Integration method

Alex Shen

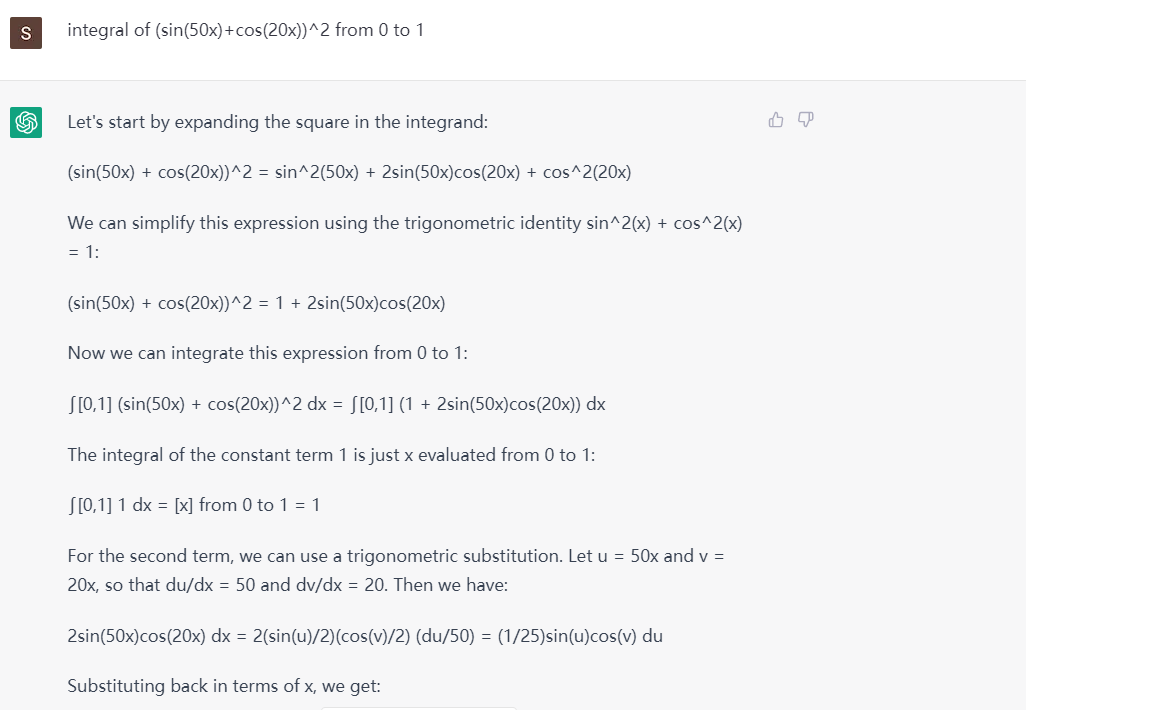
2023-11-13

Table of Contents

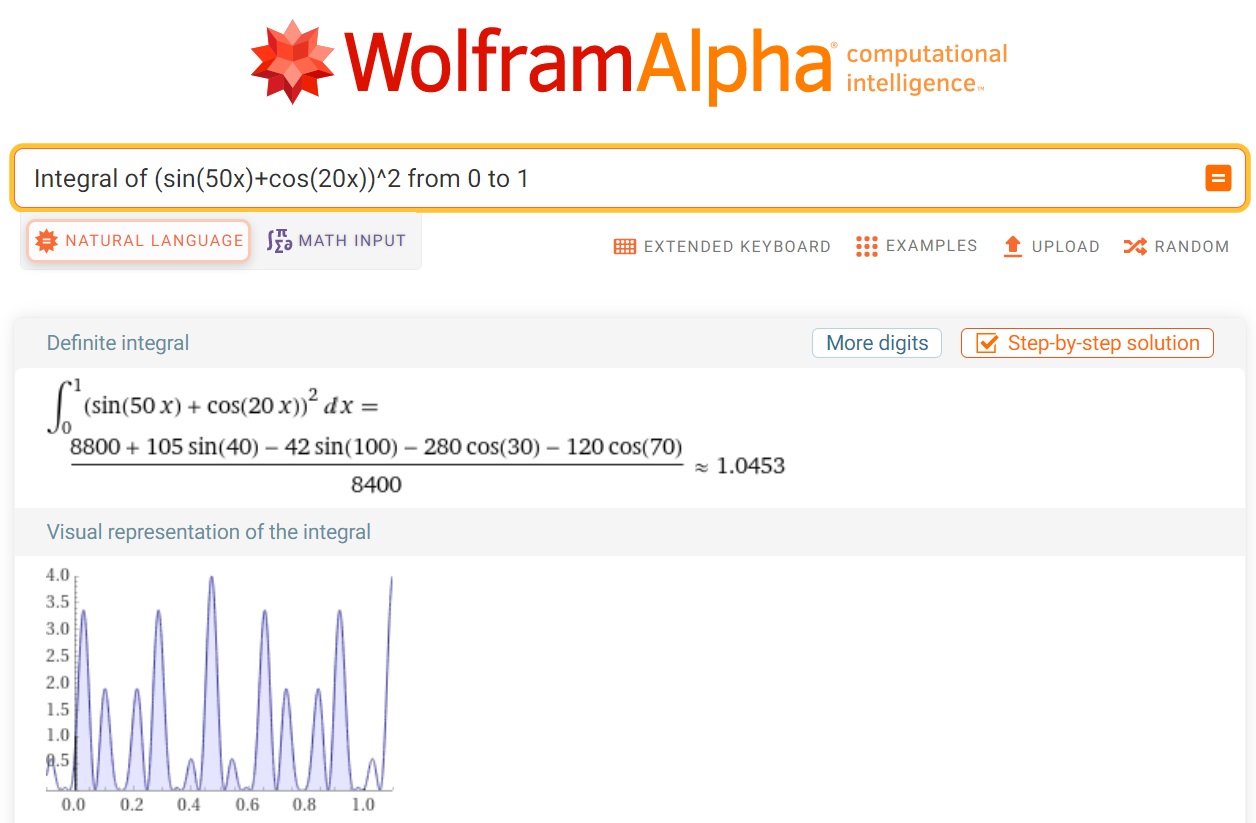
## The integration problem

How to solve the integral of (sin(50x)+cos(20x))^2 from 0 to 1

There are couple of ways we do it, for instance checking it online web, integrating it analytically, or using numerical method, or Monte Carlo method.

First, Let us ask the big boss of chat.GPT. It turns out a disaster:-) 

Secondly, Let’s try WolframAlpha online

* Copy “Integral of (sin(50x)+cos(20x))^2 from 0 to 1”
* Paste it into [WolframAlpha](https://www.wolframalpha.com/)
* The engine will translate it into a correct math integral, and give it back a number and a very nice visual representation of the integral 

## Analytic method

This is the preferable method, but many integrals may be hard or even impossible.

f<- function(x) {  
 x-0.5\*sin(100\*x)/100+0.5\*sin(40\*x)/40-cos(70\*x)/70-cos(30\*x)/30  
}  
  
f(1)-f(0)

## [1] 1.045276

## Numerical integration

### Trapezoidal rule programming by loop

integrate\_trapezoidal\_1 <- function(f, a, b, n) {  
 h <- (b - a) / n  
 result <- (f(a) + f(b)) / 2  
 for (i in 1:(n - 1)) {  
 result <- result + f(a + i \* h)  
 }  
 result <- result \* h  
 return(result)  
}  
  
myfn<- function(x)(sin(50\*x)+cos(20\*x))^2  
  
integrate\_trapezoidal\_1(myfn,0,1,10000)

## [1] 1.045276

### Trapezoidal rule probramming by vector

# I can do it myself using vector programming instead of using for loop  
integrate\_trapezoidal\_2<- function(myfn,a,b,n) {  
 h<-(b-a)/n  
 x.vec <- seq(a,b,by = h)  
 f.vec <- sapply(x.vec, myfn)   
 A<- h\*(f.vec[1]/2+sum(f.vec[2:n])+f.vec[n+1]/2)  
 return(A)  
}  
  
myfn<- function(x)(sin(50\*x)+cos(20\*x))^2  
  
integrate\_trapezoidal\_2(myfn,0,1,10000)

## [1] 1.045276

### Simpson’s rule coding 1

integrate\_simpson<- function(myfn,a,b,n) {  
 h<-(b-a)/n  
 x.vec <- seq(a,b,by = h)  
 f.vec <- sapply(x.vec, myfn)   
 sum<- f.vec[1]+f.vec[n+1]  
 for (i in 2:n){  
 if (i%%2 == 0)  
 sum = sum + 4\*f.vec[i] #f.vec[2] corresponding to f(x1)  
 else   
 sum = sum + 2\*f.vec[i] #f.vec[3] corresponding to f(x2)  
 }  
 sum = sum\*h/3  
 print(sum)  
}  
  
myfn<- function(x)(sin(50\*x)+cos(20\*x))^2  
  
integrate\_simpson(myfn,0,1,1000)

## [1] 1.045276

### Simpson’s Rule coding 2

integrate\_simpson\_2 <- function(f, a, b, n) {  
 h <- (b - a) / n  
 x <- seq(a, b, length.out = n + 1)  
 y <- f(x)  
 result <- h / 3 \* (y[1] + 4 \* sum(y[seq(2, n, 2)]) + 2 \* sum(y[seq(3, n-1, 2)]) + y[n + 1])  
 return(result)  
}  
  
myfn<- function(x)(sin(50\*x)+cos(20\*x))^2  
  
integrate\_simpson\_2(myfn,0,1,1000)

## [1] 1.045276

### R built-in function of integrate

integrate(function(x) (sin(50\*x)+cos(20\*x))^2, lower =0, upper = 1)

## 1.045276 with absolute error < 2.1e-10

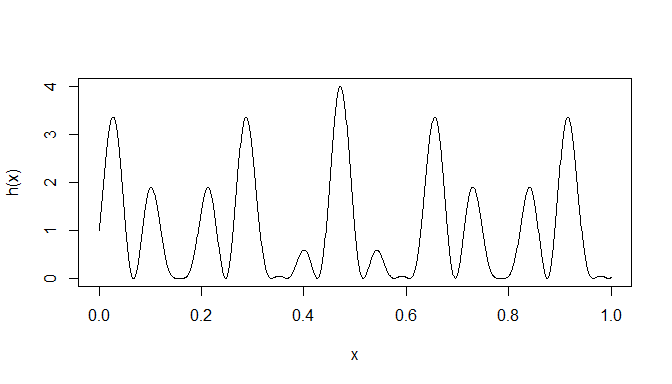
## Integration by Monte Carlo

1. generate x1, x2,…xn from uniform(a, b)
2. Compute h(x1), h(x2), …, h(xn)
3. Estimate E(h(X)), by averaging h(X)
4. Estimate the integral I=(b-a)\* average of h(x)

Here h(x)= (sin(50x)+cos(20x))^2

First define the function, and draw a visual representation of the integral

x<- seq(0,1, by = 0.001)  
h<- function(x)(sin(50\*x)+cos(20\*x))^2  
plot(x,h(x), type="l")

 Secondly, let’s do it using Monte Carlo

set.seed(1)   
n <-10^6 #Set the number of points to generate  
x<-runif(n, 0,1) #generate n number of points from uniform (0,1)  
h\_x<- h(x)  
hbar\_x <- mean(h(x)) #compute h(x), and the mean of h(x)  
(I = 1\*hbar\_x) #mean times the interval = integral

## [1] 1.04515

Third step, how much variation of the hbar\_x becomes as we increase the number of n.

n <- 10^4  
hbar\_n <-cumsum(h\_x[1:n])/1:n # to compute cumulative mean of h(x)  
  
#To estimate variance of hbar\_m  
var\_m <- function(m){  
 sum((h\_x[1:m]-hbar\_n[m])^2)/m^2  
}  
  
v\_n <- apply(matrix(1:n),1,var\_m)  
  
se\_n <- sqrt(v\_n)  
  
plot(1:n,hbar\_n, type = "l")  
  
lines(hbar\_n+1.96\*se\_n, col = "blue")  
  
lines(hbar\_n-1.96\*se\_n, col = "blue")

