + \theta + \Phi))$

where t is in minutes

θ – radians

Φ – radians

Parameters for the different locations:

Northern Pacific Ocean: $\theta = 0.000, \Phi = 0.000$ (reference point)

Southern Pacific Ocean: $\theta = 1.042, \Phi = 1.014$

Indian Ocean: $\theta = 2.552, \Phi = 0.437$

Northern Atlantic Ocean: $\theta = 4.642, \Phi = 0.200$

Southern Atlantic Ocean: $\theta = 4.87, \Phi = 0.943$

PHALCON Algorithms

The PHALCON Officer is responsible for maintaining power to the space station.

The equation for the power distribution is given below;-

Total power = Solar Power(120kW)

+ Auxiliary Power (30kW with exponential decay)

+ Hydrogen Fuel Cell (10V, 10000A => 100kW)

- Data Interface (38kW)

- Communication Interface (60kW)

- Antenna electrical power (10A, 500V)

- 11 x Power racks 11x(20A, 100V)

- Device Controls (5kW)

- Recharge of Auxiliary Power (linear recharge rate)

Total Generated Power

Total Absorbed Power

Legend

- Fluxuating variables

- Are constant extrapolated assumption

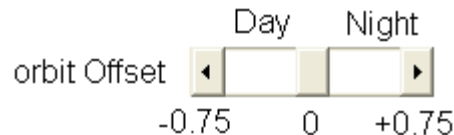
- Variable extrapolated assumption

Solar Power Panels

Theory:

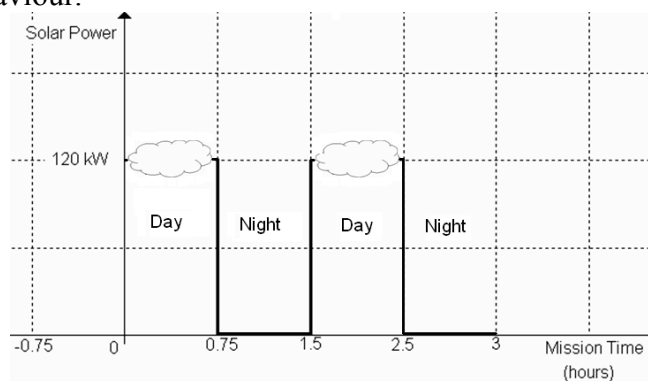
The ISS orbits the earth at a period of approximately 1.5 hours. Hence the solar panels will be completely de-energized after 0.75 hours of daylight. The mission time is between zero to 2 hours. During daylight hours the solar panels has a mean of 120kW, letting 68% of the solar power lie between 100 to 130kW, therefore the standard deviation can be approximately by $\sigma = 15\text{kW}$.

In the Minor Project the orbit offset can be changed. ie The mission can start at any time during the night or day.



In the Major Project the orbit Offset is zero.

The generated power would be as below. Where the day section represent the uncertainty of the random behaviour.



Algorithm:

Invoke: Every minute

If $(0 < (\text{MissionTime} - \text{OrbitOffset}) < 0.75)$ AND

$(1.5 < (\text{MissionTime} - \text{OrbitOffset}) < 2.25)$ then

$\text{solarPower} = \text{NormalizeRandomGenerator}(\mu = 120, \sigma = 10)$ //day time

else

$\text{solarPower} = 0$

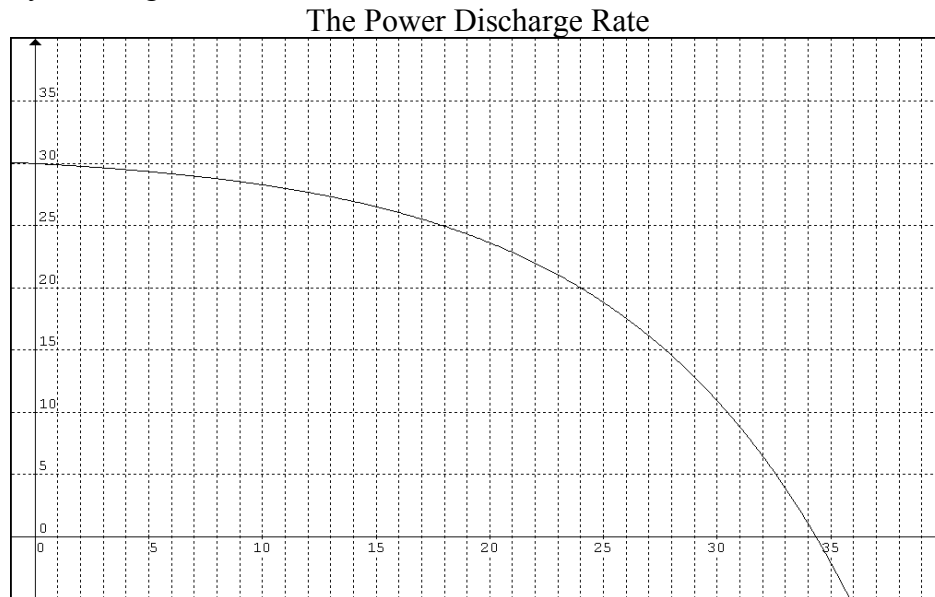
end if

Auxiliary Power

Theory:

The auxiliary power dissipates energy at an exponential rate and recharges linearly, based on the excess power.

When the Auxiliary Power is fully recharged, the power available is 30kW (extrapolated assumption). This power has a time constant of 10 minutes, thus it is considered to be completely discharged at 30 minutes.



Pseudocode:

When the Auxiliary Power is turn on (ie the button is triggered)

$t = 0$

$A_{power} = \text{AuxiliaryPower}$ //store the current value

Invoke: Every minute

If (AuxiliaryPower is On) then

$t = t + 1$

$\text{AuxiliaryPower} = -A_{power} \times e^{t/10} + 31$

Recharged = False

Else

// AuxiliaryPower is off

If (totalGeneratedPower > totalAbsorbPower) and (Recharge = False) then

$\text{AuxiliaryPower} = \text{AuxiliaryPower} + (\text{totalGeneratedPower} - \text{totalAbsorbPower})$

If (AuxiliaryPower > 30kW) then

$\text{AuxiliaryPower} = 30\text{kW}$

Recharged = True

End if

End if

End if

Experimental Power Rack

Theory:

The simulated ISS has eleven experimental power racks. Each power rack requires 100V (fixed) and 20A. The current drawn can fluctuate with a warning when the current falls below 15A. This can be model with a normal random generator.

Algorithm:

Initialize

For (n is 0 to 10)

RackCurrent(n) = 0 //racks are off.

Every 10 minutes

For (n is 0 to 10)

RackCurrent(n) = RackCurrent(n) – NormalRandomGenerator($\mu = 0$, $\sigma = 2$)

Function GetPower

Return $100 \sum_{n=0}^{10} RackCurrent(n)$

Additional Theory:

The temperature of each rack rises when the rack is on and falls when the rack is off. The rack temperature is regulated with an air-cooling unit. This can be increase or decrease resulting in more or less power consume and temperature.

The cooling unit changes the air coefficient (variable: Air) which effects the rate of the temperature increase or decrease. Hence the results are not immediately noticed, it takes at least one minute to see the effect. Although the current consumed is increased when the air cooling unit regulator is increased and decreased when the air cooling unit regulator is decreased. The current drawn updates immediately.

Algorithm:

Initialize

For (n is 0 to 10)

RackTemperate(n) = normalizeRandomGenerator(22.5, 1)

Air(n) = 0.06

Invoke: Every minute

For (n is 0 to 10)

If (Rack(n) is on) then

RackTemperate(n) = RackTemperate(n) + Air(n)

Else

//rack is off

RackTemperate(n) = RackTemperate(n) – Air(n)

End if

Temperature Control of the Air-Cooling Unit

On Manual Increase of Rack n

$$\text{RackCurrent}(n) = \text{RackCurrent}(n) + 0.5$$

$$\text{Air}(n) = \text{Air}(n) - 0.01$$

On Manual decrease of Rack n

$$\text{RackCurrent}(n) = \text{RackCurrent}(n) - 0.5$$

$$\text{Air}(n) = \text{Air}(n) + 0.01$$

Communication Antenna:

Theory:

The communication and data interface can be turned on and off, but both is critical to the mission success so must be turned off as a last resort. The signal strength is dependant on how much power is absorbed by the antenna and whether or not there is a solar flare. The electrical power supply is subjective to random fluctuation.

Algorithm:

When Data Interface is switched on

$$\text{AntennaPower} = \text{AntennaPower} - 38\text{kW}$$

When Data Interface is switched off

$$\text{AntennaPower} = \text{AntennaPower} + 38\text{kW}$$

When communication is switched of

$$\text{AntennaPower} = \text{AntennaPower} - 60\text{kW}$$

When communication is switched off

$$\text{AntennaPower} = \text{AntennaPower} + 60\text{kW}$$

Electrical Power

The mean current is 10A with a constant voltage of 500V. The current has a normal variation between 8A and 12A. Hence the standard deviation can be 2A.

$$\text{CommunicationCurrent} = \text{NormalizeRandomGenerator}(\mu = 10, \sigma = 2)$$

Function GetCommunicationPower (kW)

$$\text{Return AntennaPower} + \text{CommunicationCurrent} \times 0.5$$

Signal Strength

The mean is 95% with a tolerance of $\pm 5\%$ and a critical value of 70%.

If (flare = true) //during a solar flare

$$\text{SignalStrength} = \text{NormalizeRandomGenerator}(\mu = 95, \sigma = 5) - 30 * \text{flareTimer}^2 / 14400$$

Else

$$\text{SignalStrength} = \text{NormalizeRandomGenerator}(\mu = 95, \sigma = 5)$$

End if

Fuel Cell: Done Tony

Theory:

The fuel cells contribute approximately 5kW of power to the system. The current has a mean of 10V and has a standard deviation of 2V. This can be simulated with a normal random generator. The Fuel Level normally decreases 5% per an hour. Although if the current increases or decreases it will effect the fuel consumption rate on a linear relationship. An anomaly such as freezing of the fuel cell or overheating, in which case a bank of fuel cells becomes unavailable. This in turn, increases the current to maintain the original supply of power from the fuel cells.

Algorithm:

Initialize

fuelLevel = 100

fuelVoltage = 10

meanFuelVoltage = 10

Every Minute

If (FuelLevel > 0) then

 FuelCellVoltage = NormalRandomGenerator(μ = meanFuelVoltage, σ = 2)

 Leak = FuelCellVoltage – meanFuelVoltage

 FuelLevel = FuelLevel -5/60 – fmax(0, leak/80)

 If (FuelLevel < 0) then

 FuelLevel = 0

 End if

End if

Every Minute

BankLoss = Random(0 to 100)

If (BankLoss = 0) then

 fuelLevel = fuelLevel – 0.3*fuelLevel //in the event of an anomaly

 meanVoltage = meanVoltage – 3 //30% of the fuel is loss

end if

Function Get FuelCellPower

 Return FuelCellVoltage x 500

Device Controls:

Theory:

The heating and thermal operations are simulated in the same method. Both have a maximum of 1kW and has incremental of 0.1kW. When all lights are on, the power consumption is 3kW. However, when only the emergency lights are on the power required is 0.5kW.

Algorithm:

Function Get DevicePower

If (lights are all on)

Return HeatingPower + ThermalPower + 3kW

Else

Return HeatingPower + ThermalPower + 0.5kW

End if

Mathematical Utilities

Normalize Random Generator

Theory:

A normal probability function: $P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Where σ - standard deviation

μ - mean

x - input

$P(x) \in [0,1]$

Re-arranging the normal probability function

$$x = \mu + \sqrt{-2\sigma^2 \ln|2\pi\sigma P(x)|}, \text{ where } p(x) \in (0, \frac{1}{\sigma\sqrt{2\pi}})$$

Function NormalizeRandomGenerator:

While (L == 0)

L = rand() * int(10000 * $\sigma\sqrt{2\pi}$) / 10000; //linear random number between 1 and $\sigma\sqrt{2\pi}$.
//using discrete intervals of 1/10000.

End while

$p(x) = \text{double}(L / \sigma\sqrt{2\pi}); //p(x) \in [0,1), L \neq 0$

$p(x) = p(x) / \sigma\sqrt{2\pi}$ //now in the range of the normal probability

$$n = \mu \pm \sqrt{-2\sigma^2 \ln|2\pi\sigma p(x)|}$$

return n

Linear Random Generator

Theory:

With a minimum value and a maximum the overall result is shifted.

Algorithm:

$r = \text{rand}(0 \text{ to } (\text{maximum} - \text{minimum}))$

Return $r + \text{minimum}$