- 이번 주차 과제의 툴은 R입니다. 과제의 조건 및 힌트는 R을 기준으로 하지만, Python을 사용해도 무방합니다.
- 제출형식은 HTML, PDF 모두 가능합니다. **.ipynb 이나 .Rmd 등의 소스코드 파일은 불가능합니다.** 파일은 psat2009@naver.com으로 보내주세요.

  제출기한은 2월 15일 **목요일 자정까지** 입니다. 패키지와 마찬가지로 무단 미제출 2회 시 퇴출이니 유의해주세요.

Chapter 1 : 선형 근사와 테일러 전개, Numerical Analysis

리드오프에서 선형 근사와 테일러 전개를 배웠습니다. 이는 수치 해석이라고도 할 수 있는데요, 어떤 함수나 방정식의 해를 수치적으로 근사해서 근삿값을 구하는 알고리즘에 대한 연구를 하는 학문입니다.
Chapter 1은 이처럼 기본적인 R의 반복문과 함수를 이용해 원하는 알고리즘을 구현하고, 구하고자 하는 값들을 수치적으로 근사할 수 있음을 알아보는 것이 목적입니다. 반복 명령을 수행하는 함수를 짜고, 특정 조건이 만족되면 반복을 멈추도록 하는 알고리즘을 구현해봅시다.

문제1. 테일러 전개는 선형 근사를 확장하여, 다항함수를 이용해 함수를 근사하는 방법입니다. 즉, 차수가 많아질 수록 더 정확한 근사가 가능해집니다. 이러한 테일러 전개는 원주율 π의 값을 구하는 데에도 큰 공헌을 하였습니다. $\tan(x)$의 역함수인 $\arctan(x)$를 통해 원주율 π를 추정할 수 있는데요, $\arctan(x)$를 테일러 전개하면 $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1}$ 임을 이용하겠습니다.

문제 1-1. 테일러 전개를 이용해 π의 값을 근사하겠습니다. for() 문을 이용하며, 요구사항은 다음과 같습니다.

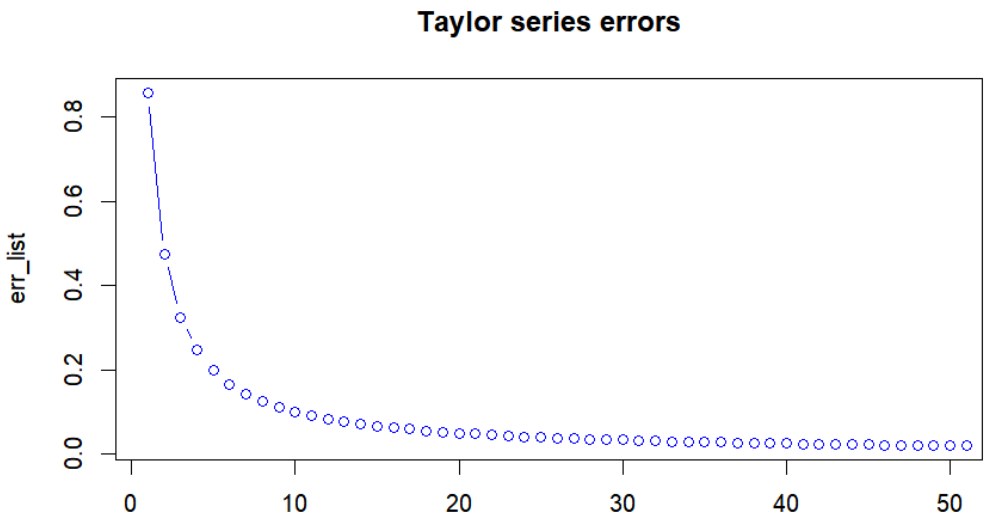
- 위 $\arctan(x)$를 테일러 전개한 식을 이용하여, 각 반복에서 pi_est 와, 실제 원주율과의 차인 err 를 구해주세요. n 번째 반복에서는 n 개의 항으로 근사함을 의미합니다.

  (HINT1) arctan(1) = $\pi/4$ 임을 이용하면 π의 값을 간단하게 추정할 수 있습니다.

  (HINT2) pi_est = pi_est + n 번째 항 과 같은 방식으로 각 반복마다 pi_est 를 구할 수 있습니다.
- 각 수행마다 pi_est 와 실제 π(pi)의 차인 err 를 err_list 에 저장해주세요.

  (HINT) err 는 절댓값 abs()를 이용합니다. 또 err_list = c(err_list, err)로 해주시면 됩니다.
- n = 50 까지 반복해주세요.

문제1-2. err_list를 다음과 같이 시각화해주세요. 형식은 자유입니다.



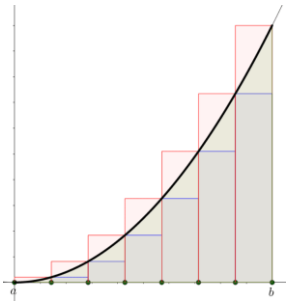
문제2. 이러한 관점 또는 반복을 이용하면 복잡한 함수에 대한 미분과 적분 또한 수치적 근사를 이용해 수행할 수 있습니다. 이러한 관점을 적용하여, 미분과 적분의 기능을 수행하는 함수를 짜봅시다.

문제 2-1. diff() 함수를 선언해주세요. 매개변수는 미분하고자 하는 함수 f, 미분 지점 x, 오차 h로 구성됩니다.

- fun()에 $x^3 - 3x^2 + 2x$ 함수를 저장해주세요.
  (HINT) 아래와 같이 선언하세요. 이후 문제에도 같은 방식으로 선언해주시면 됩니다.
  `fun <- function(x) {x^3 - 3*x^2 + 2*x}`
- 아주 작은 h에 대해 $f(x+h) \approx f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \cdots$ 입니다. 즉, $\frac{f(x+h)-f(x)}{h} \approx f'(x)$ 임을 이용해주세요.
- diff()로 fun(x)의 $x = 2$에서 미분을 수행해보세요. 이후 실제 값과 비교해보세요. h 는 10^{-4} 로 설정합니다.

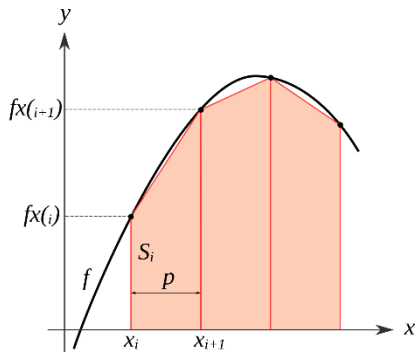
문제 2-2. integ() 함수를 선언해주세요. 매개변수는 적분하고자 하는 함수 f, 적분 시작 지점 a, 적분 종료 지점 b, 나누는 구간의 개수 n으로 구성됩니다. 함수 내에서 요구사항은 다음과 같습니다.

- 구분구적법의 아이디어를 적용합니다. 즉, a 와 b 사이의 구간을 여러 개의 쉬운 도형으로 나눈 후 넓이를 구하는 것입니다.
- 직사각형의 넓이를 구해 계산하겠습니다. 그림으로 나타내면 다음과 같습니다.



- Integ() 함수를 이용해 적분을 수행해주세요. 적분하고자 하는 함수는 $2\sin x$ 이며, 2π 부터 $5\pi/2$ 까지 적분합니다.
- n=10 일 때와 n=1000 일 때의 값을 비교해보세요.

(BONUS 1) 직사각형이 아닌 사다리꼴의 넓이를 구해 진행해봅시다(integ2() 선언). Integ2()로 다시 문제 2-2 를 풀고, 결과를 비교해 작성해주세요. 그림으로 나타내면 다음과 같습니다.



(BONUS 2) MonteCarlo 적분에 대해 알아본 후 서술해주세요. MonteCarlo 시뮬레이션을 통해 함숫값을 추정하거나 적분을 할 수 있는 통계적 근거를 알아본 후 한두 문장으로 작성해주세요.

(BONUS 3) 몬테카를로 적분을 이용해 문제 2-2 의 함수의 적분을 다시 수행해주세요.
(HINT) runif() 함수로 난수를 생성한 뒤, 그 난수에서 함수가 갖는 값들의 평균을 계산하면 됩니다.

Chapter 2: Gradient Descent, Newton's method

Gradient Descent method는 최적 값을 향한 방향을 계산하는데 1차 미분을 사용하는 방법이었습니다. 반면 뉴턴 방법(Newton's method)은 2차 미분까지 사용하는데요. 뉴턴 방법은 Gradient Descent 보다 수렴 속도가 훨씬 빠르다는 장점이 있습니다. 그 외에도 많은 Optimizing(최적화) 기법들이 존재하고, 이번엔 위 2가지 방법에 대해 알고리즘을 구현하는 연습을 해보겠습니다.

문제1. 제약식이 없는 최적화를 고려합니다. objfun()에 $x^4 - 3x^3 + 2x$ 를 저장해주세요.

문제2. Gradient Descent 알고리즘을 구현해보겠습니다. Grad_Desc() 함수를 선언해주세요. 요구사항은 다음과 같습니다.

- 매개변수는 최적화하고자 하는 함수 f, 학습비율(learning rate) r=0.01, 시작지점 s, 최대반복 max_iter=1000 입니다.
- 각 반복에서, 점 x에서의 gradient를 계산 후, Gradient Descent 알고리즘에 따라 x를 업데이트합니다. (교안 참고)
  (BONUS 4). R에 내장된 미분함수를 사용하는 것이 아닌, 문제2에서 작성했던 diff() 함수를 이용해 반복문을 짜주세요.

- 반복문을 멈추는 조건을 설정해주세요. 새로 업데이트된 값과 원래 값의 차가 0.001보다 작으면 수렴으로 판단하여 반복을 멈추도록 해주세요. 또는 반복수가 최대반복 max_iter에 도달해도 반복을 멈춰주세요.
- 함수는 반복이 멈춘 시점에서의 x값과, 그 값에서의 함수 값, 반복 수를 print하도록 해주세요.
- objfun() 함수에 대해 알고리즘을 수행해봅시다. 시작지점으로 각각 5, -5를 설정한 후 함수를 실행해 결과를 비교해주세요. 또한 learning rate를 각각 0.001과 0.01, 0.1로 설정한 후 실행시켜봅시다. 결과를 비교한 후 해석해주세요.

문제3. Newton 방법은 gradient에 곡률(curvature)을 같이 고려하면서 해를 찾아가는 방식입니다. 즉 Newton's method는 Gradient Descent와 유사한 알고리즘이지만, 각 반복마다

$$\mathbf{x} \leftarrow \mathbf{x} - \left(\nabla_{\mathbf{x}}^2 f(\mathbf{x})\right)^{-1} \nabla_{\mathbf{x}} f(\mathbf{x})$$

의 방식으로 다음 값을 업데이트합니다. Newton's method가 Gradient Descent보다 더 빨리 수렴할 수 있는 이유를 추측해보고 작성해주세요. (정답은 없습니다)

(여기부터는 보너스)

문제4. Newt()을 선언해주세요. 매개변수는 최적화하고자 하는 함수 f, 시작지점 s, 최대반복 max_iter=100입니다.

- 위의 수식을 참고하여 Newton's method 알고리즘을 짜주세요.
- (HINT) 함수의 이계도함수는 역시 테일러 전개를 이용하여 수치적으로 근사할 수 있습니다.

  $$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

  임을 이용해 함수의 곡률을 계산할 수 있습니다.
  r에 내장된 deriv 함수를 2번 적용하는 것도 하나의 방법입니다.
- 역시 업데이트 후 감소량이 일정 수준 이하로 작아지면 최적해에 거의 근접했다는 신호로 볼 수 있습니다. 감소량이 0.001보다 작은 조건을 만족하면 반복을 멈출 수 있도록 해주세요.
- objfun() 함수에 대해 알고리즘을 수행해봅시다. 시작지점으로 각각 5, -5를 설정한 후 함수를 실행해주세요. Gradient Descent 알고리즘과 결과를 비교해주세요.