CS 3430: S19: SciComp with Py
Lecture 15
Definite Integral Approximation

Vladimir Kulyukin
Department of Computer Science
Utah State University

Review

Definition of Net Change

Suppose that $F' = f$. Then,

$$\int_a^b F'(x)dx = F(b) - F(a).$$

F' is the rate of change of F. The integral of the rate of change of F is the **net change** of F as x varies from a to b.

The Area under a Graph

Let $f(x)$ be a continuous non-negative function on some interval $[a, b]$. The area under the graph of $f(x)$ from a to b is the area bounded by the graph of $f(x)$ from above, the x-axis, and the vertical lines $x = a$ and $x = b$.

Theorem 1: A Fundamental Theorem of Integral Calculus

Let $f(x)$ be a continuous non-negative function on some interval $[a, b]$. The area under the graph of $f(x)$ from a to b bounded by the graph of $f(x)$ from above, the x-axis, and the vertical lines $x = a$ and $x = b$ is

$$\int_a^b f(x)dx = F(b) - F(a),$$

where F is an antiderivative of f.

Partition

Let $f(x)$ be a continuous non-negative function on the interval $a \leq x \leq b$. Let the x-axis interval be divided into $n > 0$ equal subintervals. This subdivision is called **partition**.

The width of each subinterval is $(b-a)/n$. Another way of saying it is

$$\Delta x = \frac{b-a}{n}.$$

Riemann Sum: Definition

Suppose there is a partition with n subintervals. Let's pick a point x_i in each subinterval so that x_1 is in subinterval 1, x_2 is in subinterval 2, etc.

Let Δx be the width of each subinterval. Then $f(x_1)\Delta x$ is the area of the rectangle above the first subinterval, $f(x_2)\Delta x$ is the area of the rectangle above the second subinterval, etc.

The Riemann sum is defined as

$$f(x_1)\Delta x + f(x_2)\Delta x + ... + f(x_n)\Delta x = \\ [f(x_1) + f(x_2) + ... + f(x_n)]\Delta x.$$

Theorem 2: A Fundamental Theorem of Integral Calculus

Let $f(x)$ be a continuous non-negative function on some interval $[a, b]$. Then,

$$\lim_{\Delta x \to 0}[f(x_1) + f(x_2) + ... + f(x_n)]\Delta x = \int_a^b f(x)dx = F(b) - F(a),$$

where F is an antiderivative of f.

Consumer's Surplus

The **consumer's surplus** for a commodity having demand curve $p = f(x)$ is

$$\int_0^A [f(x) - B]dx.$$

where A is the quantity demanded and $f(A) = B$ is the current price.

Definite Integral Approximation with Riemann Sums

Motivation

It is not always possible to evaluate definite integrals that arise in practical problems by computing antiderivatives.

Mathematicians keep compiling tables of antiderivatives; software engineers working on scientific computing systems keep integrating these rules into differentiation and integration engines; there is no end in sight!

However, in many practical situations it is impossible to express an antiderivative in terms of elementary functions or or the function we want to integrate is unknown.

Using Riemann Sums to Approximate Definite Integrals

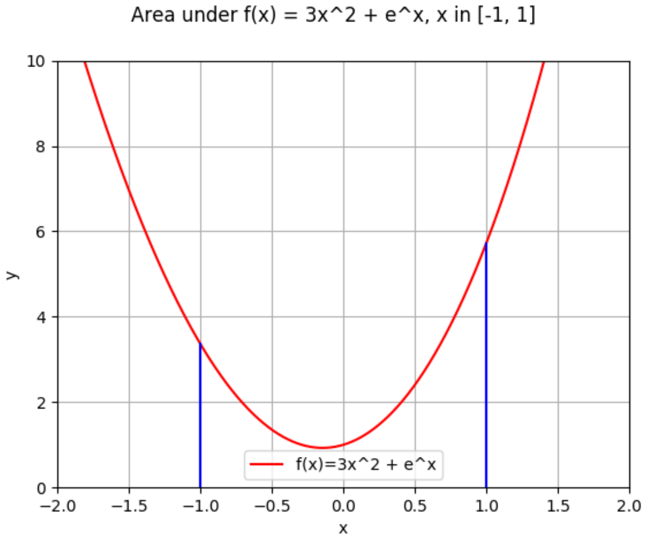
We can use theorem 2 to approximate an arbitrary intergral with a Riemann sum.

The theorem assures us that as the number of subintervals in a partition increases, the Rieman sum approaches the true value of the definite integral closer and closer.

Problem

Let $f(x) = 3x^2 + e^x$ and let $x \in [-1, 1]$. Use Riemann sums to approximate $\int_{-1}^{1}(3x^2 + e^x)dx$.

Solution: $f(x) = 3x^2 + e^x, x \in [2, 3]$



Solution

Let's establish the ground truth for this problem.

$$\int_{-1}^{1}(3x^2 + e^x)dx = (x^3 + e^x)|_1^3 = (1 + e^1) - (-1 + e^{-1}) = 2 + e - \frac{1}{e} \approx 4.35$$

Solution

Let's compute three Riemann sum approximations by taking the middle point, the left point, and the right point from each subinterval in a given partition.

```
from riemann_approx import riemann_approx
def riemann_approx_mid():
  fex = make_prod(make_const(3.0), make_pwr('x', 2.0))
  fex = make_plus(fex, make_e_expr(make_pwr('x', 1.0)))
  for n in range(5, 101):
      print(n, riemann_approx(fex, -1.0, 1.0, n, pp=0))

def riemann_approx_left():
  fex = make_prod(make_const(3.0), make_pwr('x', 2.0))
  fex = make_plus(fex, make_e_expr(make_pwr('x', 1.0)))
  for n in range(5, 101):
      print(n, riemann_approx(fex, -1.0, 1.0, n, pp=-1))

def riemann_approx_right():
  fex = make_prod(make_const(3.0), make_pwr('x', 2.0))
  fex = make_plus(fex, make_e_expr(make_pwr('x', 1.0)))
  for n in range(5, 101):
      print(n, riemann_approx(fex, -1.0, 1.0, n, pp=1))
```

Solution

Let's vary the number of subintervals from 5 to 100.

The best approximation for the mid point Riemann sum is for $n = 100$: 4.350163214371502. Error = $|4.350163214371502 - 4.35| = 0.00016321437150246254$.

The best approximation for the left point Riemann sum is $n = 100$: 4.327376709638665. Error = $|4.327376709638665 - 4.35| = 0.022623290361333437$.

The best approximation for the right point Riemann sum is $n = 100$: 4.374384757384417. Error = $|4.374384757384417 - 4.35| = 0.024384757384416922$.

Approximation of Definite Integrals with Midpoint, Trapezoidal, and Simpson Rule

Midpoint Rule

The midpoint rule is the Riemann sum approximation where we take the mid point from each subinterval in a given partition.

Problem

Let's use the midpoint rule to approximate $\int_0^2 2xe^{x^2}\,dx$.

Solution

The ground truth can be obtained by using integration by substitution.

$$\int_0^2 2xe^{x^2}dx = e^{x^2}|_0^2 = e^4 - 1 \approx 53.59815003314423.$$

Solution

We use the midpoint rule with 5000 subintervals.

```
from defintegralapprox import midpoint_rule
fex = make_prod(make_const(2.0), make_pwr('x', 1.0))
fex = make_prod(fex, make_e_expr(make_pwr('x', 2.0)))
approx = midpoint_rule(fex,
                       make_const(0.0),
                       make_const(2.0),
                       make_const(5000))
print(approx)
```

The above code outputs 53.5981434947. The ground truth is 53.59815003314423. Error = 6.538444232262464e-06.

Trapezoidal Rule

$$\int_a^b f(x)dx \approx [f(a_0) + 2f(a_1) + ... + 2f(a_{n-1}) + f(a_n)]\frac{\Delta x}{2}.$$

Problem

Let's use the trapezoidal rule to approximate $\int_0^2 2xe^{x^2}\,dx$.

Solution

We use the trapezoidal rule with 5000 subintervals.

```
from defintegralapprox import trapezoidal_rule
fex = make_prod(make_const(2.0), make_pwr('x', 1.0))
fex = make_prod(fex, make_e_expr(make_pwr('x', 2.0)))
approx = trapezoidal_rule(fex,
                          make_const(0.0),
                          make_const(2.0),
                          make_const(5000))
print(approx)
```

The above code outputs 53.59816311. The ground truth is 53.598150033144236. Error = 1.961530000471612e-05.

Simpson's Rule

Let M and T be the estimates from the midpoint and trapezoidal rules. Simpson's rule esitmation is

$$S = \frac{2M+T}{3}.$$

Problem

Let's use the Simpson rule to approximate $\int_0^2 2xe^{x^2}\,dx$.

Solution

We use the Simpson rule with 5000 subintervals.

```
from defintegralapprox import simpson_rule
fex = make_prod(make_const(2.0), make_pwr('x', 1.0))
fex = make_prod(fex, make_e_expr(make_pwr('x', 2.0)))
approx = simpson_rule(fex,
                      make_const(0.0),
                      make_const(2.0),
                      make_const(5000))
print(approx)
```

The above code outputs 53.5981500331. The ground truth is 53.59815003314423. Error = 6.5384000009771626e-06.

Error of Approximation Theorem

Let n be the number of subintervals used in an approximation of the definite integral $\int_a^b f(x)dx$.

1) The error for the midpoint rule is at most $\frac{A(b-a)^3}{24n^2}$, where A is a number such that $|f''(x)| \leq A$ for all $x \in [a, b]$.

2) The error for the trapezoidal rule is at most $\frac{A(b-a)^3}{12n^2}$, where A is a number such that $|f''(x)| \leq A$ for all $x \in [a, b]$.

3) The error for Simpson's rule is at most $\frac{A(b-a)^5}{2880n^4}$, where A is a number such that $|f''''(x)| \leq A$ for all $x \in [a, b]$.

Problem

Five milligrams of dye is injected into a vein leading to the heart. The concentration of the dye in the aorta, an artery leading from the heart, is determined every 2 seconds for 22 seconds. Let $c(t)$ be the concentration in the aorta after t seconds. Use the trapezoid rule to estimate $\int_0^{22} c(t)dt$.

Seconds after injection	0	2	4	6	8	10	12	14	16	18	20	22
Concentration (mg/liter)	0	0	0.6	1.4	2.7	3.7	4.1	3.8	2.9	1.5	0.9	0.5

Solution

Let $n = 11$. Then $a = 0$, $b = 22$, and $\Delta t = (22 - 0)/11 = 2$.
Then,

$$\int_0^{22} c(t)dt = [c(0) + 2c(1) + ... + 2c(20) + c(22)]\frac{2}{2} \approx 43.7 \text{ liters.}$$

Solution

We can define a function that computes $c(t)$ as follows.

```
def concentration_in_aorta(t):
    if t == 0:
        return 0
    elif t == 2:
        return 0
    elif t == 4:
        return 0.6
    ...
    elif t == 22:
        return 0.5
    else:
        raise Exception('Illegal value: ' + str(t))
```

Solution

The test below passes.

```
from defintegralapprox import trapezoidal_rule_aux
approx = trapezoidal_rule_aux(concentration_in_aorta,
                              0.0, 22.0, 11)
err = 0.0001
print(approx)
assert abs(approx - 43.7) <= err
```

References

1. L. Goldstein, D. Lay, D. Schneider, N. Asmar. *Calculus and its Applications*, Chapters 6, 9. Pearson.
2. `www.python.org`.