**Problem Set #2**

Numerical analysis

201621505 채진기

**1. (a)** 주어진 g(x) = x^(2) – 2에서, 함수 g의 최고 차항, x^(2)의 계수가 1이기 때문에

그래프를 그렸을 때 우 상향하는 이차곡선을 볼 수 있습니다. 몇 개의 값을 넣어보면 실제로

N 이 증가함에 따라 다음 항의 값이 더 커짐을 볼 수 있습니다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 0 | 1 | 2 | 3 | 4 | 5 |
| P(n) | 2.5 | 4.25 | 16.0625 | 256.00390 | 65535.99681 | 4294966875.88033 |

**1. (b) 코딩한 결과와 소스 코드를 첨부하였습니다. (1-b Steffensen)**

P0 = 2.5 로 설정한 후, Aitken’s delta square method를 이용해 새로운 점을 반복하여 계산하는

코드를 작성했고, 그 결과 발산하는 수열임에도 fixed point 중 2의 값에 가까이 가는 값을

볼 수 있었습니다.

**2.** **코딩한 결과와 소스 코드를 첨부하였습니다. (2 - Legendre)**

뉴턴 메소드를 적용하기 위한 함수를 정의해주고, 30번째 Legendre polynomial을 구하기 위한

방식으로, 함수를 정의하는 것이 아닌, 배열을 통한 저장방식을 채택했습니다.

배열 자체에 식을 대입하고 차곡차곡 저장해줌으로써 python 내에서 연산이 가능했습니다.

배열에 저장된 값을 이용하지 않으면 연산횟수가 너무 많아져 프로그램을 실행시키지 못했습니다.

결과적으로, 4번의 실행 후 오차가 10e-10, 즉 10의 10제곱 이내의 정확도를 가지는 해를

계산할 수 있었습니다.

**3. 코딩한 결과와 소스 코드를 첨부하였습니다. (3 – midpoint)**

**4. 코딩한 결과와 소스 코드를 첨부하였습니다. (4 – Trapezoidal, 4 - Simpson)**

Simpson’s method는 홀수항과 짝수항에 대한 시그마 연산을 각각 코딩해줘야 하기 때문에

Cost가 Trapezoidal 방식보다 더 컸습니다. 각 방법의 n에 따른 적분 값은 다음과 같습니다

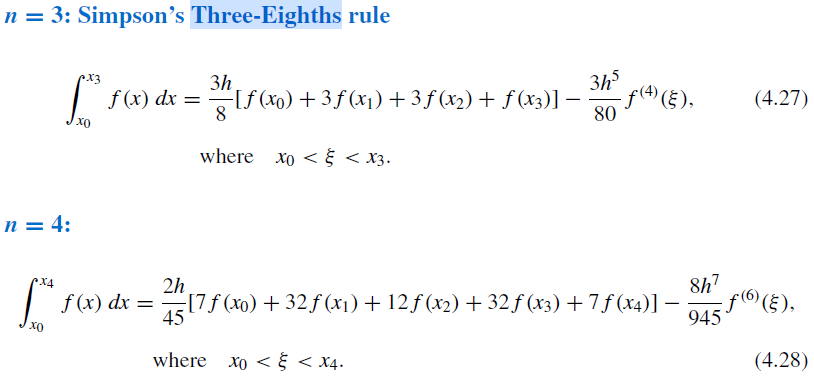
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Trape | 0.1768 | 0.2190 | 0.2363 | 0.2452 | 0.2505 | 0.2540 | 0.2564 | 0.2579 | 0.2594 |
| Simpson | 0.2357 | 0.1972 | 0.1880 | 0.1843 | 0.1824 | 0.1812 | 0.1805 | 0.1799 | 0.1796 |

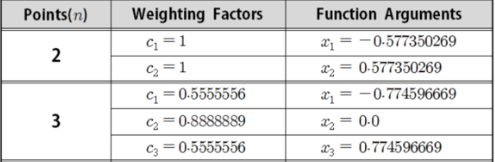
실제로 계산해보면,

Trapezoidal의 방식이 Simpson’s 보다 정확도와 비용 두가지 모두 효율적임을 볼 수 있습니다.

**5. 코딩한 결과와 소스 코드를 첨부하였습니다. (5 – higherorder, 5 – Gaussian)**

Node가 동등하게 3일 때의 결과를 비교하기 위해 Higher order quadrature로써

Simpson’s three-eight rule 중 n = 4 일 때의 가중치를 이용했습니다.

Gaussian의 가중치는 다음 표를 참고했습니다.

두 방식의 결과를 비교해보면, 프로그램을 짜는 과정의 비용은 higher order quadrature로 채택한

Simpson’s three-eights rule이 훨씬 간편했지만 정확도는 Composite gaussian method에 비해

조금 떨어졌습니다. Subinterval을 조절하여 계산할 수 있도록 프로그래밍 했기 때문에

후자의 경우가 더 정확한 결과가 나올 수 있었습니다. 효율성의 부분에서는 이번 문제의

경우에서는 higher order quadrature가 우세하지만, 더 복잡한 피적분함수가 주어졌을 때에는

Composite gaussian quadrature가 정확도 측면에서 효율적일 것이라 생각합니다.

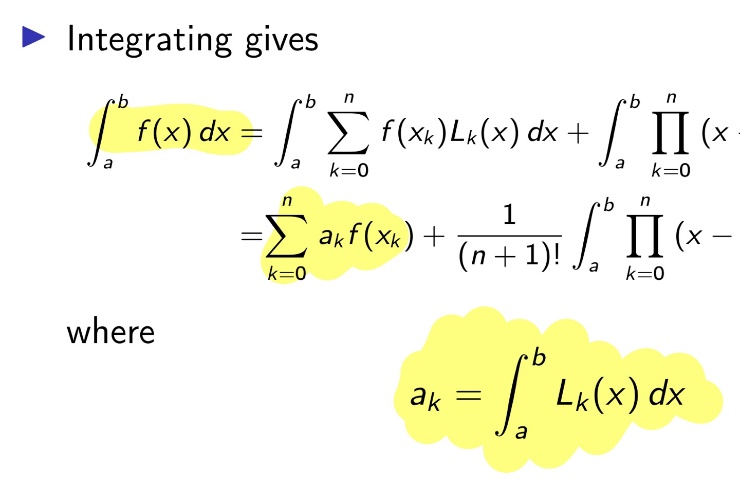
**6. 코딩한 결과와 소스 코드를 첨부하였습니다. (6 – Area, 6 – Circumference)**

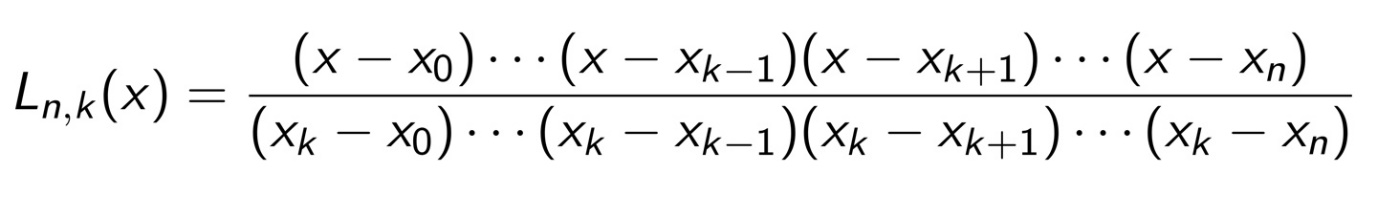
**7. 코딩한 결과와 소스 코드를 첨부하였습니다. (7 – Roomberg)**

프로그램을 짜면서 겪은 장애물은 구간 (a, b) 중 a가 0이라는 점과, 1차원 배열로 Roomberg

행렬을 표현할 수 없다는 것, 두가지였습니다. a 대신 0.000000001 이라는 매우 가까운 값을

대입하고 2차원 배열을 이용하여 Roomberg 행렬을 완성할 수 있었습니다.

**9.** 교수님의 ppt 슬라이드 중 일부를 발췌했습니다.



위의 두가지 내용을 토대로 Newton-Cotes의 근본적인 아이디어를 알 수 있습니다.

따라서 문제에서 구해야 하는 계수 Wk = p/q에 대한 답은 Lagrange polynomial의 basis를

구간 (4, 5)에서 적분한 값입니다. 또한 8개의 노드와 함숫값을 통해 7차이하의 모든 다항함수의

계수(Coefficient)를 결정할 수 있기 때문에, Measuring Precision의 정의에 의해, 최고차항이 7차

이하인 어떠한 다항함수에 대해서 오류가 없는 정확한 적분 값을 계산할 수 있습니다.

예를 들어, L8,1(x)를 구간 (4, 5)에 대해서 적분해보면 다음과 같습니다.



마찬가지로, L8,k(x)를 구간 (4, 5)에 대해서 적분해보면 다음 값들을 얻을 수 있습니다.

K=2)

K=3)

K=4)

K=5)

K=6)

K=7)

K=8)

값들을 비교해보면 규칙성을 볼 수 있습니다.

W1 = W8 = -191 / 120960

W2 = W7 = 1879 / 120960

W3 = W6 = -353 / 4480

W4 = W5 = 68323 / 120960

(통상 0부터 시작하지만, 문제에 맞춰 1에서 8까지로 설정했습니다.)