From this exercise, the perimeter P_n is

$$P_n \quad = \quad n\, d_n \quad = \quad n \sqrt{\left(1 - \cos \frac{2\pi}{n}\right)/2} \quad .$$

We can implement this formula using Maple as follows.

```
Pn := proc (n)       # perimeter of regular n-gon
     local i;
     n * sqrt( 1/2*(1-cos(2*Pi/n)) );
end:
```

When we plot the absolute error of this new formula against Leibnitz' formula as in Figure 0.8, we see that it clearly converges more quickly, at the expense of many square root and cosine computations, which are in general more expensive than simple divisions.

We have another embarrassing problem to overcome: we are using the value of π to compute it! It is one thing to exploit the properties of π, but it is quite another to assume we have a value before we have computed it. Let us use the properties of π to modify our algorithm. Rather than computing P_n for arbitrary n, let us insist that n be a power of two. That is $n = 2^k$, for some $k > 1$.

Since so many n are now forbidden, let us instead write Q_k for the perimeter of a regular 2^k-gon inscribing the unit circle. In other words, $Q_k = P_{2^k}$. Similarly, let ω_k be the angle between two endpoints of an edge in the 2^k-gon (so $\omega_k = \theta_{2^k}$). Thus $Q_2 = P_4 = 2\sqrt{2}$ is perimeter of the inscribing square. Observe that

$$\omega_k \quad = \quad 2\omega_{k+1},$$

meaning that the angles between endpoints are halved. The double-angle formula for the cosine function is

$$\cos 2\alpha \quad = \quad 2\cos^2 \alpha - 1.$$

This allows us to write out a recurrence. Define $C_k = \cos \omega_k$. Then

$$C_k \quad = \quad \cos \omega_k \quad = \quad \cos \frac{2\pi}{2^k} \quad = \quad \cos \frac{\pi}{2^{k-1}}.$$

On the other hand,

$$C_{k-1} \quad = \quad \cos \frac{\pi}{2^{k-2}} \quad = \quad \cos\left(2 \cdot \frac{\pi}{2^{k-1}}\right).$$

Applying the double angle formula on the right-hand side implies that

$$C_{k-1} \quad = \quad 2C_k^2 \; - \; 1,$$

meaning that

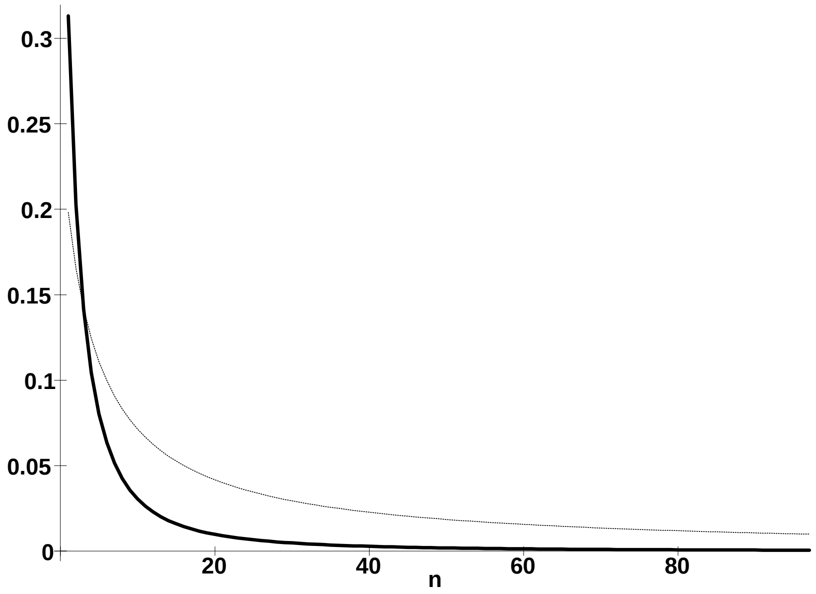

Figure 0.8. Comparison of absolute errors made by Leibnitz' rule (thin line) and perimeter computation (thick line).

$$C_k \;=\; \sqrt{(C_{k-1}+1)/2}\;.$$

This allows us to write an expression for the cosine of a small angle in terms of the cosine of an angle twice its size and a square root. We ''bottom out'' with C_2, which is simply $\cos(\pi/4) \;=\; \sqrt{2}/2$. Our expression for the perimeter of a 2^k-gon is therefore

$$Q_k \;=\; 2^k\sqrt{(1-C_k)/2}\;. \qquad (2)$$

Observe that we have avoided using the value of π at the expense of a recursive formula that creates a sequence of nested square roots. The Maple code to perform this computation is

```
perim := proc (n)
     local i;
     2^n*sqrt(1/2*(1-C(n)));       # recurrence avoids Pi
end:
```

```
C := proc (n)
    option remember;
    if   (n<=2) then 0
    else sqrt( 1/2*(C(n-1)+1) );
    fi;
end:
```

The **`option remember`** statement causes Maple to recall the result of previous computations of **`C(n)`**, which saves a great deal of time if a function is often invoked with the same values for arguments. Our derivation with respect to side length of a regular polygon is similar to that made by Archimedes more than 22 centuries ago. His recurrence is based directly on perimeter Q_k:

$$Q_{k+1} \quad = \quad 2^k \sqrt{2\left(1 - \sqrt{1-(Q_k/2^k)^2}\right)} \quad . \tag{3}$$

Kahaner *et al.* present an excellent sequence of exercises illustrating the pitfalls of numerically implementing this rule.[3] It turns out the same is true of our rule (see the exercises). However they demonstrate how this formula can be successively refined into an efficient, robust procedure. Press *et al.* describe how one might compute the value of π to very high precision (e.g., several thousand digits).[4]

Exercise 0.7. We have seen that a plot provides a visual evidence of the rate at which an approximation converges. What would be a good quantitative measure of the convergence rate? What if the value to which a series or sequence converges is not known?

Exercise 0.8. Using Maple show that Archimedes rule, Eq. 3, agrees with our rule, Eq. 2. Test your results by subtracting one rule from the other. In two separate small procedures, use **`simplify`** in one and **`evalf`** in the other to see if the result for arbitrary k is always zero. Are there any differences between the two procedures?

Exercise 0.9. Type in the Maple code **`Pn`**, corresponding to P_n above. For small values of n, use Maple to show that

$$P_{2^k} \quad = \quad Q_k.$$

Once again, use two different procedures: one using **`simplify`**, and the other using **`evalf`**. Something strange happens when using **`evalf`**. Investigate this behaviour. The Maple variable **`Digits`** allows you to control the number of digits a numerical representation has. How is the computation affected by changing **`Digits`**?

3. D. Kahaner, C.M. Moler, and S. Nash, *Numerical Methods and Software,* Prentice-Hall, New York, 1989.

4. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, Second Edition, Cambridge University Press, Cambridge, UK, 1992.