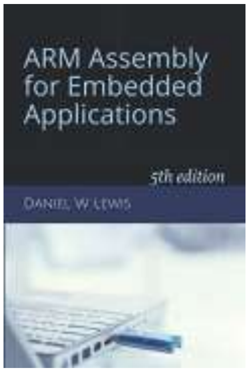


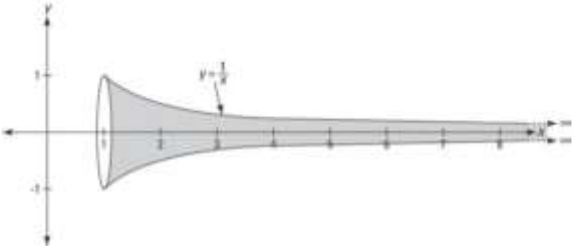
Programming Lab 9B

Gabriel's Horn

Click to download Lab9B-Main.c

Topics: Floating-point instructions, numerical integration, precision.

Prerequisite Reading: Chapters 1-9
Revised: May 27, 2021



Background*[1]*: Gabriel's horn (also known as Torricelli's trumpet) is a geometric figure formed by taking the graph of $1/x$ and rotating it about the x-axis from $x = 1$ to ∞. Calculus can be used to show that the volume it creates is finite (and equal to π), and that the surface area is infinite. This creates an apparent paradox because even though the horn could be filled with paint, there would never be enough paint to coat the surface.

In this lab, the volume (V) and surface area (A) are computed by numerical integration. Conceptually, the horn is divided into a stack of very thin disks of constant thickness dx and diminishing radius r_i so that the volume of the horn may be computed as the sum of their individual volumes. (A similar approach is used for the surface area.) The thickness of a disk spans two successive points on the x-axis; since the actual radius at each value of x differs, the radius of the disk is approximated as the average of these two values.

$$V = \sum (\pi r_i^2)\, dx$$

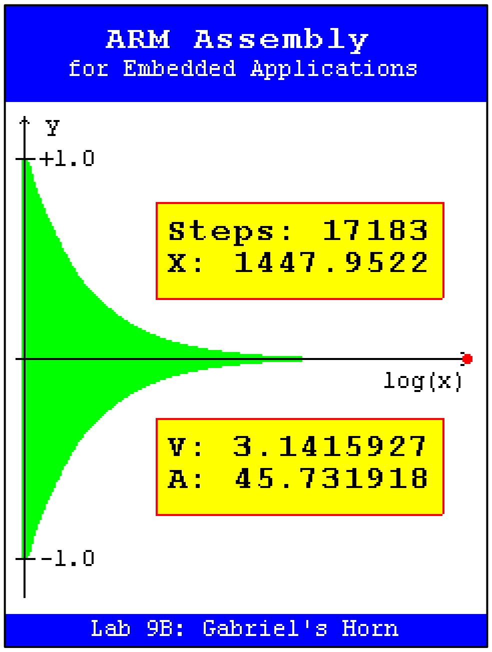
$$A = \sum (2\pi r_i)\, dx$$

$$r_i = (\frac{1}{x_i} + \frac{1}{x_{i+1}})/2$$

Implementation: The limited precision of floating-point arithmetic raises three implementation issues. (1) First, each of the arithmetic operations may produce an imprecise result; this introduces small errors that accumulate during the summation. Accuracy may be improved however, by removing the common parameters π and dx from each term and performing these multiplications only *once* after the summation is completed. (2) Second, if dx is too small, more error will accumulate due to the larger number of terms, but if dx is too large, the approximation of the radius as an average becomes less accurate. The best value of dx for this application was found experimentally to be 0.08420836. (3) Third, the diminishing amounts added to the sum eventually become so small relative to the sum that the sum no longer changes and thus determines when the summation should be terminated.

Assignment: The main program includes a C function named `Integrate` that performs the numerical integration. I.e., you can compile and run the program without writing any assembly. However, your task is to create an equivalent replacement for `Integrate` in assembly. The original C version has been defined as "weak", so that the linker will automatically replace it in the executable image by the one you create in assembly; you do not need to remove the C version.

Note that your `Integrate` function must call two C functions (`DeltaX` and `UpdateDisplay`) that are provided in the main program; do *not* recreate these in assembly. Your assembly language code will likely need to push and pop several registers; be sure that the total number of registers you push and pop is even so that the address held in the stack pointer remains a multiple of eight to satisfy the data alignment requirements of compiled C code. If your code is correct, the display should look similar to the image shown. The display is updated as the integration proceeds. The volume (V) should converge to π, but the surface area (A) should continue to grow until the process terminates when the volume stops changing. Holding down the blue pushbutton will pause the process; releasing it allows the process to resume. Once the process completes, pressing and releasing the blue pushbutton will restart the process.


ARM Assembly
for Embedded Applications

```
Steps:  17183
X:  1447.9522
```

```
V:  3.1415927
A:  45.731918
```

Lab 9B: Gabriel's Horn

[1] https://en.wikipedia.org/wiki/Gabriel%27s_Horn