**Problem Set #3**

Numerical analysis

201621505 채진기

**1 –(a). 소스코드와 코딩한 결과, 그리고 파이썬 파일을 첨부하였습니다.**

원 또는 정사면체와 유사한 구조체의 부피를 근사하는 과정에서 어려운 부분이,

강의영상에서 말씀해 주셨던 것처럼 직선함수 혹은 평면을 추가해서, 구간을 다시 한번 나눠주는

과정이었습니다. x-y 평면에서 y=x 함수를 통해 섹터를 나눠주고, 전체에서 16등분된 조각 중

하나인 부분의 부피를 근사할 때 꼭지점(끝점)에 해당하는 부분에서의 근사값이 복소수가 포함

된 채로 계산되었습니다. 간단한 해결책으로, 복소수 값이 계산되는 애매한 부분인 가장 끝에

점을 제외하고 근사를 시도했고 온전한 실수 값을 얻었습니다.

**1-(b). 소스코드와 코딩한 결과, 그리고 파이썬 파일을 첨부하였습니다.**

부피를 근사하는 과정과 유사했지만 조금 달랐던 점이 있었습니다. 이번에는 z를 바꿔주면서

단면 원의 둘레를 구한 후 더해주었는데, 둘레를 구할 때 x 값의 범위를 알기 위해 x-y평면에서

y = x 와 y = (1 – x^(4))^(0.25) 가 z가 움직임에 따라 어떤 점에서 만나는지 계산해야 했습니다.

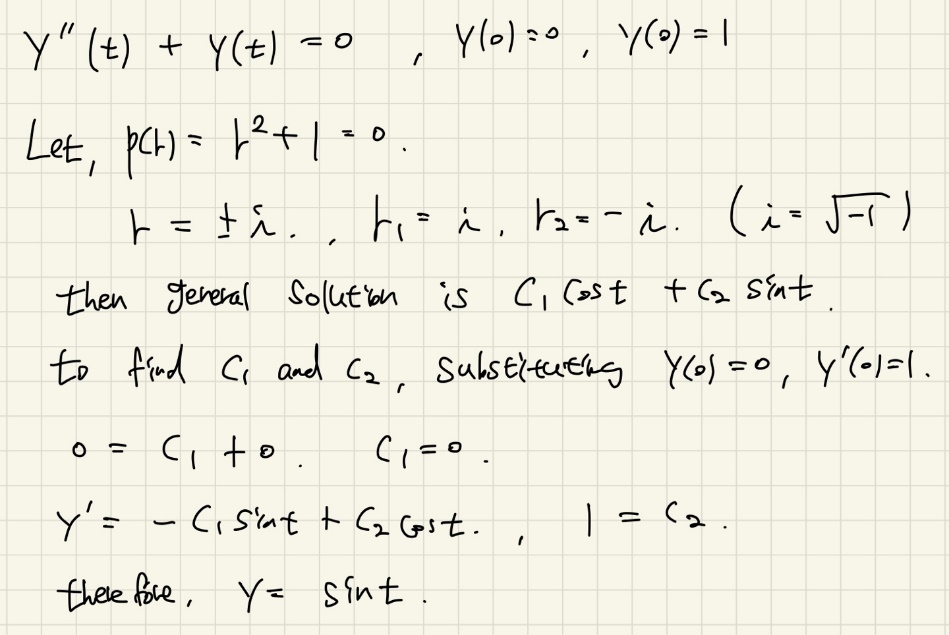
그래서 그 x 값을 계산해주는 함수에 방정식의 해를 구해주는 solve 내장함수를 이용했는데,

이것이 첫번째 난점이었습니다. 그 외에는 크게 어려운 것은 없었는데, 결과가 구체의 표면적과

거의 같아서 잘못된 결과가 나온 것 인지 의아했습니다. 또한 프로그램을 실행하는데 방정식을

푸는 과정 때문인지 시간이 오래 걸렸습니다.

**2 - (a)** analytic solution을 구하는 풀이 과정입니다.

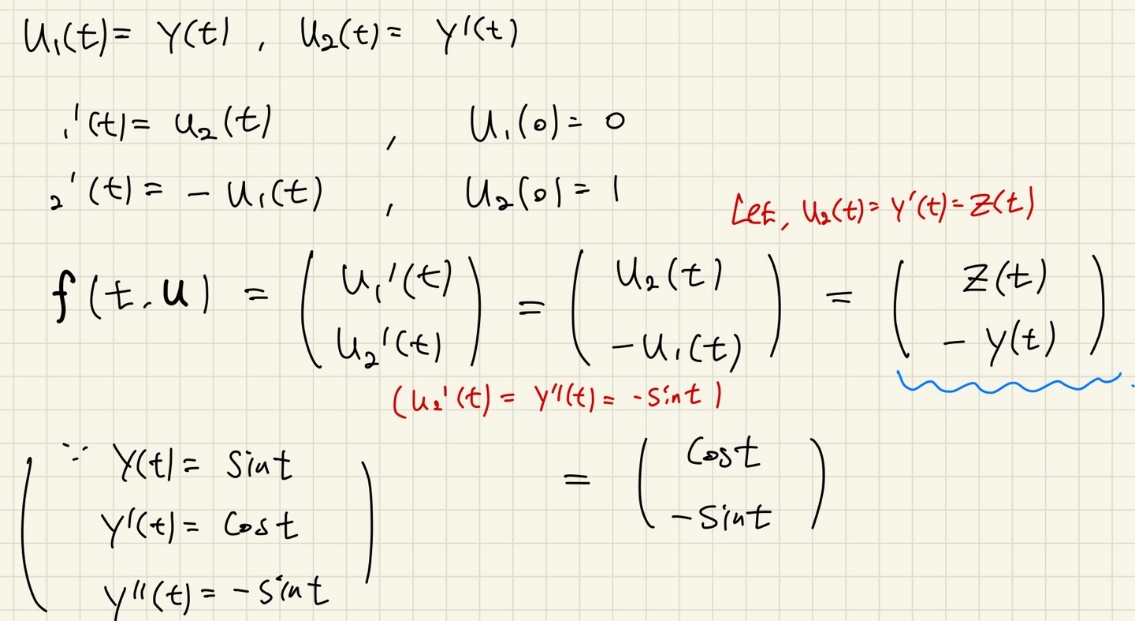


교재 ELEMENTARY DIFFERENTIAL EQUATIONS 중 Chapter 5 Linear second Order Equations를

참고했습니다. 부연설명으로, Characteristic polynomial p(r)의 해가 복소수의 형태일 때 풀이와

같이 일반해를 정의할 수 있습니다. initial value를 대입하여 analytic solution을 도출합니다.

**2 - (b)** Second order equation을 두개의 first order differential equation으로 조정해준 내용입니다.

****

강의에서 다룬 예제와는 다르게, t 만으로 방정식을 표현할 수 있었습니다. 그 결과 코딩하는

과정에서 **w**에 대한 항이 필요없이, t 값 만을 이용해서도 근사가 가능했습니다.

**2 – (c)** RungeKutta vector version을 이용해 y값과 z 값을 근사한 후 그래프로 나타냈습니다.



**(자료는 엑셀파일 2-unit circle data에 두개의 sheet로 저장하여 첨부했습니다.**

**소스코드와 코딩한 결과, 그리고 파이썬 파일을 첨부하였습니다.)**

**2 – (d)** Trapezoidal vector version을 이용해 y값과 z 값을 근사한 후 그래프로 나타냈습니다.



**(자료는 엑셀파일 2-unit circle data에 두개의 sheet로 저장해 첨부했습니다.**

**소스코드와 코딩한 결과, 그리고 파이썬 파일을 첨부하였습니다.)**

**2 – (e).**

Trapezoidal method, 혹은 implicit one-step difference method의 구조를 보면 그 이름에서

알 수 있듯 구하고자 하는 값이 계산 과정에 포함되어 있습니다. implicit method의 특성상

함수가 복잡하다면 연산 과정에서 문제가 있을 수 있지만, 이번 문제에서는 두개의 first order

differential equation을 cos(t) 와 – sin(t)로 설정하고 t에 대해서만 다른 값을 주는 함수였기

때문에 implicit method를 사용하는데 어려움이 없었고 explicit method인 Runge-Kutta method

보다 더 작은 error term을 갖기 때문에 정확한 값을 구할 수 있었다고 생각합니다.

또한 그 함수가 sin 과 cos으로 이루어져 단위원의 개형을 띄게 되었습니다.

**3.** Adams-Bashforth-Moulton fourth-order predictor-corrector method를 이용한 추정치의 absolute

Error는 다음과 같습니다. 에러의 개형을 보았을 때 t의 값이 커짐에 따라 에러 또한 커집니다.

그래서 method의 방식이 잘못된 것인지 확인하기위해 Bashforth 4 order method만 계산한

오류도 비교해보았습니다. (**소스코드와 코딩한 결과, 그리고 파이썬 파일을 첨부하였습니다.**)



absolute Error의 그래프입니다.

두 그래프의 개형은 유사하지만(Bashforth의 결과로부터 재 연산한 결과가 Predictor-corrector이기

때문에), 그 에러값의 크기는 10배 정도 차이가 있음을 볼 수 있습니다.