수리계획론 0 : 목차

2007년 9월 10일 월요일
오후 4:25

[목차 및 설명]

아래와 같은 영역에서 문제가 나올 것으로 여겨진다.

1. **[선형 계획법] page 4-5, page 293-294**

   출제 경향 : 선형 계획 문제를 하나 제시하고 이 의미를 설명하며 실제로 푸는 문제가 나올 예정이다. 실제 lingo program으로도 구현할 수 있어야 할 듯 하다. Simplex Method는 배우지 않은 관계로 나오지 않을 것으로 여겨진다.

2. **[비선형 계획법 기초] page 15 - 20**

   비선형 계획법은 제약 조건이 있는 경우와 없는 경우로 크게 생각해 볼 수 있다. 두 가지 모두 다 풀 수 있어야 하며, 시험 문제는 교과서의 문제에서 계수 혹은 제약식이 조금 바뀐 형태로 나올 것으로 보인다.

   이 때 알아야 할 개념은 아래와 같다.
   - Gradient Method에 대해서 공부할 것.
   - 주어진 문제를 Lagrange 형태로 바꾸어서 풀 수 있어야 할 것.
   - 이를 통하여 수치 해법(18p)으로 적용하여 말안장점을 찾는 문제가 될 수 있음을 설명할 것.

   그리고 아래의 2가지 형태 문제에 대해서 모두 공부해야 한다.

   Sub #1. 제약 조건이 없는 경우 실전 문제 page 304. [zexmin0.c]

   이 문제는 실제로 코딩하고 답을 풀어보는 문제가 될 예정이다. 이 때

   계수는 아래와 같이 변화된다.

변경 전	변경 후
Iter <= 100	Iter <= 3
Epsil=0.1	Epsil=?
X1=5.0	X1=?
X2=5.0	X2=?

   루프 안쪽 내용은 예전과 같을 것으로 여겨지며, 이 상태에서 각 for문을 수행하였을 때 iter, x1, x2, J 값을 쓰는 문제가 나올 예정이다.

   Sub #2. 제약 조건이 있는 경우 실전 문제 page 304. [zexmin.c]

   이 문제도 마찬가지로 실제로 코딩하고 답을 풀어보는 문제가 될 예정이다. 이 때 계수는 아래와 같이 변화된다.

변경 전	변경 후
Iter <= 100	Iter <= 3
Epsil=0.1	Epsil=?
X1=5.0	X1=?
X2=5.0	X2=?
Ramda=0	Ramda=?

   또한 Lagrange와 관련된 부분이 빠질 확률도 있으나 아직 확정된 것은 아니라고 한다.

   나머지 내용은 이전과 동일하며, 역시 이 상태에서 각 for문을 수행하였을 때 iter, x1, x2, ramda, j 값을 쓰는 문제가 나올 예정이다.

3. **[이산형 동적 계획법] page 29 - 32**

여기에서는 생산/재고 계획 문제를 연속형이 아닌 이산형으로 푸는 문제가 나온다. 계산량이 많은 관계로 실제 문제에 나올지는 미지수이지만, 혹시 모르니 어떻게 푸는지는 연습해 두는 것이 좋다.

4. **[증명 부분] page 51 - 53, 57**

이 부분은 이론 파트로서, 라그랑지 함수를 통해 말 안장점을 풀면 항상 최적해가 나옴을 증명하는 내용이다. 실제 증명 문제는 시험에 나오지 않을 것으로 생각되므로, 참고로 보도록 할 것.

계속해서 수치 해법을 위한 다양한 알고리즘이 제시되는데, 이 중에서 우리는 경사법에 대해서만 다루도록 한다. 역시 시험에 나올 가능성은 희박하지만, 수치 해법을 사용하는 것이 왜 최적해로 수렴하는지에 대한 이론적인 부분이므로 참고 삼아서 보면 좋다.

5. **[연속형 동적 계획법/벨만 방정식] page 97 - 104**

연속형 동적 계획법은 벨만 방정식에 의해서 풀 수 있다. 이 때 생산계획 예제를 바탕으로 문제를 풀게 되며, 수치 해법으로 풀 수 있어야 한다. 코딩 문제는 아마 나오지 않을 것으로 예상된다.

6. **[최적 제어 이론] page 105-127**

이번 학기에서 가장 어려운 내용으로 다음과 같은 내용을 커버하고 풀 수 있어야 한다.

Sub #1. [최적 제어 이론 소개] page 105-116 (optional)

여기에서는 폰트리야긴의 최적 제어 이론에 대한 내용이 주로 다루어진다. 이론적인 부분이 많이 있으며 꽤 어려운 내용에 속한다. 하지만 수업 시간이 깊이 있기 다루지는 않았고, 다음 학기의 주된 강의 내용이라고 하므로 굳이 시간을 내어서 볼 필요는 없으리라 생각된다.

다만 105-110page는 중요한 내용에 속하며, 특히 식 8.15, 8.27을 유도하는 것이 핵심적이므로 한 번씩 읽어보는 편이 좋다.

Sub #2. 생산, 재고, 판매 모형 연속형 예제

생산, 재고, 판매 모형의 연속형 문제가 나오는데, 여기에서는 아래와 같은 단계별 문제가 나오리라 예상된다.

(1) 주어진 생산, 재고, 판매 모형을 "말"로 풀어서 설명하기 (p117-120)

(2) 라그랑지 함수를 이용하여 이를 Saddle Point 문제로 변환시키고 문제의 최적 조건을 제시하기 (p125-127)

(3) 위의 문제에 대해 최적 조건을 프로그래밍하는 문제를 내고 이를 3회 돌려서 결과를 기록하기 (p307-308) [zexhmms.c]

역시 zexhmms.c 프로그램을 조금 변형시킨 형태가 될 것이다. 이 때 kk=5, x0, …, dt, s[k], u[k], ramda[k], iter 이러한 수치들이 모두 바뀌게 될 것이다.

또한 시험 때에는 for문 밖에 있는 문장이 안쪽으로 들어오고 안의 루프는 사라진다. (리포트 5와 같다)

그리고 문제의 조건에 따라 접선의 기울기(2.*c* …)랑 (u[k]-s[k]), 이런 것들도 달라질 수 있다. 즉 "최적 조건을 프로그래밍하시오"

하는 형태의 문제로 나올 수 있다.

(4) 주어진 모형을 벨만의 최적 원리를 이용하여 벨만 방정식까지 유도할 것 ["연속형 동적 계획법"에서 다루었음]

(5) 주어진 벨만 방정식의 내용을 바탕으로 벨만 방정식의 수치 해법을 제시하기 ["연속형 동적 계획법"에서 다루었음]

수리계획론 1 : 선형 계획법

2007년 12월 20일 목요일
오전 8:29

1. 선형 계획법

다음의 선형 계획 예제가 있다.

어떤 공장에서 두 종류의 제품(A, B)을 생산한다. 이 때

<A 제품>
판매 이익 : 4원
작업 1 소요 시간 : 2시간
작업 2 소요시간 : 6시간

<B 제품>
판매 이익 : 3원
작업 1 소요시간 : 4시간
작업 2 소요시간 : 2시간

작업 1 가능시간 : 240시간
작업 2 가능시간 : 240시간

과 같은 조건이 주어져 있다고 하자. 그렇다면 위의 문제 상황은 아래와 같은 수식으로 나타내는 것이 가능하다.

$$Z(\text{원}) = 4(\text{원}/\text{개}) \cdot x_1(\text{개}) + 3(\text{원}/\text{개}) \cdot x_2(\text{개})$$
$$2(\text{시간}/\text{개}) \cdot x_1(\text{개}) + 4(\text{시간}/\text{개}) \cdot x_2(\text{개}) \leqq 240(\text{시간})$$
$$6(\text{시간}/\text{개}) \cdot x_1(\text{개}) + 2(\text{시간}/\text{개}) \cdot x_2(\text{개}) \leqq 240(\text{시간})$$
$$x_1(\text{개}) \geqq 0, x_2(\text{개}) \geqq 0$$

그리고 위의 문제에서 단위를 생략하면 다음과 같은 식으로 쓰는 것이 가능하다.

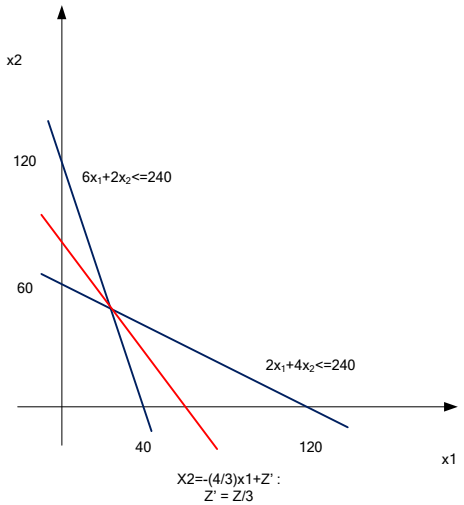
$$Max\, Z = 4x_1 + 3x_2$$

$$s.t.\ 2x_1 + 4x_2 \leqq 240$$

$$6x_1 + 2x_2 \leqq 240$$

$$x_1 \geqq 0, x_2 \geqq 0$$

위의 문제는 다음과 같이 그림으로 나타낸 후 꼭지점의 좌표를 구해서 답을 구할 수도 있다.



Simplex method에 대한 내용이 나오는데, 여기에 대해서는 일단 생략하도록 한다. (실제 시험에서 이렇게 계산하라고 나올지 의문임)

그리고 위의 문제를 Lingo를 이용하여 풀 수도 있다. 이는 아래와 같다.

```
MODEL:
Max = 4*x1 + 3*x2;
6*x1 + 2*x2 < 240;
2*x1 + 4*x2 < 240;
x1 > 0;
x2 > 0;
```



```
end
```

- **LINGO 사용법**

:take ZEX1.LP	파일 실행 준비하라는 뜻
:go	실행하라
:dive ZEX1.TXT	답 저장을 준비하라는 뜻
:solu	답을 저장하라
:quit	빠져나가라

이렇게 하면 DOS 상태에서 zex1.txt 파일이 생성이 되며, 여기에 결과가 담기게 된다. 이 결과는 아래와 같다.

```
VARIABLE VALUE REDUCED COST
X1 24.00000 .0000000
X2 48.00000 .0000000

ROW SLACK OR SURPLUS DUAL PRICE
1 240.0000 1.000000
2 .0000000 .5000000
3 .0000000 .5000000
4 24.00000 .0000000
5 48.00000 .0000000
```

수리계획론 2 : 비선형 계획법 기초

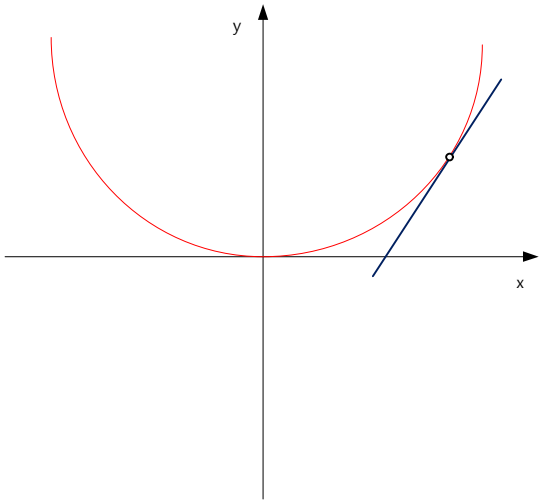
2007년 10월 1일 월요일
오후 4:27

- **비선형 계획법 기초**
  - **기본적인 미분법**

아래는 기본적인 미분법이다. 상식 선에서 알아두도록 하자.

$$\frac{d(x^2)}{dx} = \lim_{\Delta \to 0} \frac{[(x+\Delta)^2 - x^2]}{\Delta} = \lim_{\Delta \to 0} [2x + \Delta] = 2x$$

$$\frac{d(x^n)}{dx} = nx^{n-1}$$



$$x^{i+1} = x^i - \epsilon \frac{dy}{dx}$$

- **비선형 계획법 - 기초 -**

비선형 계획법은 (1) 제약 조건이 없는 경우, (2) 부등호 제약 조건이 있는 경우, (3) 등호 제약 조건이 있는 경우의 3가지로 나누어서 생각해 볼 수 있다. 각각의 경우에 대해서 알아보도록 하자.

1. **제약 조건이 없는 경우 (p15)**

우선 다음의 간단한 2차 계획문제를 검토하여 보자.

$$\min_{x1,x2} J(x_1, x_2) = x_1^2 - x_1 + x_2^2 - x^2 - x_1 x_2$$

위 문제에서 J(x1, x2)값을 최소로 만들어주는 x1, x2를 구하는 것이 목표이다. 이를 구하기 위해서는 J(x1,x2)를 x1, x2에 대해서 각각 미분해 주어서 직접 풀거나 혹은 수치 해법을 이용하여 구할 수 있다.

우선 J(x1,x2)를 각각 x1, x2에 대해 미분하면 아래와 같다.

$$\frac{\partial J}{\partial x_1} = 2x_1 - x_2 - 1 = 0$$

$$\frac{\partial J}{\partial x_2} = 2x_2 - x_1 - 1 = 0$$

이 식은 전형적인 연립방정식이므로, 이를 연립하면

$x_1^*=1$, $x_2^*=1$

을 얻을 수 있다.

혹은 위의 문제에 아래와 같은 수치 해법을 이용하여 구할 수도 있다. (제약 조건이 없기 때문에 라그랑제 법을 쓰지 않고 구함에 유의한다.)

$$x_1^{i+1} = x_1^i - \epsilon \frac{\partial J(x^i)}{\partial x_1}$$
$$x_2^{i+1} = x_2^i - \epsilon \frac{\partial J(x^i)}{\partial x_2}$$

이 때 목적함수 J가 2차 함수로서 계수가 (+)이기 때문에, 임의의 초기 임시해 x^0에 대해 알고리즘이 수렴함을 알 수 있다.

§ 행렬 형태로 계산하기 (optional)

위의 문제는 다음과 같이 행렬, 벡터 형태로도 쓸 수 있다.

$$\min_{x_1,x_2} J(x_1,x_2) = \frac{1}{2}[x_1,x_2]\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - [1,1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

실제로 행렬을 계산해 보면 원래 식과 같음을 쉽게 알 수 있을 것이다.

이 때 Q, r이라는 행렬과 벡터를 정의하면 일반적 형태의 2차 계획 문제로 표현할 수 있다.

$$Q = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \quad r = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\min_x J(x) = \frac{1}{2} x' Q x - r' x$$

이 때 해석적으로 구하기가 힘든 경우에는 아래와 같이 수치 해법으로 구하는 것이 가능하다.

$$x^{i+1} = x^i - \epsilon \frac{\partial J(x^i)}{\partial x}, \quad 0 < \epsilon < 1$$

이 때 gradient vector dJ/dx는 아래와 같이 정의된다. (일종의 행렬 미

분인데, 상당한 고급 수학 기법에 속하므로 그냥 그렇구나 하고 알아두면 될듯하다)

$$\frac{\partial J(x)}{\partial x} = Qx - r$$

- **관련 프로그래밍 문제 : [zexmin.c] 304page**

```
#include <stdio.h>
#include <math.h>

main()
{
int iter;
double X1, X2, J, epsil;
epsil = 0.1;
x1 = 5.0;
x2 = 5.0;

    for (iter = 1; iter <= 100; iter++ )
    {
        x1 = x1 - epsil*(2*x1 - x2 - 1);
        x2 = x2 - epsil*(-x1+2*x2 - 1);
        j = x1 * x1 - x1 + x2 * x2 - x2 - x1 * x2;

        printf("iteration : %d  x1 = %7.4f  x2 = %7.4f
                j = %7.4f      \n", iter, X1, X2, J);
    }
}
```

2. 부등호 제약조건이 있는 경우

아래와 같은 문제가 있다고 하여 보자.

$$\min_{x1,x2} J(x_1,x_2) = x_1^2 - x_1 + x_2^2 - x^2 - x_1 x_2$$

$$s.t. - x_1 - x_2 \leqq -3$$

위와 같은 문제에서는 부등호를 표준형태(어쩌구 저쩌구 <= 0)로 바꾸어 주어야 한다. 그러면 아래와 같이 됨을 알 수 있다.

$$-x_1-x_2+3\leq 0$$

- **라그랑지 함수를 이용하여 해석적으로 푸는 방법**

그러면 아래와 같이 라그랑지 함수를 사용할 수 있다. (이는 기계적으로 외우면 된다. 이는 제약조건과 목적함수를 하나로 합쳐서 주어진 문제를 푸는 방법이라고 생각하면 쉽다.)

$$L(x_1,x_2,\lambda)=[\text{목적함수}]+\lambda(-[\text{제약조건좌변}])$$

이에 맞게 뽑아서 대입하면 아래와 같이 됨을 알 수 있다.

$$L(x_1,x_2,\lambda)=x_1^2-x_1+x_2^2-x_2-x_1x_2+\lambda(3-x_1-x_2)$$

이를 통해서 말안장점(saddle point)문제의 형태로 바뀌게 된다. 그리고 이를 lambda 관점에서 보면 최대화시키는 것이고, x_1, x_2 관점에서 보면 최소화 시키는 문제가 된다.

$$\max_{\lambda\geq 0}\min_{x_1x_2}L(x_1,x_2,\lambda)$$

이를 x_1, x_2의 관점에서 최소화시키면 아래와 같다. 즉 위의 $L(x_1,x_2,\lambda)$식을 각각 x_1, x_2에 대해서 미분하면 된다. 그러면 아래와 같은 식을 얻을 수 있다.

$$\frac{\partial L}{\partial x_1}=2x_1-x_2-1-\lambda=0$$

$$\frac{\partial L}{\partial x_2}=-x_1+2x_2-1-\lambda=0$$

즉 위의 식을 만족하면 x_1, x_2관점에서 최소화가 만족되는 것이다. 위의 식을 x_1, x_2의 입장에서 연립하여 정리하면 아래와 같은 결과를 얻을 수 있다.

$$x_1=\lambda+1,\quad x_2=\lambda+1$$

이를 다시 원래의 $L(x_1, x_2, \lambda)$에 대입하여 정리해보자. 그럼 아래와 같은 식을 구할 수 있을 것이다.

$$L(\lambda)=-\lambda^2+\lambda-1=-(\lambda-\frac{1}{2})^2-\frac{3}{4}$$

이 때 lambda에 대해서는 최대화시키는 것이므로, $\lambda = 1/2 = 0.5$ 이면 최고값을 얻을 수 있다. 이것을 다시 위의 x_1, x_2, λ 에 대입하면 $x_1^*= 1.5$, $x_2^*=1.5$를 얻을 수 있다.

- **수치적으로 푸는 방법**

그런데 위의 문제는 아래와 같이 수치적으로도 풀 수 있다. 라그랑제 함수를 도입하는 것 까지는 같은데, 이를 대입해가면서 해석적으로 푸는 것이 아니라 수치적으로 다음과 같이 풀게 된다.

$$x_1^{i+1}=x_1^i-\epsilon\frac{\partial L(x_1^i,x_2^i,\lambda^i)}{\partial x_1}$$

$$x_2^{i+1}=x_2^i-\epsilon\frac{\partial L(x_1^i,x_2^i,\lambda^i)}{\partial x_2}$$

$$\lambda^{i+1}=\left[\lambda_i+\epsilon\frac{\partial L(x_1^i,x_2^i,\lambda^i)}{\partial \lambda}\right]^+$$

이 때의 0 < ε < 1 이며, 기울기 벡터는 다음과 같이 정의된다.

$$\frac{\partial L(x_1^i, x_2^i, \lambda^i)}{\partial x_1} = 2x_1 - x_2 - 1 - \lambda$$

$$\frac{\partial L(x_1^i, x_2^i, \lambda^i)}{\partial x_2} = -x_1 + 2x_2 - 1 - \lambda$$

$$\frac{\partial L(x_1^i, x_2^i, \lambda^i)}{\partial \lambda} = 3 - x_1^i - x_2^i$$

그리고 19p에 나오는 말안장 그림은 이 중에서 x_1, L의 관점에서 표현한 것이다. 즉 x_1은 최소화되고, L은 최대화가 된다면 주어진 조건을 만족시킴을 알 수 있다. (x_2, L)에 대해서도 마찬가지이다. 따라서 x1의 관점에서는 최소화를 이루는 점을 구하고, L의 관점에서는 최대를 이루면 된다.

즉 팽이를 짜를 때는 판때기 형태로 잘 짤라야 uniqueness가 보장되는 것이다. 즉 이게 보장이 되는 문제만 풀어야 한다!

- **관련 프로그래밍 문제 : [zexmin.c] 304-305page**

```
/* -x1 -x2+3 <= 0   (== x1+x2>=3)  */
#include <stdio.h>
#include <math.h>

main()
{
    double epsil, x1, x2, ramda, l
    int iter;

    epsil = 0.1;
    x1 = 5.;
    x2 = 5.;
    ramda = 0.;

    for( iter = 1; iter <= 100; iter++ )
    {
        x1 = x1 - epsil * (2*x1 - x2 - 1 - ramda);
        x2 = x2 - epsil * (-x1 +2*x2 - 1 - ramda);
        ramda = ramda + epsil * (-x1 - x2 + 3 );
        if( ramda < 0)
            ramda = 0;

        l = x1 * x1 - x1 + x2 * x2 - x2 - x1 * x2
            + ramda * (-x1-x2+3);

        printf("iter= %4d x1 = %7.4f x2= %7.4f
                RAMDA= %7.4f J= %7.4f \n",
                iter, x1, x2, ramda, l);
    }
}
```


3. 등호 제약조건이 있는 경우

아래와 같이 등호 조건이 들어가게 되는 문제를 한 번 풀어보도록 하자.

$$\min_{x1,x2} J(x_1, x_2) = x_1^2 - x_1 + x_2^2 - x^2 - x_1 x_2$$

$$s.t. x_1 + x_2 = 3$$

이 경우 라그랑지를 사용한 말안장점 조건은 아래와 같다.

$$L(x_1, x_2, \lambda) = x_1^2 - x_1 + x_2^2 - x_2 - x_1 x_2 + \lambda(-x_1 - x_2 + 3)$$

그리고 이 때의 lambda를 shadow price라고 한다.

(나머지 푸는 방법은 위의 부등호 제약 조건과 똑같으므로 생략하도록 한다. 교과서 21-22page 참조할 것.)

- **행렬을 이용하여 푸는 방법 (optional)**

위의 문제를 다음과 같은 행렬/벡터 형태로 써 보도록 하자.

$$\min J = \frac{1}{2}[x_1, x_2]\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - [1\ 1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$s.t.\,[1,1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 3$$

이 때에 다음과 같이 행렬, 벡터를 정의하면 라그랑지 함수에 대입할 수 있다.

$$Q = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, r = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$E = [1,1], c = [3]$$

$$\max_{\lambda}\min_{x} L(x,\lambda) = \frac{1}{2}x'Qx - r'x + \lambda'[Ex - c]$$

그리고 이 말안장점 문제의 수치해는 다음의 알고리즘을 적용하여 구할 수 있다.

$$x^{i+1} = x^i - \epsilon\frac{\partial L(x^i, \lambda^i)}{\partial x}$$

$$\lambda^{i+1} = \lambda^i + \epsilon\frac{\partial L(x^i, \lambda^i)}{\partial \lambda}$$

이 때의 gradient vector는 아래와 같다.

$$\frac{\partial L}{\partial x} = Qx - r + E'\lambda$$

$$\frac{\partial L}{\partial \lambda} = Ex - c$$

위의 문제는 목적함수가 2차 형식으로 볼록 함수이고, 제약 조건이 선

형이므로 최적해야 수렴하게 된다.

수리계획론 3 : 이산형 동적 계획법

2007년 12월 20일 목요일
오전 8:33

- **이산형 동적 계획법 (page 29-32)**

  여기에서는 아래와 같은 생산/재고 동적 계획 문제를 이산형으로 푸는 방법에 대해서 생각해 보도록 한다.

$$\min J = \sum_{k=0}^{\bar{k}-1} \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\}$$

$$x_{k+1} = x_k + u_k - s_k$$

  위에서 bar k는 고려할 최대 기간 개수(T)를 의미한다.

  x(t)는 t 시점의 재고함수, bar x(t)는 t시점의 목표 재고함수, u(t)는 t시점의 생산함수, bar u(t)는 t 시점의 목표 생산함수, s(t)는 t 시점의 판매함수를 의미한다. 즉 우리는 위와 같은 생산/재고모형에서 비용을 가장 최소화시키는 재고함수 x(t)와 생산함수 u(t)를 찾고자 하는 것이다. 판매함수 s(t)는 주어져 있다.(given condition)

  - **실전 문제 풀이**

    위의 문제는 연속형 모형으로 만든 후 벨만-방정식으로 풀 수도 있지만, 이산형 문제를 그대로 벨만의 최적원리에 의한 순환관계식으로 푸는 방법도 있다. 계수가 아래와 같이 주어졌다고 가정해보자.

$$h = 0.01,\ c = 1,\ \bar{x}_k = 33,\ \bar{u}_k = 40,\ \bar{k} = 4$$

$$x_0 = 30,\ s_k = 35,\ U = \{35, 36\},\ X = \{30, 31, 32, 33\}$$

    여기에서 U와 X가 set으로 주어져 있는 것은, 생산함수 u_k와 재고함수

x_k는 각각 {35, 36}, {30, 31, 32, 33}만의 값을 가질 수 있음을 의미하는 것이다. 이러한 제한이 없다면 모든 값이 고려되기 때문에 문제를 푸는 것이 (거의) 불가능해 질 것이다.

즉 정리하자면, bar k가 4이므로 u0, u1, u2, u3, x0, x1, x2, x3이 각각 {35, 36}, {30, 31, 32, 33} 중에서 하나의 값을 택할 수 있음을 의미한다. 따라서 모든 경우의 수는 2*2*2*2*4*4*4*4 = 4,096 이 된다. 이 모든 경우를 다 고려하여 최적해를 구하는 것은 쉽지 않은 일일 것이다. 이를 효율적으로 풀어나가는 것이 바로 동적 계획법으로서, "부분적인 최적해를 이용해 전체적인 최적해를 빠르게 구해나가는 기법"으로 생각하면 된다.

일단 위의 문제에 대해서 벨만의 최적원리를 적용하면 다음의 순환 관계식을 얻을 수 있다.

$$J(x_k, k) = \min_{u_k \in U,\ x_k \in X,\ x_{k+1} = x_k + u_k - s_k} J_k$$

$$J_k = \{0.01[x_k - 33]^2 + [u_k - 40]^2\} + J(x_{k+1}, k+1)$$

$$J(33, 4) = 0$$

다시 한 번 정리하여 보면,

$$J(x_k, k) = \min_{(\ldots)} \left[\{0.01[x_k - 33]^2 + [u_k - 40]^2\} + J(x_{k+1}, k+1)\right]$$

를 구하는 것이라고 할 수 있다.

위에 나오는 변수의 의미를 설명하면 아래와 같다.

x_k : k번째 단계에서의 상태변수(재고)
u_k : k번째 단계에서의 제어변수(생산)
s_k : k번째 단계에서의 외생변수(판매)
$J(x_k,k)$: 마지막 단계 bar k부터 k 단계의 특정한 x_k 상태까지의 최소비용

J(x_k,k)의 의미가 조금 어렵게 다가올 수 있는데, 쉽게 설명해 서울에서 부산까지 이동하는 최소비용 경로를 구한다고 생각해 보자. 그렇다면 중간에 많은 경유지점들이 있을 수 있는데, 이 때 부산에서부터 어느 한 특정한 지점, 이를테면 {전주-부산} 경로의 최소비용이라고 생각하면 된다. 물론 전주 대신 대구라든지 진주와 같은 다른 지점이 오는 것도 가능하다.

위에서 J(33,4)=0인 이유는, x_k=33, k=4 인 지점은 이미 최종 시점에 도달하였기 때문에 이 이상으로 무엇인가를 연산할 필요가 없기 때문이다. 그리고 위의 문제는 J(33,4)를 기준으로 거꾸로 구해나가면서 풀어나가면 된다.

위의 문제를 그냥 풀면 xk, uk, xk+1,sk 의 관계 때문에 생각보다 쉽지 않으므로 아래와 같이 xt를 결정변수로 하여 정리해서 풀면 편하다.

$$\min J = \sum_{t=0}^{3} \left\{ 0.01\,(x_t - 33)^2 + (u_t - 40)^2 \right\}$$

$$x_{t+1} = x_t + u_t - s_t$$

St=35로 이미 주어져 있으므로 ut에 대해서 정리하면

$$u_t = x_{t+1} - x_t + 35$$

그러면 아래와 같이 식을 정리할 수 있다.

$$\min J = \sum_{t=0}^{3} C_t$$

$$C_t = 0.01\,(x_t - 33)^2 + (x_{t+1} - x_t - 5)^2$$

그리고 각 시점별로는 아래를 구하면 된다.

$$C_t[x_t, t] = \min\left\{ 0.01\,(x_t - 33)^2 + (x_{t+1} - x_t - 5)^2 + C_{t+1}[x_t, t+1] \right\}$$

이를 바탕으로 t=4인 시점에서부터 거꾸로 구해주면 된다. 우선 t=4는 일종의 경계값으로서, 이 때 C_t =C_4= 0, x_4=33이 되어야 한다. (J(33,4)=0이므로)

t=3 시점을 보도록 하자. 이 때 x_3 = {30, 31, 32, 33}, x_4={33}을 가질 수 있으므로 이를 고려하면 아래와 같이 풀 수 있다.

(1) C_3 (x_3={30, 31, 32, 33} 일 때)
X_3 = 30, t=3
C_3 = 0.01*(x_3 - 33)^2 + (x_4 - x_3 - 5)^2
= 0.01*(30 - 33)^2 + (33 - 30 - 5)^2 = 4.09

X_3 = 31, t=3
C_3 = 0.01*(31 - 33)^2 + (33 - 31 - 5)^2 = 9.04

X_3 = 32, t=3
C_3 = 0.01*(32 - 33)^2 + (33 - 32 - 5)^2 = 16.01

X_3 = 33, t=3
C_3 = 0.01*(32 - 33)^2 + (33 - 33 - 5)^2 = 25.01

이를 바탕으로 t=2 시점을 구할 수 있다. 이 때는 x_2={30, 31, 32, 33}과 x_3={30, 31, 32, 33}을 **모두** 고려해 주어야 한다.(즉 모두 16경우를 고려한다.)

(1) C_2 (x_2={30, 31, 32, 33}, x_3={30, 31, 32, 33} 일 때)
X_2 = 30, t=2
C_2 = min(
0.01*(30 - 33)^2 + (30 - 30 - 5)^2 + 4.09 = 29.18
0.01*(30 - 33)^2 + (31 - 30 - 5)^2 + 9.04 = 25.13
0.01*(30 - 33)^2 + (32 - 30 - 5)^2 + 16.01 = 25.1
0.01*(30 - 33)^2 + (33 - 30 - 5)^2 + 25.01 = 29.1
) = 25.1

... 이런 식으로 모두 구하면 된다. 관련된 내용은 page 31-32 에 있으니 참조하도록 하면 된다.

수리계획론 4 : 증명 부분

2007년 12월 20일 목요일
오전 8:33

- **비선형 계획법 (이론)**

  여기에서는 라그랑지 함수로 비선형 계획법을 풀었을 때 최적해가 항상 보장됨을 증명한다.

1. **최적조건 및 쌍대이론 (page 51-53)**

   다음의 비선형 계획 문제를 보자.

$$\min_x J(x)$$
$$g(x) \leqq 0$$

   이 문제의 라그랑지 함수는 아래와 같이 정의될 수 있다. (이 방법은 그대로 외우도록 한다. 즉 목적함수와 제약식이 어떻게 들어가는지 알아두도록 하자.)

$$L(x,\lambda) = J(x) + < \lambda, g(x) >$$

   이의 최적 조건은 아래와 같다.

   - **말 안장점 정리 I**

     만약 x*, λ* 가 다음의 조건을 충족시키면

$$L(x^*,\lambda) \leqq L(x^*,\lambda^*) \leqq L(x,\lambda^*),\ \lambda \geqq 0$$

     (1) <λ*, g(x*)>=0 이고

(2) x*는 원래 문제의 최적해가 된다.

- **(1)의 증명**

이를 증명하기 위해서 라그랑지 함수를 직접 대입해보도록 하자. 그러면 아래와 같은 결과를 얻을 수 있다.

$$J(x^*)+\langle\lambda,g(x^*)\rangle\leqq J(x^*)+\langle\lambda^*,g(x^*)\rangle\leqq J(x)+\langle\lambda^*,g(x)\rangle$$
[식2]

그렇다면 왼쪽의 관계식에서 J(x*)를 소거하면 아래와 같은 관계를 얻을 수 있다.

$$\langle\lambda,g(x^*)\rangle\leqq\langle\lambda^*,g(x^*)\rangle$$
[식3]

이 때 λ = 0으로 선택하면 위의 식의 좌변은 0이 되고, 우변은 적어도 0보다는 크게 된다. 따라서

$$0\leqq\langle\lambda^*,g(x^*)\rangle$$

이다. 그리고 [식3]은 임의의 λ에 대하여 성립하므로, 충분히 큰 λ >= 0을 선택한다면 g(x*) <= 0일 때만 성립한다. 그리고 λ* >= 0, g(x*)<=0 이므로 위의 부등식은 등호일때만 성립한다. 따라서

$$0\leqq\langle\lambda^*,g(x^*)\rangle\leqq 0$$

이고 이를 통해

$$\langle\lambda^*,g(x^*)\rangle=0$$
[식4]

를 얻을 수 있다.

- **(2)의 증명**

[식2]의 우변에서 아래와 같은 관계를 얻을 수 있다.

$$J(x^*)+\langle\lambda^*,g(x^*)\rangle\leqq J(x)+\langle\lambda^*,g(x)\rangle$$

이 때 만약 x가 실현 가능해라면 문제의 원래 조건에 의해 g(x) <=0이며, 따라서 위의 식과 [식4]에 의해 아래와 같은 결과를 얻을 수 있다.

$$\begin{aligned}J(x^*)+\langle\lambda^*,g(x^*)\rangle&\leqq J(x)+\langle\lambda^*,g(x)\rangle\\ J(x^*)&\leqq J(x)+\langle\lambda^*,g(x)\rangle\\ &\leqq J(x)\end{aligned}$$

이것이 모든 x에 대해서 성립하므로, 따라서 x*는 최적해이다.

- **수치 해법 알고리즘 (page 57)**

제약 조건이 없는 다음과 같은 최소화 문제가 있다.

$$\min_x J(x)$$

이 때 x는 실수이고 J는 미분 가능하다. 그러면 J(x+Δx)의 근사식을 테일러 급수로 다음과 같이 나타낼 수 있다.

$$J(x+\Delta x)=J(x)+\frac{\partial J}{\partial x}\cdot\Delta x+0(\|\Delta x\|^2)$$

이 때에 0(||Δx||^2) 는 급수의 항 중에서 Δx가 2차 이상인 항들의 합을 나타낸다. 그러면 Δx를 다음과 같이 결정하자.

$$\Delta x = -\epsilon \frac{\partial J}{\partial x},\ \epsilon > 0$$

이를 대입하여 정리하면 아래와 같다.

$$J(x+\Delta x) = J(x) - \epsilon \| \frac{\partial J}{\partial x} \|^2 + 0(\|\Delta x\|^2)$$

이 때에 Δx가 충분히 작으면 $0(\|\Delta x\|^2)$는 0에 수렴하며, 충분히 작은 ε에 대하여 다음이 성립한다.

$$J(x+\Delta x) \leqq J(x)$$

즉 Δx를 위와 같이 결정하면 J는 감소한다. 이를 통해서 x를 반복적으로 개선하면 항상 최소해에 도달하게 된다고 생각할 수 있다.
이 반복 개선의 식을

$$x \rightarrow x^i,\ x + \Delta x \rightarrow x^{i+1}$$

로 설정하면, 반복 개선 알고리즘을 다음과 같이 정의할 수 있다.

$$x^{i+1} = x^i - \epsilon \frac{\partial J(x^i)}{\partial x}$$

즉 위와 같은 알고리즘을 사용하면 최소해에 도달하게 되는 것이다. (다만 이것이 최적해라는 의미는 아니다. 초기치 x_0에 따라서 달라질 수 있다)

* 방향 도함수에 대한 식은 page 57, (6.41)에 있으니 참고하여 보도록 할 것.

수리계획론 5 : 연속형 동적 계획법

2007년 12월 20일 목요일
오전 8:35

- **연속형 동적 계획법**

1. **벨만 방정식 (이론 파트) (97-98 page)**

   다음의 동적 최적화 문제를 검토하여 보자.

$$\min J(x,t) = \int_t^T f^0(x,u,\tau)d\tau + \phi(x_T, T)$$

$$\dot{x}(\tau) = f(x,u,\tau),\ \dot{x}(\tau) = \frac{dx(\tau)}{d\tau}$$

$$x(0) = x_0,\ 0 \leqq t \leqq \tau \leqq T$$

   첫 번째 식은 0에서부터 T까지의 기간에 걸쳐 발생하는 비용 J를 최소화 시키는 문제로서, 결정변수는 u이다. 두 번째 식은 이에 대한 미분방정식이다.

   위의 식을 벨만의 최적 원리에 의해 나타내면 다음과 같다. (증명은 일단 생략하고 외우도록 하자)

$$J^*(x,t) = \min_u \{ f^0(x,u,t)\Delta t + J^*(x+\Delta x, t+\Delta t) \}$$

   간단하게 설명하자면 3장에서 배웠던 이산형 동적 계획법을 수식화하여 나타낸 것이라고 생각하면 된다. 여기에서 $f^0(x, u, t)\Delta t$는 중간지점까지의 비용 및 거리를 의미하고, $J^*(x+\Delta x, t+\Delta t)$는 그 이후 부분을 의미한다고 보면 된다. 그렇게 우변에서 min을 구하여 왼쪽에 기억시킨다고 생각하면 쉽다.

원래 주어진 최적화 문제의 최소의 값인 J를 구한다는 것은, 벨만 식으로 나타낸 { } 내의 부분이 최소가 되게 하는 것과 같은 의미이다.

위의 식 중에서 J*(x+Δx, t+Δt)는 Δx, Δt 이후의 최적비용을 나타내는데, 테일러 급수로 근사적으로 나타낼 수 있다.

$$J(x+\Delta x,t+\Delta t)=J(x,t)+\frac{\partial J}{\partial x}\Delta x+\frac{\partial J}{\partial t}\Delta t+\ldots$$

따라서 벨만 방정식을 테일러 전개하여 표현하면 아래와 같이 나타낼 수 있다.

$$J^*(x,t)=\min u_u\left\{f^0(x,u,t)\Delta t+J^*(x,t)+\frac{\partial J}{\partial x}\Delta x+\frac{\partial J}{\partial t}\Delta t+\ldots\right\}$$

이 때 괄호 안의 J*(x, t)와 괄호 내의 J*(x, t)는 소거 가능하므로 다시 한 번 정리하고 양변을 Δt 로 나누어 주도록 하자. 그러면 아래와 같은 식을 얻을 수 있다.

$$0=\min_u\left\{f^0(x,u,t)+\frac{\partial J}{\partial x}\frac{\Delta x}{\Delta t}+\frac{\partial J}{\partial t}+\ldots\right\}$$

Δt가 0으로 수렴함에 따라 Δx/Δt는 연속형으로 바뀐다.

$$\lim_{\Delta t\to 0}\frac{\Delta x}{\Delta t}=\frac{x(t+\Delta t)-x(t)}{\Delta t}=\frac{dx(t)}{dt}=\dot{x}(t)$$

즉 위와 같이 x'(t)로 나타낼 수 있고, (...)항은 0이 되므로 이를 정리하고, x'(t) = f(x, u, t)를 대입하여 이항하면 아래와 같이 나타낼 수 있다.

$$-\frac{\partial J}{\partial t}=\min_u\left\{f^0(x,u^*,t)+\frac{\partial J}{\partial x}f(x,u^*,t)\right\}$$

이 식이 바로 벨만방정식의 일반 수식이다.

2. 벨만 방정식 생산 계획 예제 (99-100 page) [중요]

벨만 방정식을 이용하여 풀 수 있는 대표적인 예제는 생산/재고 함수 모델이다. 앞서 이산형 동적 계획법에서도 제시되었지만, 앞으로도 지속적으로 계속 나오는 문제이니 잘 보아두도록 하자.

생산-재고 모형에 있어서 관련된 비용 함수들은 2차 함수 형태로 표시할 수 있다. (원래는 v 자 형태의 1차 함수인데, 해가 쉽게 나올 수 있도록 2차 함수 형태로 만들어서 푸는 것이다) 이 때 총 비용 J는 아래와 같이 구할 수 있다.

$$\min J=\int_t^T\{h[x(\tau)-\bar{x}]^2+c[u(\tau)-\bar{u}]^2\}d\tau \quad (1)$$

$$\dot{x}(\tau)=u(\tau)-s(\tau),\ x(0)=x_0 \quad (2)$$

이 때 위의 변수들은 각각 아래와 같이 정의할 수 있다.

x : 재고, $\bar{x}$: 목표재고
u: 생산율, $\bar{u}$: 최적조업도
s: 판매율, t: 시점
h: 재고비용계수, c: 생산비용계수

이 때의 일반적 벨만 방정식은 아래와 같이 나타낼 수 있다.

$$-\frac{\partial J}{\partial t}=\min_u\left\{f^0(x,u,t)+\frac{\partial J}{\partial x}f(x,u,t)\right\} \quad (3)$$

$$\dot{x}(\tau)=f(x,u,t)$$

그리고 f^0, f는 아래와 같이 나타내는 것이 가능하다.

$$f^0(x,u,t) = h[x(t) - \bar{x}]^2 + c[u(t) - \bar{u}]^2$$

$$f(x,u,t) = \dot{x}(t) = u(t) - s(t)$$

이를 정리하여 (3)의 식에 대입하면 아래와 같은 식을 얻을 수 있다.

$$-\frac{\partial J}{\partial t} = \min_u \left\{ h[x(t) - \bar{x}]^2 + c[u(t) - \bar{u}]^2 + \frac{\partial J}{\partial x}[u(t) - s(t)] \right\} \quad (4)$$

그리고 u를 최소화 시키는 것이 문제의 목적이 된다.

이는 미분의 특성을 이용하면 쉽게 풀 수 있다. 즉 { }내의 부분을 u로 미분한 값이 0이 된다면 해당 지점에서 위의 함수는 최소가 될 것이다.

따라서 위의 함수에서 { } 부분을 u로 미분하고 이를 0으로 놓으면 된다. 이를 구하면

$$\frac{\partial \{\ \}}{\partial u} = 2c[u(t) - \bar{u}] + \frac{\partial J}{\partial x} = 0$$

이 식을 u(t)에 대해서 정리해 보자. 이 때 아래 식을 만족하는 u(t)는 최적값이므로, u*(t)로 나타내면 아래와 같다.

$$u^*(t) = \bar{u} - \frac{1}{2c}\frac{\partial J}{\partial x} \quad (5)$$

이 식을 다시 (4)의 u(t)에 거꾸로 대입해 보자. 그렇다면 아래와 같이 정리할 수 있다.

$$-\frac{\partial J}{\partial t} = h[x(t) - \bar{x}]^2 + c[\bar{u} - \frac{1}{2c}\frac{\partial J}{\partial x} - \bar{u}]^2 + \frac{\partial J}{\partial x}[\bar{u} - \frac{1}{2c}\frac{\partial J}{\partial x} - s(t)]$$

이 때 $\min_u$가 사라지는 이유는 이미 최적해에 도달했기 때문이다. 그리고 이 식은 아래와 같이 다시 정리할 수 있다.

$$-\frac{\partial J}{\partial t} = -\frac{1}{4c}[\frac{\partial J}{\partial x}]^2 + \frac{\partial J}{\partial x}[\bar{u} - s(t)] + h[x(t) - \bar{x}]^2 \quad (6)$$

즉 위의 식에 의해서 각 시점에서의 최적 비용 J*를 구할 수 있고, 이에 의해 얻은 u*은 다시 (5)식에 의해서 구할 수 있다.

그런데 위의 벨만 방정식은 편미분 방정식으로서 해를 해석적인 방법으로 구하기가 불가능하다. 따라서 수치적인 방법으로 구할 수 밖에 없다.

3. 벨만 방정식의 수치 해법: 유한 차분법 (중요) (page 101)

위의 문제 (1), (2)의 최적 조건인 벨만 방정식 (6)을 수치 해법으로 구하기 위해서 bar u와 bar x 값을 주어야 한다. 여기에서는

$$\bar{u} = s(t),\ \bar{x} = 0$$

으로 놓도록 한다. 이 경우 (6)에 대입해서 정리하면 아래와 같은 식을 얻을 수 있다.

$$-\frac{\partial J}{\partial t} = -\frac{1}{4c}\left[\frac{\partial J}{\partial x}\right]^2 + h[x(t)]^2$$
$$J(T,x) = 0,\ J(t,0) = 0 \qquad (7)$$

위의 최적조건은 비선형 편미분 방정식으로서, 계산하기 쉽게 하기 위해서는 이산형 모형으로 변경시키는 것이 좋다. 이 때 구간 [0, T]를 동일한 세분구간 Δt로 나누고, 각 구간 분할 점에서의 해의 근사값을 구하도록 한다.

우선 다음과 같이 등분하고, 새로운 첨자를 정의하도록 하자.

$0 \leqq t \leqq T$: Δt로 등분

$x_0 \leqq x \leqq x_T$: Δx로 등분

$$t = k\Delta t$$
$$x = x_0 + i\Delta x$$

그렇다면 이 경우 J(t, x)의 미분은 다음 근사식으로 나타낼 수 있다. 조금 어렵게 보이기는 하지만, 그냥 일반적인 미분의 정의를 그대로 이요한 것이며 이를 위에서 새로 정의한 첨자로 나타낸 것 뿐이니 크게 어려운 것은 없다.

$$\frac{\partial J}{\partial t} \cong \frac{J(x,t+\Delta t)-J(x,t)}{\Delta t} = \frac{J_{i,k+1}-J_{i,k}}{\Delta t}$$

$$\frac{\partial J}{\partial x} \cong \frac{J(x+\Delta x,t+\Delta t)-J(x,t+\Delta t)}{\Delta x} = \frac{J_{i+1,k+1}-J_{i,k+1}}{\Delta x}$$

이 때 아래의 식에서 보면 J(x+Δx, t+Δt) - J(x, t+Δt) 로 되어 있는 것이 보일텐데, 엄밀히 원래의 미분 정의를 따르자면 J(x+Δx, t) - J(x, t)로 계산해야 하나 이 경우 계산이 상당히 힘들어지게 되므로 Δt만큼을 t에 더하여 근사식을 구하는 팁을 잊지 말도록 하자.

여하튼 (7)의 편미분 방정식에 위의 두 근사식을 대입하여 정리하면 아래와 같은 식을 얻을 수 있다.

$$J_{i,k} = J_{i,k+1} - \frac{1}{4c}\frac{\Delta t}{(\Delta x)^2}[J_{i+1,k+1}-J_{i,k+1}]^2 + h\Delta t[i\Delta x]^2$$

여기까지 정리하면 시험 문제를 풀 수 있다.

- **실제 수치 해법 코드 (page 101-102)**

여기는 보너스 숙제에 대한 부분인데, 실제 시험에서는 나오지 않을 것

으로 보인다. 하지만 실제로 구현할 때 어떻게 구현하는지에 대한 내용이므로 참고하자.

```
/* Homework6.c */

#include <stdio.h>
#include <math.h>
#define maxn  101

int main()
{
	double h, c;
	double t0, tmax, dt, x0, xmax, dx;
	int tnum, tpresent, xnum, xpresent;
	int i, k;
	double j[maxn][maxn];

	FILE *ofp;
	ofp = fopen("hw6_output.txt", "w");

	h = 0.1;
	c = 1;

	/*
	 * t0, tmax는 t의 첫값(0)과 끝값(T)를 의미합니다. 문제에서 주어진 대로 0, 10을 입력합니다.
	 * tnum은 몇 개로 분할할 것인지 그 개수를 의미하고, tpresent는 출력할 열 개수를 의미합니다.
	 */

	t0 = 0;
	tmax = 10;
	tnum = 100;
	tpresent = 10;

	dt = (tmax-t0)/tnum;

	/*
	 * x0, xmax는 x의 첫값(x0)과 끝값(xT)를 의미합니다. 문제에서 주어진대로 0, 10을 입력합니다.
	 * xnum은 몇 개로 분할할 것인지 그 개수를 의미하고, xpresent는 출력할 행 개수를 의미합니다.
	 */

	x0 = 0;
	xmax = 10;
	xnum = 50;
	xpresent = 10;

	dx = (xmax-x0)/xnum;

	for( i=0; i<maxn; i++ )
		for( k=0; k<maxn; k++ )
			j[i][k] = 0.0;

	// (1) setting J(T, x) = 0
	for( i=0; i<=xnum; i++)
		j[i][tnum] = 0.0;

	// (2) setting J(t, 0) = 0
	for( k=0; k<=tnum; k++)
		j[0][k] = 0.0;
```

```
        // (3) 수식 7.22를 계산한다. 이 때 i+1 대신 i-1을 사용하는 것은 행의 위치가 뒤바뀌어 있기
        때문이다.
        for( k=tnum-1; k>=0; k-- )
        {
                for( i=1; i<=xnum; i++)
                {
                        j[i][k] = j[i][k+1] - 1.0 / (4 * c) * (dt / (dx*dx)) * (j[i-1][k+1] - j[i][k+1]) *
                        (j[i-1][k+1] - j[i][k+1]) + h * dt * (i*dx)*(i*dx);
                }
        }

        for( i=0; i<=xpresent; i++ )
        {
                for( k=0; k<=tpresent; k++ )
                {
                        printf("%2.2f ", j[i*xnum/xpresent][k*tnum/tpresent]);
                        fprintf(ofp, "%2.2f ", j[i*xnum/xpresent][k*tnum/tpresent]);
                }
                printf("\n");
                fprintf(ofp, "\n");
        }

        fclose(ofp);

        return 0;
}
```


수리계획론 6 : 최적 제어 이론

2007년 12월 20일 목요일
오전 8:35

- **최적 제어 이론**

1. **최적 제어 모형의 의미 설명 (page 120)**

   우선 page 120에 있는 최적 제어 모형을 제시하고, page 117-118에 걸쳐서 이에 대한 내용을 설명하고자 한다.

   우선 생산, 재고 및 판매의 관계를 미분방정식으로 설명하려면, 우선 주어진 문제를 아래와 같은 수식으로 나타내어 보도록 하자.

$$\min J = \int_0^T \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\} dt$$

$$\dot{x}(t) = u(t) - s(t),\ x(0) = x_0$$

   이 때 x'(t)의 생산 재고판매의 관계를 미분방정식으로 나타내면 아래와 같이 나타내는 것이 가능하다.

$$\frac{dx(t)}{dt} = u(t) - s(t) = \lim_{\Delta \to 0} \frac{x(t+\Delta) - x(t)}{\Delta} = u(t) - s(t)$$

   이 때 Δ를 빼고 그 의미를 설명하는 것이 가능하다. 즉 Δ를 오른쪽으로 이항하여 곱하면 다음과 같은 식을 얻을 수 있다.

$$x(t+\Delta) = x(t) + \Delta[u(t) - s(t)]$$

   그리고 이를 k를 이용하여 정리하면

$$x_{k+1} = x_k + \Delta \cdot u_k - \Delta \cdot s_k$$

이 되는 것이다.

혹은 거꾸로 정차 방정식에서 미분 방정식을 만들어 설명할 수도 있다. 우선 연속형 모형, 즉 생산, 재고, 판매 사이의 정차 방정식은 아래와 같다.

$$x_{k+1} = x_k + u_k - s_k \quad (1)$$

위의 식은 k 시점에서 기초 재고(x_k)에 생산(u_k)을 더하고 판매(s_k)를 빼면 다음 시점의 기초 재고(x_{k+1})이 됨을 의미한다. 이 때

- x_k : k 시점에서의 재고(단위 : 개)
- u_k : k 시점에서의 생산율(단위: 개/단위기간)
- s_k : k 시점에서의 판매율(단위: 개/단위기간)

이므로, 식을 더욱 정밀하게 나타내기 위해서

- Δ : 단위기간

을 추가로 도입하여 설명하면 아래와 같이 나타내는 것이 가능하다.

$$x_{k+1} = x_k + \Delta \cdot u_k - \Delta \cdot s_k \quad (2)$$

이 때 계획기간 [0, T]를 작은 구간 Δ씩 K개로 나누면($0<\Delta<=1$), $t=k\cdot\Delta$, $t\in[0,T]$이다. 그리고 계획 기간에서 현재의 시점을 0으로 놓으면 계획 종료 시점은 T이다. 이 때 (2) 수식을 다음과 같이 정리할 수 있다.

$$\lim_{\Delta\to 0} \frac{x_{k+1} - x_k}{\Delta} = u_k - s_k$$

이 식에서 좌변은 미분의 정의와 같으며, 이 수식은 다음과 같이 나타낼 수 있다.

$$\frac{dx(t)}{dt} = u(t) - s(t)$$

$$\dot{x}(t) = u(t) - s(t) \quad (3)$$

따라서 미분 방정식으로 설명할 수 있음을 알 수 있다.

그리고 (3)은 다음과 같이 적분으로 변환하여 볼 수도 있다.

$$\int_0^t \dot{x}(\tau)d\tau = \int_0^t [u(\tau) - s(\tau)]d\tau$$

이 때 위 식의 좌변을 적분한 후 정리하면 다음과 같은 수식이 된다.

$$x(t) = x(0) + \int_0^t [u(\tau) - s(\tau)]d\tau$$

즉 같은 대상이지만 이를 어떻게 인식하느냐에 따라서 다양하게 표현할 수 있는 것이다.

§ 교수님 해설 I (교수님 설명)

위 문제의 (2) 유도과정을 좀 더 풀어서 써 보도록 하자.

예를 들어 k시점의 재고(x_k)가 100개, k시점의 생산율(u_k)이 50개/달, k시점의 판매율(s_k)이 30개/달 이라고 해 보자. 그러면 아래와 같이 놓고 다음 시점의 재고를 구할 수 있다.

$$x_1 = x_0 + \Delta \cdot u_0 - \Delta \cdot s_0$$

이 때 단위 Δ 가 1달이라고 해 보자. 그럼 기말 시점의 재고는 100 + 50 - 30 = 120 이 된다. 즉 1월 말에는 120개가 창고에 남게 된다.

그런데 이 때 이 미분방정식은 단위 Δ가 0.5, 0.1씩으로 줄어들어도 성립이 되게 된다.

만약 Δ=0.5일 때는 어떻게 되겠는가?

1월 1일에 100개 있었다. 그리고 1달에 50개 생산(u_k)하는 속도로 0.5달 동안 생산을 했다. 그러면 기말까지 50 * 0.5 = 25 가 창고로 들어갈 것이다. 그리고 1달에 30개 판매(s_k)하는 속도로 0.5달 동안 판매를 했다. 그러면 30 * 0.5 = 15 개만큼 창고에서 빠졌다. 이를 종합하면 100 + 25 - 15 = 110 이 된다. 이것이 1월 15일 시점의 재고량이 되는 것이다.

그렇다면 1월 16일을 생각해 보자. 110을 받아서 똑같이 연산을 수행해 주면 되며, 이 때는 110 + 50 * 0.5 - 30 * 0.5 = 120 이 된다. 즉 위에서 단위를 1로 잡으나 0.5로 잡으나 특정 시점에 도달하면 그 값은 모두 같음을 알 수 있다. 따라서 이를 일반화시키면 Δ -> 0 으로 갈 때에도 마찬가지가 된다.

혹은 아래와 같이 표현할 수 있다.

$$\min J = \int_0^T \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\} dt$$

위의 식에서 bar term들을 잠깐 0으로 놓고 생각하자. 그러면

$$\lim_{\Delta \to 0} \sum_{t=0}^{T} \{hx(t)^2 + cu(t)^2\} \Delta$$

와 같다. 이를 다시 옮겨 쓰면

$$\lim_{\Delta \to 0} \sum_{k=0}^{\bar{k}-1} \{hx_k^2 + cu_k^2\} \Delta$$

와 같이 나타낼 수 있다. 이 때 bar k를 그냥 쓰는 것 보다는 bar k-1을 쓰는 것이 보다 정확하다. 왜냐하면 hkx는 1월 한달 동안의 재고비가 되고, cuk는 1월 한달 동안의 생산비가 되기 때문이다. 이 때 t는 0~11까지 총 12번이므로 k 역시도 0~11까지 만 해야 하고, 여기에 맞추어서 k-1을 하는 것이다.

2. 적분 형태 목적 함수 (page 118-120) [중요]

Case 1. 이산형 -> 적분형 (적분 형태의 목적 함수로 설명하라는 시험 문제가 나올 수도 있다.)

위에서 주어진 문제는 적분 형태의 목적함수로도 생각해 볼 수 있다.

우선 재고/생산비용 함수에서 특정 시점의 비용을 여러 기간에 걸친 비용의 합으로 나타내어 보도록 하자. 그렇다면 아래와 같은 이산형 모형을 유도할 수 있다.

[이산형 모형]

$$\min J = \sum_{k=t}^{k-1} \{h[x_k - \bar{x}_k]^2 + c[u_k - \bar{u}_k]^2 + c_1(k) + c_2\} \Delta$$

이 때 계획기간은 [0, bar k]이며 매 시점 k에서의 재고 관련비용과 생산 관련 비용을 계획 기간 동안 합하여, 총 비용을 최소화 시키고자 하는 것이 목적이다. 이 때 실제 계획 기간은 bar K * Δ이다.

위의 수식은 Δ->0으로 수렴하면 다음과 같은 적분 형태가 된다. 즉 아래는 연속형 모형이다.

[연속형 모형]

$$\min J = \int_0^T \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\} dt$$

즉 단위 기간을 정하는 방법에 따라 적분 기간 bar k, T가 달라진다. 그러나 적분한 총액은 같다. 이 때 t = k·Δ이다.

Case 2. 연속형 -> 이산형

아래의 식은 연속형 생산 재고 모형이다.

$$\int_0^T \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2 + c_1 + c_2\} dt$$

이를 이산형 생산 재고 모형으로 바꾸면 아래와 같다.

$$J = \lim_{\Delta \to 0} \sum_{k=0}^{\bar{k}-1} \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\} \cdot \Delta$$

$$\simeq \{h[x_0 - \bar{x}_0]^2 + c[u_0 - \bar{u}_0]^2\} \cdot \Delta$$
$$+ \{h[x_1 - \bar{x}_1]^2 + c[u_1 - \bar{u}_1]^2\} \cdot \Delta$$
$$+ \dots$$
$$+ \{h[x_k - \bar{x}_k]^2 + c[u_k - \bar{u}_k]^2\} \cdot \Delta$$
$$+ \{h[x_{\bar{k}-1} - \bar{x}_{\bar{k}-1}]^2 + c[u_{\bar{k}-1} - \bar{u}_{\bar{k}-1}]^2\} \cdot \Delta$$
$$+ \{h[x_{\bar{k}} - \bar{x}_{\bar{k}}]^2\} \cdot \Delta$$

즉 여러 기간에 걸친 생산 판매 비용 사이의 관계를 적분으로 설명하라고 하면 위의 적분 식을 제시하고 이를 이산형으로 옮겨서 쓰면 되는 것이다. 반대로 Discrete 버전을 먼저 쓴 다음에 continuous로 옮겨와도 상관없다.

§ **적분 형태의 단위 설명**

그렇다면 이 중에서 h를 바탕으로 한 번 설명해 보도록 하자.

$$h[x_k - \bar{x}_k]^2 \cdot \Delta$$

위에서 h는 재고 유지비(원/개/달)로 나타낼 수 있다. 따라서 위의 식을 계산하면 (원/개/달)*(개)*(달) = (원/달)이 된다. 이런 식으로 표현하는 것이 가능하다.

또한 최적조건 u(t) = 50으로 주어졌는데, 이 의미에 대해서는 다음과 같이 설명하는 것이 가능하다. 즉

$u(t) = u_k = 50$(개/달)

인데, 이 경우 50개가 최적이므로 50개보다 많은 100개를 생산할 경우에는 야간 작업을 병행해야 할 것이다. 그런데 한계 생산은 체감하므로 50개가 100개가 되면 비용이 오히려 더 든다는 이야기이다. 또는 50개 생산하면 좋은데 20개밖에 생산하지 않는 경우도 있을 수 있다. 그럼 기계가 놀고 작업자들이 놀게 되어 그만큼의 개당 기회비용이 발생하게 되고, 이는 개당 생산비용 증가로 이어지게 된다. 따라서 50개가 최적이라는 의미는 그보다 적거나 많으면 비용이 증가한다는 의미인 것이다.

3. 최적 제어 모형 (page 120)

위의 적분 형태 목적 함수는 전형적인 최적 제어 모형이다.

$$\min J = \int_0^T \{h[x(t) - \bar{x}(t)]^2 + c[u(t) - \bar{u}(t)]^2\} dt$$

$$\dot{x}(t) = u(t) - s(t),\ x(0) = x_0$$

$x(t)$: t시점의 재고(단위 : 개), $\bar{x}(t)$: 목표재고
$u(t)$: t시점의 생산율(단위 : 개/단위기간), $\bar{u}(t)$: 최적조업도
$s(t)$: 판매율(단위 : 개/단위기간), t: 시점
h: 재고 비용계수, c: 생산비용계수

4. 이산형 최적 제어 이론 적용 (page 124-127)

위에서 제시한 생산.재고 계획을 이산형 모형으로 만들고, 여기에 최적화 알고리즘을 적용하여 최적해를 구하는 것이 목표이다.

우선 위의 생산/재고 모형에서 특정 시점에서의 비용을 여러 기간에 걸친 비용의 합으로 다음과 같이 나타낼 수 있다. (이산형이다)

$$\min J = \sum_{k=t}^{k-1} \left\{ h[x_k - \bar{x}_k]^2 + c[u_k - \bar{u}_k]^2 + c_1(k) + c_2 \right\} \Delta$$

위의 여러 기간에 걸친 제약조건과 비용함수를 다음과 같이 동적 최적화 모형으로 정형화 시킬 수 있다.

$$\min J = \sum_{k=0}^{\bar{k}-1} \left\{ h[x_k - \bar{x}_k]^2 + c[u_k - \bar{u}_k]^2 \right\} \Delta$$

$$\frac{x_{k+1} - x_k}{\Delta} = u_k - s_k,\ \ x_0 = x^0$$

이 문제를 다음과 같이 동적 최적화 이론(이산형 최적 제어 이론)으로 다음과 같이 풀 수 있다.

§ 최적화 알고리즘 (125-127page)

위 문제에 대한 말 안장점 문제는 다음과 같다. 라그랑제 식에 대입하는 것은 기본적으로 위의 비선형 계획법에서 배웠던 것과 유사하나, 이산형이기 때문에 λ_{k+1} 이 들어가는 것을 유의하면 된다.

$$\max_{\lambda} \min_{x,u} L(x,u,\lambda)$$
$$= \sum_{k=0}^{\bar{k}-1} \left\{ h[x_k - \bar{x}_k]^2 \cdot \Delta + c[u_k - \bar{u}_k]^2 \cdot \Delta + \lambda_{k+1} [- \frac{x_{k+1} - x_k}{\Delta} + u_k - s_k] \cdot \Delta \right\}$$

이 때 L(x, u, λ)는 라그랑제 함수로서 풀어서 쓰면 아래와 같은 식이다. (중요) (위에서 Sum k=0 부터 bar k-1까지로 되어 있는 것을 다 풀어쓴다고 생각하면 된다) 이 식을 전개할 줄 알아야 하는데, 왜냐하면 이를 미분해서 풀어야 하기 때문이다.

$$L(x,u,\lambda) =$$

(1)k=0 $h[x_0 - \bar{x}_0]^2 \cdot \Delta + c[u_0 - \bar{u}_0]^2 \cdot \Delta + \lambda_1 [- \frac{x_1 - x_0}{\Delta} + u_0 - s_0] \cdot \Delta$

(2)k=1 $h[x_1 - \bar{x}_1]^2 \cdot \Delta + c[u_1 - \bar{u}_1]^2 \cdot \Delta + \lambda_2 [- \frac{x_2 - x_1}{\Delta} + u_1 - s_1] \cdot \Delta$

……

(3)k=k-1 $h[x_{k-1} - \bar{x}_{k-1}]^2 \cdot \Delta + c[u_{k-1} - \bar{u}_{k-1}]^2 \cdot \Delta + \lambda_k [- \frac{x_k - x_{k-1}}{\Delta} + u_{k-1} - s_{k-1}] \cdot \Delta$

(4)k=k $h[x_k - \bar{x}_k]^2 \cdot \Delta + c[u_k - \bar{u}_k]^2 \cdot \Delta + \lambda_{k+1} [- \frac{x_{k+1} - x_k}{\Delta} + u_k - s_k] \cdot \Delta$

……

(5)k=$\bar{k}$-1 $h[x_{\bar{k}-1} - \bar{x}_{\bar{k}-1}]^2 \cdot \Delta + c[u_{\bar{k}-1} - \bar{u}_{\bar{k}-1}]^2 \cdot \Delta + \lambda_{\bar{k}} [- \frac{x_{\bar{k}} - x_{\bar{k}-1}}{\Delta} + u_{\bar{k}-1} - s_{\bar{k}-1}] \cdot \Delta$

이렇게 전개할 줄 알아야 하고, 이 중에서 표시해 둔 것들을 미분해서 0이 만족되는 조건을 뽑아내어야 이산형 최적 제어 모형의 최적 조건을 뽑아낼 수 있다. 이를 계산하면

$$\frac{\partial L}{\partial x_k} = 2h[x_k - \bar{x}_k] \cdot \Delta - \lambda_k + \lambda_{k+1} = 0, \quad k = 1, \ldots, \bar{k} - 1 \qquad (1)$$

$$\frac{\partial L}{\partial x_{\bar{k}}} = -\lambda_{\bar{k}} = 0 \qquad (2)$$

$$\frac{\partial L}{\partial \lambda_k} = -x_k + x_{k-1} + \Delta \cdot u_{k-1} - \Delta s_{k-1} = 0, \quad k = 1, \ldots, \bar{k} \qquad (3)$$

$$\frac{\partial L}{\partial u_k} = 2c[u_k - \bar{u}_k] \cdot \Delta + \Delta \cdot \lambda_{k+1} = 0, \quad k = 0, \ldots, \bar{k} - 1 \qquad (4)$$

를 얻을 수 있다. 이 때 k가 bar k-1까지일 때는 (1)이 쓰이고, k가 bar k 일 때에는 (2)가 쓰이는 거라고 생각하면 된다.

위의 식의 (1), (2), (3), (4)를 각각 정리하면 아래와 같은 식을 얻을 수 있다.

$$\frac{\lambda_{k+1} - \lambda_k}{\Delta} = -2h[x_k - \bar{x}_k], \; k = \bar{k} - 1, \ldots, 1, \; \lambda_{\bar{k}} = 0 \qquad (1), (2)$$

$$\frac{x_{k+1} - x_k}{\Delta} = u_k - s_k, \; k = 0, \ldots, \bar{k} - 1, \; x_0 = x^0 \qquad (3)$$

$$u_k = -\frac{1}{2c}\lambda_{k+1} + \bar{u}_k, \; k = 0, .., \bar{k} - 1 \qquad (4)$$

그리고 이것이 이산형 최적 제어 모형에 대한 폰트리야긴의 최소치 원리이다. 만약 Δ->0이면 위의 최적 조건은 연속형 모형에서의 폰트리야긴의 최소치 원리와 같다.

그런데 위의 최적 조건은 정차 방정식의 2점 경계치 문제로서, 이를 직접 푸는 것 보다는 점별 이완법으로 푸는 것이 더 간단하다. 이는 아래와 같다.

$$u_k^{i+1} = u_k^i - \epsilon \frac{\partial L}{\partial u_k}, \quad k = 0, \ldots, \bar{k} - 1$$

- **실제 프로그래밍으로 구현하기 (중요) (307-308page)**

실제 프로그래밍으로서 이를 구현하기 위해서는 위에서 계산된 수식들을 점별 이완법을 활용할 수 있도록 다시 정리해 주는 과정이 필요하다.

이 때 구해야 하는 것은 u_k, x_{k+1}, $lambda_k$, 그리고 J이다. 각각의 식을 한 번 정리해 보도록 하자.

$$u_k^{i+1} = u_k^i - \epsilon \frac{\partial L}{\partial u_k}, \quad k = 0, \ldots, \bar{k} - 1$$

이 때 편미분 부분을 위에서 구한 (4) 식으로 치환하면 아래를 얻을 수 있다.

$$u_k^{i+1} = u_k^i - \epsilon \{2c[u_k - \bar{u}_k] \cdot \Delta + \lambda_{k+1} \cdot \Delta\}, \quad k = 0, \ldots, \bar{k} - 1$$

$$x_{k+1}^{i+1} = x_k^i + (u_k - s_k) \cdot \Delta, \quad k = 0, \ldots, \bar{k} - 1$$

$$\lambda_k^{i+1} = \lambda_{k+1}^i + 2h[x_k - \bar{x}_k], \; k = \bar{k} - 1, \ldots, 1$$

이를 바탕으로 구현한 코드가 바로 zexhmms.c 이다.

- **코드 : zexhmms.c (원본)**

```
/* ZEXHMMS.c */

#include <stdio.h>
#include <math.h>
#define kk 120
void main()
{
        int iter, k;
        double x[kk+1], u[kk+1], ramda[kk+1], s[kk+1];
        double h, c, x0, xx, uu, tt, dt, epsil, cj;
        FILE *fp;
        fp = fopen("zexhmms_out.txt", "w");

        x0 = 0.;
        xx = 0.;
        uu = 50.;
        h = 1.;
        c = 0.5;
        tt = 12.;
        dt = 0.1;

        for (k=0; k<=kk; k++)
                s[k] = 50. - 10. * sin(2. * 3.141592*(double) k*dt/tt );
```

```
        for(k=0;k<=kk;k++)
        {
                x[k]=x0; u[k]=0.; ramda[k] = 0.;
        }
        epsil = 0.1;

        for(iter = 1; iter <= 500; iter++)
        {
                for( k=0; k<=kk-1; k++)
                        u[k] = u[k] - epsil*(2.*c*(u[k]-uu)+ramda[k+1])*dt;

                for(k=0; k<=kk-1; k++)
                        x[k+1] = x[k] + dt * (u[k] - s[k]);

                for( k=kk-1; k>=1; k--)
                        ramda[k] = ramda[k+1] - dt*(-2.*h*(x[k]-xx));

                cj=0.;
                for(k=0; k<=kk-1; k++)
                        cj = cj + dt * (h* (x[k]-xx) * (x[k]-xx) + c * (u[k]-uu)*(u[k]-uu));
                printf("ITER= %4d   J = %6.4f \n", iter, cj);
                fprintf(fp, "ITER= %4d   J = %6.4f \n", iter, cj);
        }

        printf("\tT\tX\tU\tX\tRAMDA\n");
        fprintf(fp, "\tT\tX\tU\tX\tRAMDA\n");
        for( k=0; k<=kk; k=k+5 )
        {
                printf(" %6.2f %6.2f %6.2f %6.2f %6.2f\n", (double) k*dt, x[k], u[k], s[k],
                ramda[k]);
                fprintf(fp, " %6.2f %6.2f %6.2f %6.2f %6.2f\n", (double) k*dt, x[k], u[k], s[k],
                ramda[k]);
        }

        fclose(fp);
}
```

○ 코드 : zexhmms.c (기말시험용)

```
/* ZEXHMMS.c */

#include <stdio.h>
#include <math.h>
#define kk 6
void main()
{
        int iter, k;
        double x[kk+1], u[kk+1], ramda[kk+1], s[kk+1];
        double h, c, x0, xx, uu, tt, dt, epsil, cj;
        FILE *fp;
        fp = fopen("zexhmms_out.txt", "w");

        x0 = 0.;
        xx = 0.;
        uu = 0.;
        h = 1.;
        c = 1.;
        tt = 6.;
        dt = 1;

        for (k=0; k<=kk; k++)
                s[k] = 50.;

        for(k=0;k<=kk;k++)
        {
                x[k]=x0; u[k]=0.; ramda[k] = 0.;
        }
```



```
        epsil = 0.1;

        for(iter = 1; iter <= 3; iter++)
        {
                for( k=0; k<=kk-1; k++)
                        u[k] = u[k] - epsil*(2.*c*(u[k]-uu)+ramda[k+1])*dt;

                for(k=0; k<=kk-1; k++)
                        x[k+1] = x[k] + dt * (u[k] - s[k]);

                for( k=kk-1; k>=1; k--)
                        ramda[k] = ramda[k+1] - dt*(-2.*h*(x[k]-xx));

                for( k=0; k<=kk; k=k+5 )
                {
                        printf(" %6.2f %6.2f %6.2f %6.2f %6.2f\n", (double) k*dt, x[k], u[k], s[k],
                        ramda[k]);
                        fprintf(fp, " %6.2f %6.2f %6.2f %6.2f %6.2f\n", (double) k*dt, x[k], u[k],
                        s[k], ramda[k]);
                }
        }


        fclose(fp);
}
```