How Long is a Piece of Christmas Lighting

T. Mark Ellison*

2012-11-30

The goal of this short paper[1] is to detail how long a piece of Christmas lighting you need in order to decorate a Christmas tree.

1 The Model

The model of the Christmas tree and the lighting is very simple. It rests on 8 assumptions and notations. For clarity, I'll use capital letters for constants or parameters to the problem, and lower case letters for variables.

- The part of the Christmas tree you want to put lights onto is a cone.
- The lights will be strung on the outside of the Christmas tree.
- The height of the part of the tree that you want to decorate is H.
- The radius of the tree at the bottom of the part you want to decorate is R.
- The lights will be strung around the tree in such a way that at each circuit of the tree, the height of the line increases by the same amount (even though the distance from the centre line of the tree will vary).
- You want N circuits of the tree.
- The path of the wire can be parameterised by the number of radians wound around the centre of tree as the wire goes from the top of the cone to the bottom, and we'll usually symbolise this parameter by θ. For N windings around the tree, θ varies from 0 at the top of the tree to $2\pi N$ at the bottom.

*School of Psychology, The University of Western Australia. May be contacted at mark.ellison@uwa.edu.au.

- The length of lighting you need to decorate the whole tree will be L.

This is enough information to work out the answer. Let's get calculating.

Because it's what I'm used to, I'll use metric in the examples. But only one measure is involved, namely distance, so you can just as easily use feet, yards, cords or furlongs.

2 The Integral

The trick to working out the integral equation is to look at the infinitesimal case. So imagine the wire curving around the tree, and at some point in its curve, you look at just a tiny extent of that wire. That tiny extent of wire corresponds to a tiny change in the radians used to parameterise the path the wire is taking. Let's write that tiny change as $d\theta$.

2.1 Infinitesimal Change in Height

Here's the first step in the calculation. We'll work out how the vertical position changes with the small change in θ.

The total height of the cone part of the tree is h. The number of windings around the tree is N, giving the total radians traversed from bottom to top as $2\pi N$. We know that height changes linearly (so that the vertical gap between each winding is a constant), so the height gap per winding is H/N. We divide this further by 2π to get the vertical distance travelled per radian of the parameter. This equation appears as (1).

$$dz \quad = \quad -\frac{H}{2\pi N} d\theta. \tag{1}$$

The sign of the fraction is negative because we defines $\theta = 0$ as the *top* of the tree.

2.2 Infinitesimal Change in Radius from Centreline

As θ increases, the path moves progressively down the cone. But cones being cones, moving down the cone means moving further away from the vertical centreline of the cone. By the definition of conic structure, this movement away is linear in the drop in height. We develop this relationship by defining dr, the change in the radius of the horizontal cross section of the tree, first in terms of dz, and then later in terms of $d\theta$.

Firstly, the radius from the centreline of a cone at height position z will be whatever fraction of the total height that z is, multiplied by the radius of the base. In other words, r is equal to zR/H, or in differential form $dr = R\ dz/H$. In equation (2), this result is combined with that of (1).

$$
\begin{aligned}
dr &= -\frac{R}{H}dz \\
&= \frac{R}{2\pi N}d\theta
\end{aligned} \tag{2}
$$

2.3 Infinitesimal Change in Yaw

The third dimension in our coordinate system is rotation around the vertical centreline. In aeronautics, this type of rotation is called yaw. The coordinate is none other than our old friend θ, and so expressing this rotation in terms of our parameter θ is rather tautologous.

$$
d\theta = d\theta \text{ duh!} \tag{3}
$$

2.4 Distance

To calculate the length of wire corresponding to a change $d\theta$ in the wire parameter, we calculate the distance using the squares of the three differential terms: dz, dr and $d\theta$. But because the space defined in these coordinates is not Euclidean, we need to draw on a non-identity metric tensor with diagonal $(1, 1, r^2)$. (If you don't know about metric tensors, the discussion in Wikipedia is quite detailed, and can tell you a lot, but you may need to do some other background reading to make sense of it.)

We'll use ds (a notation borrowed from relativistic physics) to represent the small change in total line length that corresponds to the change in coordinates $(dz, dr, d\theta)$.

$$
\begin{aligned}
ds^2 &= (dz^2 + dr^2 + r^2 d\theta^2) \text{ or in other words,} \\
ds &= \sqrt{dz^2 + dr^2 + r^2 d\theta^2}.
\end{aligned} \tag{4}
$$

3 The Integral

So we now have an expression (4) for how the length of wire changes as the coordinates change. We also know how each of these coordinates change in terms of our parameter θ (1,2 and 3).

To calculate the entire length, we must sum the length of bracketed wire over all points between the top and bottom of pyramid. This total distance is notated as L. Over this distance, the lines goes around our tree N times, so our parameter θ (remember that it is measured in radians) varies from 0 to $2\pi N$.

First, we fix the end point of our integral (5). Then we replace the coordinates changes dz, dr by their expression in terms of $d\theta$, the change in our parameter (6). Finally, we extract the $d\theta$ term from the summation and square root (7).

$$
\begin{aligned}
L &= \int_0^{\theta=2\pi N} \sqrt{dz^2 + dr^2 + r^2 d\theta^2}, && (5) \\
&= \int_0^{2\pi N} \sqrt{(\frac{H}{2\pi N})^2 d\theta^2 + (\frac{R}{2\pi N})^2 d\theta^2 + (\frac{R\theta}{2\pi N})^2 d\theta^2}, && (6) \\
&= \int_0^{2\pi N} \sqrt{(\frac{H}{2\pi N})^2 + (\frac{R}{2\pi N})^2 + (\frac{R\theta}{2\pi N})^2} d\theta. && (7)
\end{aligned}
$$

3.1 Solving the Integral

Many practical problems that end up being described by equations such as (7). Their solution depends on transforming them into the form of standard integrals whose solution is known. Here the first step is to extract a common multiplier outside the integral to transform the constant expression to 1, as shown in (8).

$$
\begin{aligned}
L &= \int_0^{2\pi N} \sqrt{(\frac{H}{2\pi N})^2 + (\frac{R}{2\pi N})^2 + (\frac{R\theta}{2\pi N})^2} d\theta, \\
&= \sqrt{\frac{H^2 + R^2}{4\pi^2 N^2}} \int_0^{2\pi N} \sqrt{1 + (\frac{R\theta}{H^2 + R^2})^2} d\theta. && (8)
\end{aligned}
$$

Now let's change the variable θ to a different variable ρ. We have two equations, first ρ in terms of θ (9) and then the reverse (10).

$$
\begin{aligned}
\rho &= \sqrt{\frac{R^2}{H^2 + R^2}} \theta && (9) \\
\theta &= \sqrt{\frac{H^2 + R^2}{R^2}} \rho. && (10)
\end{aligned}
$$

Substituting into (8), we have a new equation for L.

$$
\begin{aligned}
L &= \sqrt{\frac{H^2 + R^2}{4\pi^2 N^2}} \int_0^{\theta=2\pi N} \sqrt{1.0 + (\frac{b\theta}{H^2 + R^2})^2} d\theta, && (11) \\
&= \sqrt{\frac{H^2 + R^2}{4\pi^2 N^2}} \sqrt{\frac{H^2 + R^2}{R^2}} \int_0^{\rho=2\pi N \sqrt{\frac{R^2}{H^2+R^2}}} \sqrt{1.0 + \rho^2} d\rho. && (12)
\end{aligned}
$$

Equation (12) has the form $\int_0^x \sqrt{1.0 + \rho^2} d\rho$, which is a standard integral with a well-known solution. The solution in (13) comes from *http://planetmath.org/?method=js&id=10662&op=getobj&from=objects.*

$$\int_0^x \sqrt{1.0+\rho^2}d\rho \quad = \quad \frac{x}{2}\sqrt{x^2+1}+\frac{1}{2}\sinh^{-1}x. \tag{13}$$

3.2 The Boundary

We get the final value for the integral by substituting our upper limit for the integral, namely $x = 2\pi N\sqrt{\frac{R^2}{H^2+R^2}}$, into equation (13).

$$\int_0^x \sqrt{1.0+\rho^2}d\rho$$

$$= \pi N\sqrt{\frac{R^2}{H^2+R^2}}\sqrt{(4\pi^2N^2\frac{R^2}{H^2+R^2})+1}+\frac{1}{2}\sinh^{-1}2\pi N\sqrt{\frac{R^2}{H^2+R^2}}. \tag{14}$$

For convenience, we'll abbreviate this quantity to Z.

3.3 The Final Result

We obtain a complete expression for L by substituting (14) into (12).

$$\begin{aligned} L &= \sqrt{\frac{H^2+R^2}{4\pi^2N^2}}\sqrt{\frac{H^2+R^2}{R^2}}\int_0^{2\pi N\sqrt{\frac{R^2}{H^2+R^2}}}\sqrt{1.0+\rho^2}d\rho, \\ &= \sqrt{\frac{H^2+R^2}{4\pi^2N^2}}\sqrt{\frac{H^2+R^2}{R^2}}Z, \\ &= \frac{H^2+R^2}{2\pi NR}Z, \qquad (15) \\ &= \frac{H^2+R^2}{2\pi NR}(\pi N\sqrt{\frac{R^2}{H^2+R^2}}\sqrt{(4\pi^2N^2\frac{R^2}{H^2+R^2})+1}+ \\ &\quad \frac{1}{2}\sinh^{-1}2\pi N\sqrt{\frac{R^2}{H^2+R^2}}). \qquad (16) \end{aligned}$$

There we have it. Equation (15), or its full form (16), will tell you what length of lighting wire you need to wrap a Christmas tree of given height and radius.

Now we have a formula, we'll make the calculation process easier by implementing it in the scripting language PYTHON.

4 Python Code

```
#!/usr/bin/python
```

```python
# (c) 2012 T. Mark Ellison
# Released under the Australian creative commons licence
#   Attribution-NonCommercial-ShareAlike (CC BY-NC-SA)
# You can distribute, remix and build upon the work,
#  but only if it is for non-commercial purposes,
#  you credit the original creator/s (and any other
#  nominated parties) and you license your derivative
#  works under the same terms.
# http://creativecommons.org/licenses/by-nc-sa/3.0/au/legalcode

import math, sys

class ChristmasTreeLights:
 """
 This class is initialised with three arguments - ChristmasLightsLength(H,B,N)
   H is the height of the cone to be lit
   B is the radius of the cone's base
   N is the number of windings around the cone
 Command-line usage:
   ./ChristmasTreeLights.py H B N
 """
 def __init__(self, H,B,N):
  self.H                        = float(H)
  self.B                        = float(B)
  self.N                        = float(N)
  self.K                        = math.sqrt( self.H ** 2 + self.B ** 2 )
  self.L                        = self.B / self.K
  #
  self.computeZ()
  self.L                        = self.H ** 2 + self.B ** 2
  self.L                       /= 2 * math.pi * self.N * self.B
  self.L                       *= self.Z
 def computeZ(self):
  self.Z                        = math.pi * self.N * self.L
  self.Z                       *= math.sqrt( 4 * self.Z ** 2 + 1 )
  self.Z                       += 0.5 * math.asinh( 2 * math.pi * self.N * self.L )
 def __str__(self):
  return "%f" % (self.L)

print ChristmasTreeLights(sys.argv[1],sys.argv[2],sys.argv[3])
```

5 Some Numbers

5.1 Sanity Check

But before we get too excited about that result, let's try some sanity-check calculations.

If the base radius is very small, making the tree very thin, and we do very little winding around the tree, then the wire should basically be strung in a straight line from bottom to top of the stem. In other words, the length of lighting needed should approximate the height of cone part of the tree. This is verified with the following run of the program, where the height of the cone is 4 metres, but its base has a 1mm radius and we only do a thousandth of a revolution around the tree.

```
$ ./ChristmasTreeLights.py 4 0.001 0.001
4.000000
```

Now suppose we have a large radius to the tree's cone. If there is (almost) no movement helically around the tree, but purely a vertical movement, the length of the path is approximately the hypotenuse of the right-angled triangle formed by this base radius and the height. So if the base radius is 3, and the height is 4, we should expect to see a needed lighting length of 5. This is just what we do see.

```
$ ./ChristmasTreeLights.py 4 3 0.001
5.000012
```

To check that our formula is doing the right thing with the winding distance, let's work out the winding length needed for the bottom circuit when there are many circuits of the tree. This should approximate $\sqrt{(\frac{R}{N})^2 + (2\pi R)^2}$, the diagonal on the surface of a cylinder of radius R and height $1/N$ of H. If the whole tree has parameters H, R, N, then the tree minus the bottom winding will have parameters $\frac{N-1}{N}H, \frac{N-1}{N}R, N-1$. Let's take $H = 100$, $R = 1$ and $N = 100$ for simplicity ... we are using metric, after all!

```
$ ./ChristmasTreeLights.py 100 1 100
338.306448
$ ./ChristmasTreeLights.py 99 0.99 99
331.975197
```

The difference between these two values, ie 6.331251 should approximate $\sqrt{1 + (2\pi)^2}$, the distance that winding would have taken on a perfect cylinder of radius 1 and height 1.

```
$ python
Python 2.7.1 (r271:86832, Jul 31 2011, 19:30:53)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> math.sqrt(1+4*math.pi**2)
6.362265131567328
```

As it turns out, the approximation is quite good: 6.33 vs 6.36, ie a difference of around 0.5%. Try for yourself changing the 100s to 1000s, 99s to 999s, and 0.99 to 0.999: does the approximation get better?

The sanity checks are thus declared to be passed.

5.2 A Reasonable Tree

Now consider a reasonable house tree, that has a 2 metre cone, whose base is 0.5m in radius. Ok, with pot and all, it's a big house tree. We want 12 windings, to make it really light up, roughly one circuit every 17cm in height. That makes for 19.08 metres of lights, or one 20m light strand with nearly a metre left to make it to the wall socket or extension cord.

```
$ ./ChristmasTreeLights.py 2 0.5 12
19.080645
```

5.3 A Final Question

We have just assumed by fiat that the lights should wrap around the tree so that there is a constant vertical gap of H/N between consecutive windings. Here's a question. Is this path the shortest path that winds around the cone in N complete circuits from top to base? If so, how would you prove it is the shortest path?