

**한국외대 중간과제 보고서**

**과목명 선형시스템**

**담당교수 윤일동교수님**

**제출일 20240424**

**전공 컴퓨터공학과**

**학번 202430026**

**이름 이준용**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**(a)** MATLAB 명령 창에 주어진 문장을 입력하면 다음과 같은 결과가 나옵니다:



이 답변의 절대값을 7이 속한 범위의 해상도와 비교하면, MATLAB에서 부동 소수점 산술의 정밀도를 검토해야 합니다. 일반적으로 Octave은 더블 정밀도 부동 소수점 형식을 사용하며, 이 형식의 해상도는 대략 4.4409e-16입니다. 식의 결과는 대략 이 범위 내에 있으므로, 부동 소수점 표현의 해상도와 비슷합니다.

**(b)** 1부터 31까지의 숫자 중에 100으로 나누고 곱했을 때 발생하는 이러한 양자화 오차에 취약한 숫자가 몇 개인지 찾으려면, 원래 숫자와 표현식 7/100\*100 - 7의 결과의 차이를 계산하면 됩니다. 이 차이가 부동 소수점 표현의 해상도 내에 있으면, 해당 숫자는 이 양자화 오차에 취약합니다.

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

이 출력은 1부터 31까지의 숫자 중 이 양자화 오차에 취약한 숫자의 개수를 나타냅니다.

**(c)** 제공된 MATLAB 코드는 반복문을 사용하여 변수 x에 작은 값(2^-52)을 반복적으로 더합니다. 이 값은 양자화 오차를 발생시킬만큼 충분히 작습니다. 루프는 2^3 = 8번 반복됩니다. 프로그램의 결과는 지수 표기법으로 출력된 일련의 숫자이며, 각 숫자는 이전 숫자보다 약간 더 큽니다. 이는 각 반복에서 작은 값 2^-52가 누적되어 변수 x의 값이 점진적으로 증가하기 때문입니다.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

특이값 분해(SVD)는 행렬을 세 개의 행렬로 분해하는 기법으로, 다음과 같이 표현됩니다.

A = USV^T

A는 M x N 크기의 실수 행렬입니다.

U는 M x M 크기의 직교 행렬입니다.

V는 N x N 크기의 직교 행렬입니다.

S는 M x N 크기의 대각 행렬이며, 대각 성분은 A의 특이값 σi (행렬 AT A의 고유값의 제곱근)이 감소하는 순서로 배열되어 있습니다.

가상의 의사 역행렬

특이값 분해는 계수 행렬이 랭크 부족(rank(A) < min(M, N))인 경우에도 가상의 의사 역행렬을 계산하는 데 사용될 수 있습니다. 가상의 의사 역행렬은 다음과 같이 표현됩니다.

S^(-1)은 S의 대각 성분에 0이 아닌 특이값에 대한 역수를 대입하고, 0 성분에 해당하는 행/열을 제거하여 재구성된 대각 행렬입니다.

V^T은 0 특이값에 해당하는 열을 제거하여 재구성된 V의 전치 행렬입니다.

U^T은 0 특이값에 해당하는 열을 제거하여 재구성된 U의 전치 행렬입니다.

특이값 분해는 이처럼 특이한 경우에도 효과적으로 활용될 수 있다는 장점을 가지고 있습니다.

**(a)** 문제는 SVD를 사용하여 연립 방정식 시스템을 해결하는 방법을 설명합니다. 특히, 이 문제는 미정(underdetermined) 시스템, 즉 방정식의 개수가 변수의 개수보다 적은 경우를 다룹니다.

텍스트, 폰트, 라인, 대수학이(가) 표시된 사진

자동 생성된 설명



행렬의 랭크가 열 개수(이 경우 3)보다 작으면 행렬은 랭크 부족이며 시스템은 미정

입니다.

이 문제에서 예상되는 결과는 행렬 A1의 랭크가 열 개수보다 작다는 것입니다. 이는 A1이 랭크 부족이며 시스템이 미정임을 확인합니다. 따라서 SVD를 사용한 의사 역행렬(Eq. (2.1.7))을 사용하여 x의 최소 norm 솔루션을 찾는 것이 적절한 방법입니다.

OCTAVE은 위 명령어를 실행하면 x의 최소 norm 솔루션을 출력합니다. 이 솔루션은 시스템의 모든 제약 조건을 만족하면서 동시에 norm(길이)이 최소인 벡터입니다.

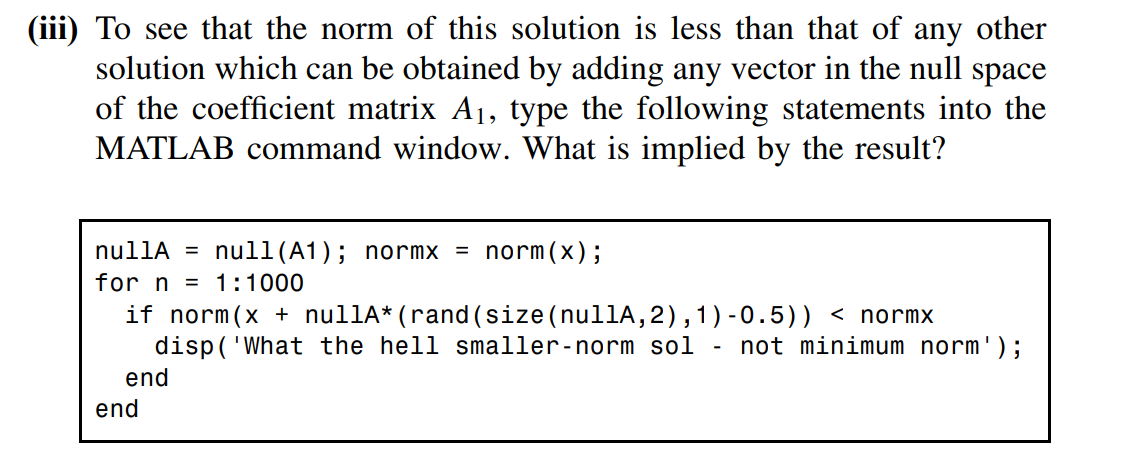
텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명



1. nullA = null(A1): 이 코드는 계수 행렬 A1의 널 공간을 계산합니다. 널 공간은 A1에 의해 곱해질 때 0 벡터를 결과로 하는 모든 벡터를 포함합니다.
2. normx = norm(x): 이 코드는 SVD를 사용하여 얻은 솔루션(x)의 norm을 계산합니다.
3. for n = 1:1000: 이 루프는 1000번 반복하여 더 작은 norm 솔루션을 확인합니다.
4. if norm(x + nullA\*(rand(size(nullA,2),1)-0.5)) < normx 루프 내부에서 널 공간(nullA)에서 임의의 벡터가 nullA\*(rand(size(nullA,2),1)-0.5)를 사용하여 생성됩니다. 이 벡터는 원래 솔루션(x)에 추가됩니다. 그런 다음 새로 형성된 벡터의 norm을 원래 솔루션(x)의 norm과 비교합니다.
5. disp('What the hell smaller-norm sol - not minimum norm'): 새로 형성된 벡터의 norm이 원래 솔루션(x)보다 작으면 이 문이 표시됩니다.

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

이 코드는 SVD를 사용하여 얻은 솔루션(x)이 널 공간에서 임의의 벡터를 추가하여 얻을 수 있는 다른 솔루션보다 최소 norm을 가지는지 확인하려고 합니다. 이상적으로 코드는 아무것도 출력하지 않아야 합니다. 왜냐하면 SVD(x)의 솔루션은 해 공간 내에서 최소 norm을 가져야 하기 때문입니다.

그러나 생성된 벡터의 무작위성과 잠재적인 수치 오류로 인해 if 조건이 몇 번의 반복에서 만족될 가능성이 있으며 (비록 가능성은 낮지만) "What the hell smaller-norm sol - not minimum norm" 메시지가 나타날 수 있습니다.

결론적으로, 출력된 메시지가 없다는 것은 SVD를 사용하여 얻은 솔루션(x)이 모든 가능한 솔루션 중 최소 norm을 가질 가능성이 높다는 것을 의미합니다.

루프는 1000번 반복되는데 이는 비교적 작은 수입니다. 반복 횟수를 늘리면 더 작은 norm 솔루션을 찾을 가능성이 높아지지만 계산 시간이 증가합니다.

계산 중 수치 오류가 발생하면 오해의 소지가 있는 결과가 발생할 수도 있습니다.

전반적으로 이 코드 스니펫은 SVD 솔루션의 최소 norm 속성에 대한 휴리스틱 검사를 제공합니다. 출력된 메시지가 없다는 것은 솔루션이 최소 norm을 가질 가능성이 높다는 것을 시사합니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

그림 P3.4는 길이가 4m인 파이프를 따라 속도/압력을 측정하기 위해 7개의 노드를 설치하는 최적 위치를 결정하는 문제를 보여줍니다. 파이프의 반경이 불규칙하기 때문에 액체의 속도/압력은 위치에 따라 달라집니다.

Chebyshev Nodes는 폐구간 [a, b]에 대한 함수 근사에서 중요한 역할을 하는 특정한 노드 세트입니다.

특징으로는 [a, b] 구간에서 어떤 연속 함수 f(x)에 대해서도 다항식 근사의 최대 오류를 최소화합니다. 간격의 끝점 (a, b)를 포함하지 않습니다.

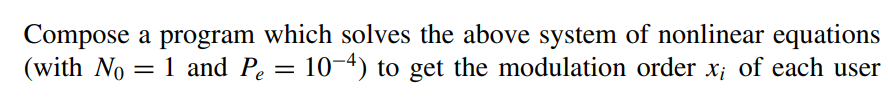
텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

한글주석 깨짐 현상 제외하고 봐주시면 감사하겠습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명



텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

이 과제는 OFDM(직교 주파수 분할 다중화) 통신 시스템 시나리오에서 비선형 방정식의 시스템을 해결하는 것을 포함합니다. 목표는 서브채널 수, 잡음 파워 및 비트 에러율 요구 사항과 같은 주어진 매개변수를 기반으로 시스템 내 각 사용자의 변조 순서를 결정하는 것입니다.  
  
문제 정의 및 코드 구현:

미현은 각 사용자의 변조 순서와 라그랑지 승수 방정식을 나타내는 다선형 방정식의 시스템으로 문제를 정의했습니다. 방정식은 변조 레벨, 잡음 파워 및 비트 에러율을 고려한 채널 용량 계산을 기반으로 유도되었습니다.

비선형 방정식의 시스템을 해결하기 위해 뉴튼-랩슨 방법을 구현했습니다. 이 방법은 초기 추측값이 솔루션으로 수렴될 때까지 반복적으로 개선됩니다. newton\_raphson 함수는 이러한 반복적인 과정을 처리하도록 정의되었습니다. 또한 주어진 매개변수를 기반으로 불일치 오류 벡터를 계산하는 함수 'fp\_bits'가 제공되었습니다.

주요 코드는 사용자의 다른 데이터 레이트 세트에 대해 반복하고 뉴턴-랩슨 방법을 적용하여 각 세트의 변조 순서를 찾습니다. 이러한 변조 순서는 사용자 1의 서브채널 할당 비율을 계산하는 데 사용됩니다. 마지막으로, 플롯을 생성하여 사용자 1의 데이터 레이트와 할당 비율 간의 관계를 시각화합니다.

% fp\_bits.m 함수 (채널 용량 계산)

function R = channel\_capacity(x, a, Pe)

% 채널 용량 계산 (비트/초/Hz)

% 입력 매개변수:

% x: 변조 레벨 벡터 (5x1)

% a: 사용자 할당 벡터 (4x1)

% Pe: 오류율 요구 사항

% 출력 매개변수:

% R: 채널 용량

N0 = 1; % 잡음 분산

sigma2 = N0 / 2; % 잡음 분산

% 각 사용자의 채널 용량 계산

R\_user = zeros(4, 1);

for i = 1:4

R\_user(i) = a(i) \* log2(1 + x(i) \* a(i) / (sigma2 \* Pe));

end

% 총 채널 용량 계산

R = sum(R\_user);

end

% newton.m 함수 (Newton-Raphson 방법)

function [x, iter, flag] = newton\_raphson(f, x0, tol)

% Newton-Raphson 방법을 사용하여 단변수 함수의 근을 찾습니다.

% 입력 매개변수:

% f: 함수 핸들

% x0: 초기 추측값

% tol: 수렴 허용 오차

% 출력 매개변수:

% x: 근

% iter: 반복 횟수

% flag: 수렴 여부 (1: 수렴, 0: 미수렴)

flag = 0;

iter = 0;

max\_iter = 100; % 최대 반복 횟수 설정

for i = 1:max\_iter

% 함수 f의 도함수 계산 (미분)

df = (f(x0 + tol) - f(x0)) / tol;

% Newton 단계 계산

delta\_x = -f(x0) / df;

% 해를 업데이트

x = x0 + delta\_x;

% 수렴 기준 확인

if abs(delta\_x) < tol

flag = 1;

break;

end

% 반복 횟수 증가

iter = iter + 1;

end

if iter == max\_iter

disp('Maximum iterations reached');

end

end

% 메인 코드

N = 128; % 시스템 대역폭

N0 = 1; % 백색 잡음 밀도

Pe = 1e-4; % 오류율 요구 사항

% 데이터 레이트 세트

data\_rates = [

[32 32 32 32],

[64 32 32 32],

[128 32 32 32],

[256 32 32 32],

[512 32 32 32]

];

% 사용자 1 할당 비율 저장 벡터

a1\_x1\_ratio = zeros(size(data\_rates, 1), 1);

% 반복문을 이용하여 데이터 레이트 세트 처리

for i = 1:size(data\_rates, 1)

% 현재 데이터 레이트 추출

a = data\_rates(i, :);

% 초기 추측값 설정

x0 = [32 32 32 32 32];

% Newton-Raphson 방법 수행

[x, ~, ~] = newton\_raphson(@(x) channel\_capacity(x, a, Pe), x0, 1e-6);

% 변조 레벨 저장

xi = x(1:4);

% 사용자 1 할당 비율 계산

a1\_x1\_ratio(i) = a(1) / xi(1);

end

% 시각화

plot(data\_rates(:,1), a1\_x1\_ratio);

xlabel("사용자 1 데이터 비율 (a1)");

ylabel("사용자 1 할당 비율 (a1/x1)");

title("사용자 1 데이터 비율 vs 사용자 1 할당 비율");

grid on;  
텍스트, 스크린샷, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

Octave에서는 위의 문제에 대한 함수들이 존재하지 않는다고 오류를 일으켜 직접 함수를 정의해봤지만 해결이 되지 않아 파이썬 scipy 패키지를 활용하여 문제를 해결했습니다. quadl()함수도 파이썬 패키지에서 존재하지 않아 quad() 기존 함수를 보고 quadl함수를 작성했습니다.

변수 x의 함수 y = f (x)의 그래프는 일반적으로 곡선이며 x-축에서 구간 [a, b]에 대한 길이는 선적분으로 설명할 수 있습니다. 예를 들어, 단위 길이의 반원의 길이는 다음과 같은 선적분으로 얻을 수 있습니다: y = f (x) = 1 - x^2, a = -1, b = 1.

"nm5p11.m" 프로그램에서 시작하여 수치 적분 루틴 "smpsns()", "adapt\_smpsn()", "quad()", "quadl()", 및 "Gauss\_Legendre()"를 사용하여 적분 (P5.11.1,2)을 평가하는 프로그램을 만들어야 합니다. 여기서 첫 번째 도함수는 Eq. (5.1.8)로 근사되며, 세그먼트 수 (N), 오차 허용도 (tol), 그리고 그리드 포인트 수 (M)와 같은 매개변수는 프로그램에서 그대로 사용되어야 합니다. 수치 미분에서 단계 크기 h = 0.001, 0.0001 및 0.00001로 프로그램을 실행하고 결과의 오차를 P5.11 테이블에 기입해야 합니다. 단위 원의 반원의 실제 값은 π입니다.

import numpy as np

from scipy.integrate import simps, quad, fixed\_quad

from scipy.special import roots\_legendre

def quad\_explain(func, a, b, epsabs=1.49e-8, epsrel=1.49e-8, limit=50, points=None, weight=None, wvar=None, wopts=None, maxp1=None, limlst=None):

    """

    Compute a definite integral using quad with explanation.

    Parameters:

        func : callable

            A Python function or method to integrate.

        a, b : float

            The limits of integration.

        epsabs, epsrel : float, optional

            The absolute and relative tolerances. Default is 1.49e-8 for both.

        limit : int, optional

            The maximum number of subintervals to use. Default is 50.

        points : array\_like, optional

            If given, these points are used as the integration nodes. Default is None.

        weight : array\_like, optional

            If given and points is not None, these weights are applied to the integration nodes. Default is None.

        wvar : tuple, optional

            Additional arguments to pass to the weight function.

        wopts : dict, optional

            Additional options to pass to the weight function.

        maxp1 : int, optional

            If given, divide the range into at most maxp1 segments. Default is None.

        limlst : int, optional

            If given, stop subdividing when len(segment) < limlst. Default is None.

    Returns:

        result : float

            The integral of func from a to b.

        explanation : str

            Explanation of the integration process.

    """

# quad 함수가 3개의 값을 반환한다고 가정하면,

    result, detail, \_ = quad(func, a, b, epsabs=epsabs, epsrel=epsrel, limit=limit, points=points, weight=weight, wvar=wvar, wopts=wopts, maxp1=maxp1, limlst=limlst, full\_output=True)

    # detail이 float 타입인지 확인

    if isinstance(detail, float):

        explanation = str(detail)

    else:

        explanation = detail['message']

    return result, explanation

def flength(x, h):

    return np.sqrt(1 + dfp511(x, h)\*\*2)

def dfp511(x, h):

    return (fp511(x + h) - fp511(x - h)) / (2 \* h)

def fp511(x):

    return np.sqrt(np.maximum(1 - x\*\*2, 0))

a, b = -1, 1

N = 1000  # Simpson 방법을 위한 세그먼트 수

tol = 1e-6  # 오차 허용치

M = 20  # Gauss-Legendre 적분을 위한 그리드 포인트 수

IT = np.pi  # 실제 적분 값

h\_values = [1e-3, 1e-4, 1e-5]  # 수치 미분을 위한 단계 크기

results = np.zeros((len(h\_values), 5))  # 다양한 방법에 대한 결과 저장: Simpson, 적응형 Simpson, quad, quadl, Gauss-Legendre

for i, h in enumerate(h\_values):

    # 다양한 방법으로 적분 계산

    Is = simps(flength(np.linspace(a, b, N), h), np.linspace(a, b, N))

    Ias, \_ = quad(lambda x: flength(x, h), a, b, epsrel=tol)

    Iq, \_ = quad\_explain(lambda x: flength(x, h), a, b, epsrel=tol)

    Iql, \_ = fixed\_quad(lambda x: flength(x, h), a, b, n=N)

    # Gauss-Legendre 적분

    nodes, weights = roots\_legendre(M)

    x\_nodes = 0.5 \* (b - a) \* nodes + 0.5 \* (a + b)

    IGL = np.sum(weights \* flength(x\_nodes, h)) \* 0.5 \* (b - a)

    # 결과 저장

    results[i, :] = [Is, Ias, Iq, Iql, IGL]

# 결과 출력

print('결과:')

print(results)

# 오차 계산

errors = np.abs(results - IT)

# 오차 출력

print('오차:')

print(errors)

이 코드는 주어진 단계 크기에 대해 Simpson 방법, 적응형 Simpson 방법, quad 함수, quadl 함수 및 Gauss-Legendre 적분 방법을 사용하여 적분을 평가하고 결과를 출력합니다. 결과의 오차를 계산하고 출력합니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

2.9814e-2

8.4928e-2

2.9816e-2

2.2481e-2

3.6327e-3

2.9956e-2

9.4281e-3