|  |  |  |
| --- | --- | --- |
| 5주차 선형시스템 보고서 | | |
| 제출일 : 2024년 04월 8일 | | 작성자 : 이준용 |
| 구분 | 내용 | |
| 학습 범위와 내용 | 5주차 온라인 강의 내용 | |
| 정리 내용 | **4.1 ITERATIVE METHOD TOWARD FIXED POINT**  **고정점 정리(수축 정리)**: 함수 *g*(*x*)가 정의되고 그 첫 번째 도함수 *g*′(*x*)가 고정점 *g*(*x*)의 주변에서 일정 구간 *I*=[*x*0​−*r*,*x*0​+*r*]에서 연속적으로 존재한다고 가정합시다: *g*(*x*0​)=*x*0​ (4.1.1)  그러면, 첫 번째 도함수 *g*′(*x*)의 절댓값이 1보다 작은 양의 수 *α*보다 작거나 같은 경우, 즉, ∣*g*′(*x*)∣≤*α*<1(4.1.2)  임의의 점 *x*0​∈*I* 에서 시작하는  반복 과정:*xk*+1​=*g*(*xk*​)with*x*0​∈*I*(4.1.3)  함수 *g*(*x*)의 고정점으로 수렴합니다.  이 정리는 본질적으로 함수 *g*(*x*)의 수축 매핑임을 보장하는 조건을 제공합니다. 즉, 거리를 줄이는 효과를 갖고 있으므로 초기 추정값*x*0​에 관계없이 반복 과정이 고정점으로 수렴합니다.  **4.2 BISECTION METHOD**  이분법(Bisection Method)은 수치 해석에서 비선형 방정식의 해를 근사적으로 찾는 방법 중 하나입니다. 이 방법은 구간을 반복적으로 절반으로 나누어 해가 존재하는 구간을 좁혀가는 방식으로 동작합니다.  아래는 이분법의 기본적인 아이디어와 단계를 설명한 것입니다:  1. 구간 설정: 먼저 방정식의 해가 존재할 것으로 예상되는 구간을 선택합니다. 이 구간은 해가 하나 존재하며, 구간의 양 끝점에서 함수 값의 부호가 서로 다른 것으로 정의됩니다.  2. 반복적인 구간 축소: 선택한 구간을 반으로 나눕니다. 중간점에서 함수 값을 계산하고, 이를 통해 새로운 구간을 결정합니다. 새로운 구간은 기존 구간보다 더 작은 길이를 가지며, 방정식의 해가 포함되어 있을 가능성이 높아집니다.  3. 반복: 구간을 충분히 축소할 때까지 위의 단계를 반복합니다. 일반적으로 충분히 작은 구간이 얻어질 때까지 이 과정을 반복합니다.  4. 종료 조건: 구간의 길이가 충분히 작아지거나 원하는 수렴 기준이 충족될 때까지 반복을 계속합니다. 종료 조건에는 원하는 해의 정확도나 반복 횟수 등이 포함될 수 있습니다.  이분법은 간단하고 안정적인 방법으로, 해가 존재하는 구간을 반복적으로 축소함으로써 해를 찾습니다. 하지만 이 방법은 수렴 속도가 다른 방법들에 비해 상대적으로 느릴 수 있습니다.  **4.3 FALSE POSITION OR REGULA FALSI METHOD**  "False Position" 또는 "Regula Falsi Method"는 비선형 방정식의 근을 찾기 위한 반복적인 방법 중 하나입니다. 이 방법은 이분법과 유사하지만, 각 반복 단계에서 구간을 새로운 위치에 대해 수정하는 방법에서 차이가 있습니다.  아래는 False Position 또는 Regula Falsi Method의 기본 아이디어와 단계를 설명한 것입니다:  1. 구간 설정: 먼저 방정식의 해가 존재할 것으로 예상되는 구간을 선택합니다. 이 구간은 해가 하나 존재하며, 구간의 양 끝점에서 함수 값의 부호가 서로 다른 것으로 정의됩니다.  2. 선분 그리기: 선택한 두 점을 통해 해당 선분을 그립니다. 이 선분은 선택한 두 점에 해당하는 함수의 값들을 직선으로 연결합니다.  3. 근의 예측: 선분의 두 끝점에 해당하는 함수 값들의 직선을 이용하여 x 축과의 교점을 찾습니다. 이 교점은 새로운 추정값으로 사용됩니다.  4. 새로운 구간 결정: 이전 구간과 새로운 추정값을 사용하여 새로운 구간을 결정합니다. 이전 구간과 새로운 추정값을 통해 선분을 그린 후, 이전 구간과 새로운 추정값의 함수 값들을 비교하여 새로운 구간을 결정합니다.  5. 반복: 새로운 구간을 얻을 때까지 위의 단계를 반복합니다. 새로운 구간이 충분히 작아질 때까지 이 과정을 반복합니다.  6. 종료 조건: 원하는 정확도가 달성되거나 원하는 수렴 기준이 충족될 때까지 반복을 계속합니다.  False Position 방법은 이분법과 유사하지만, 구간을 선형 보간하는 대신 두 점 사이의 직선을 사용하여 근을 찾습니다. 이로써 수렴이 빠를 수 있지만, 때로는 잘못된 방향으로 수렴할 수 있으므로 주의가 필요합니다.  **4.4 NEWTON(–RAPHSON) METHOD**  뉴턴-랩슨 방법은 비선형 방정식의 해를 찾는 강력한 수치적 기법 중 하나입니다. 특히, 이분법 또는 가짜 위치 등의 방법이 적용되지 않는 경우에 유용합니다.  뉴턴-랩슨 방법의 작동 방식은 다음과 같습니다:   1. **초기화**: 해에 가까운 초기 추정값 *x*0​를 선택합니다. 2. **반복과정**: 다음 반복값을 다음과 같은 공식을 사용하여 계산합니다:*xk*+1​=*xk*​−*f*′(*xk*​)*f*(*xk*​)​(4.4.2) 여기서 *f*′(*xk*​)는 *xk*​에서의 *f*(*x*)의 도함수입니다. 3. **종료**: 원하는 정확도가 달성되거나 수렴 기준이 충족될 때까지 반복합니다.   이 방법은 현재 추정값 *xk*​에서 함수 *f*(*x*)의 접선을 활용하여 해 주변의 지역적 선형성을 이용합니다. 접선을 0으로 설정함으로써 다음 추정값 *xk*+1​을 구합니다.  오차 분석을 위해 함수 *f*(*x*)의 *x*=*xk*​ 주변의 이차 테일러 다항식이 고려됩니다.  뉴턴 방법은 MATLAB의 "newton()" 루틴을 사용하여 구현될 수 있으며, 이는 미분이 제공되지 않은 경우에는 수치 미분을 처리합니다.  뉴턴 방법은 초기 추정값이 해로부터 멀거나 근처에 여러 해가 있는 경우에는 수렴하지 않을 수 있습니다. 또한 함수의 도함수가 항상 사용 가능하지 않을 수 있음에 유의해야 합니다.  **4.5 SECANT METHOD**  Secant Method(세점법)은 뉴턴-랩슨 방법과 비슷한 방식으로 비선형 방정식의 해를 근사하는 수치해석 기법입니다. 뉴턴-랩슨 방법과 달리 Secant Method는 함수의 도함수가 필요하지 않으며, 대신에 두 점을 이용하여 접선을 근사합니다.  다음은 Secant Method의 기본 아이디어와 단계입니다:   1. **초기 추정값 선택**: 초기 추정값으로 두 점을 선택합니다. 보통 이전 추정값 *xk*−1​과 현재 추정값*xk*​을 선택합니다. 2. **접선 근사**: 선택한 두 점을 이용하여 접선을 근사합니다. 이를 통해 다음 추정값 *xk*+1​을 찾습니다. 접선은 다음과 같이 근사됩니다: *f*′(*xk*​)≈*xk*​−*xk*−1​*f*(*xk*​)−*f*(*xk*−1​)​ 3. **반복**: 새로운 추정값 *xk*+1​을 사용하여 위의 단계를 반복합니다. 4. **수렴**: 원하는 수준의 정확도에 도달하거나 수렴 기준을 충족할 때까지 반복합니다.   Secant Method는 뉴턴-랩슨 방법과 비교하여 함수의 도함수를 필요로하지 않기 때문에 더 간단할 수 있습니다. 그러나 수렴 속도가 조금 느릴 수 있습니다.  **4.6 NEWTON METHOD FOR A SYSTEM OF NONLINEAR EQUATIONS**  비선형 연립 방정식의 뉴턴 방법(Newton Method)은 비선형 연립 방정식의 해를 근사하는 수치적 기법 중 하나입니다. 이 방법은 뉴턴 방법을 단일 변수 함수에서 다변수 함수로 확장한 것입니다.  다음은 비선형 연립 방정식의 뉴턴 방법의 기본 아이디어와 단계입니다:  초기 추정값 선택: 해에 가까운 초기 추정값을 선택합니다. 이는 각 변수에 대한 초기 추정값의 벡터로 표현됩니다.  Jacobi 행렬 계산: 초기 추정값에서의 Jacobian 행렬을 계산합니다. Jacobian 행렬은 각 변수에 대한 편미분으로 구성됩니다.  반복적인 선형 방정식 풀기: 현재 추정값에서의 Jacobian 행렬과 함수값을 사용하여 선형 방정식을 해결합니다. 이를 통해 다음 추정값을 계산합니다.  수렴 검사: 원하는 수준의 정확도에 도달하거나 수렴 기준을 충족할 때까지 반복합니다.  비선형 연립 방정식의 뉴턴 방법은 단일 변수 방정식의 경우와 마찬가지로 초기 추정값이 해에 충분히 가깝고 함수가 충분히 매끄럽고 선형 근사가 적용 가능한 경우에 효과적입니다. | |
| 질문 내용 | 1. **질문 없습니다.** | |