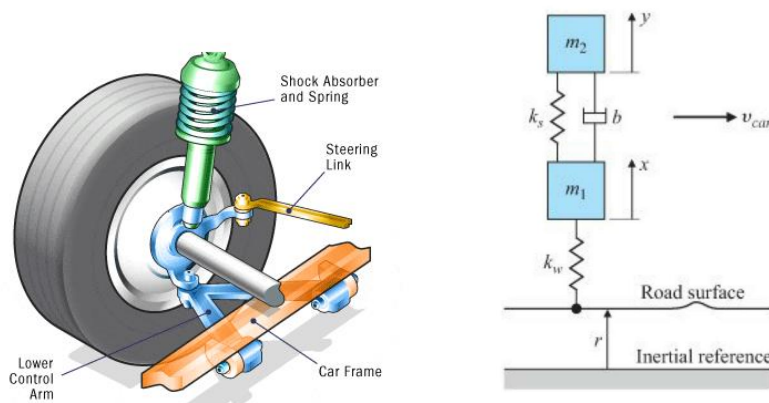


자동차 서스펜션 컨트롤

21011890 노지민

1. 서론

차량이 주행하는 도로면은 항상 일정하지 않으며, 주행 속도 또한 매순간 달라지기 마련이다. 이에 따라 차량에서 진동이 발생하는데, 이 진동은 인체에 많은 피로를 유발한다. 이러한 배경 속에서 진동 감쇄하는 방식으로 차량의 승차감을 증대하기 위해 서스펜션 시스템이 적용된다. 서스펜션 시스템은 차량의 수직 운동에 관여한다. 이때 차량에 전달되는 진동과 같은 충격을 흡수하도록 스프링 같은 역할을 하므로 운전자로 하여금 진동 및 충돌과 같은 요소를 잘 느껴지지 않도록 한다. 비록 차량의 조향과는 직접적인 관련은 없지만 서스펜션 컨트롤은 주행 중의 승차감에 관련된 것으로 매우 강조되고 있으며 계속해 발전하고 있다. 본 연구에서는 서스펜션 시스템의 response 특성을 개선하기 위해 변수를 조정하는 서스펜션 컨트롤 과정을 다룬다.



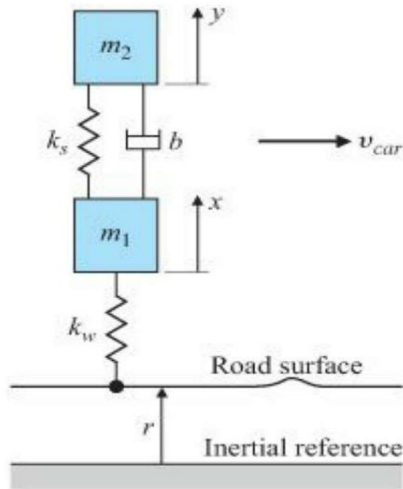
(그림 1 서스펜션 시스템)

서스펜션 시스템

서스펜션 시스템은 그림 1과 같이 4개의 차량 타이어에 각각 연결되어 작용한다. 서스펜션은 spring, shock absorber, suspension arm으로 구성되어 있다. Spring은 탑승자와 차량으로 오는 충격을 감쇄하는 주된 역할을 한다. 하지만, 스프링은 충격을 받으면 계속 진동하는 특성이 있어서 위, 아래로 요동치게 되는데, 이때 shock absorber가 spring이 계속 진동하는 것을 막아주는 역할을 한다. 마지막으로 suspension arm은 차량과 바퀴를 연결해주는 역할을 하며, 바퀴의 움직임을 제어한다. 이러한 구성으로 서스펜션의 시스템은 구성되어 있고, 구성요소의 특성에 따라 서스펜션 종류도 달라진다.

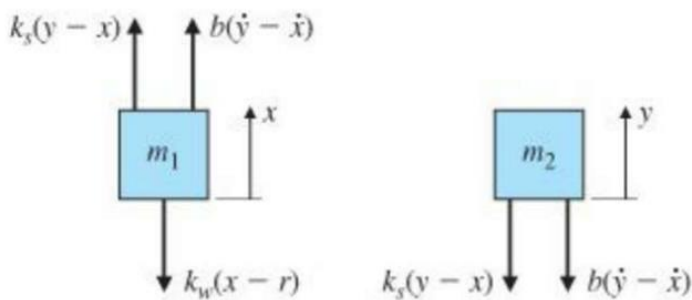
2. 본론

2-1) Modeling



(그림 2 서스펜션 모델링)

서스펜션 컨트롤을 위해 우선 서스펜션 시스템의 수학적 모델링 식을 세워야 한다. 서스펜션 시스템에서 작용하는 물리적 요소를 단순화하여 적용하면 그림 2와 같이 나타낼 수 있다. 이때 서스펜션이 받치는 차체의 질량은 차량의 전체 무게/타이어 개수(4)를 한 m_2 가 되고, 타이어 무게를 m_1 이라고 두며, 그 사이를 있는 K_s 의 탄성 계수를 갖는 spring과 감쇠 상수 b 를 가지는 Damper가 서스펜션 시스템을 구성한다. 또 도로의 시간에 따른 수직 변위를 r 로 설정했다.



(그림 3 작용하는 F)

서스펜션 시스템에서 적용되는 힘 F 를 보면 위와 같이 나타낼 수 있고, 이를 통해 방정식을 세울 수 있다. 우선 m_1 에서 작용하는 힘은 r 에 의한 위아래로 작용하는 힘으로 나타낼 수 있다. 좀 더 자세히 살펴보면 훅의 법칙인 $y = -kx$ (k : 용수철 상수, x : 늘어난 길이) 식을 통해 위로는 $Kx(y-x)$ 만큼 아래로는 $Kw(x-r)$ 만큼 힘이 가해짐을 알 수 있다. 그리고 가장 중요한 마찰력은 $b(y'-x')$ 만큼 적용된다. 이를 다 더하면 최종적으로 알짜 힘 $m_1 \cdot x''$ 과 같아진다. 식을 정리하면 아래와 같이 나타낼 수 있다.

$$1) \quad m_1 x'' = b(y' - x') + K_s(y - x) - K_w(x - r)$$

m_2 도 m_1 과 마찬가지로 식을 세울 수 있다. m_2 에서는 위로 작용하는 힘이 없으므로 아래로 작용하는 스프링에 의한 힘과 마찰력을 더 해서 $m_2 y''$ 과 같다는 것을 통해 식을 세울 수 있다. 스프링에 의한 힘은 $K_s(y-x)$ 마찰력은 $b(y'-x')$ 만큼 적용된다. 식은 아래와 같다.

$$2) \quad m_2 y'' = -K_s(y-x) - b(y'-x')$$

이렇게 서스펜션 시스템을 방정식을 모델링 할 수 있다. 이제 방정식을 미정계수법을 eigenvalue를 통해 General solution을 도출하는 방법과, Laplace transform을 통한 방법 총 두가지 방법으로 solution을 구해볼 것이다.

2-1) General solution with The Method of Undetermined Coefficient with Eigenvalue

먼저 위의 방정식에서 사용할 상태 변수를 설정하여 각 상태 변수의 미분 값에 대해 표현해 행렬식 형태의 상태방정식으로 나타낸다. 그 후 eigenvalue를 통해 특성방정식을 유도한다.

상태변수는 아래와 같이 $z_1 \sim z_4$ 로 정의했다.

$$z_1 = y - x \quad \rightarrow \text{서스펜션의 변위}$$

$$z_2 = y' \quad \rightarrow m_2 \text{의 속도}$$

$$z_3 = x - r \quad \rightarrow \text{타이어 스프링에서의 변위}$$

$$z_4 = x' \quad \rightarrow m_1 \text{의 속도}$$

각 상태변수의 정의를 바탕으로 F에 대한 4개의 방정식을 아래와 같이 도출할 수 있다.

$$z_1' = z_2 - z_4$$

$$m_2 z_2' = -K_s z_1 - b z_1'$$

$$z_3' = z_4 - r'$$

$$m_1 z_4' = K_s z_1 + b z_1' - K_u z_3$$

이를 각 상태변수의 미분 값에 대해 표현해 상태 방정식을 표현할 수 있다. 이를 비제차 선형 상미분방정식인 $y' = Ay + g$ 의 형태로 만들면 아래와 같이 나타낼 수 있다.

$$\begin{bmatrix} z_1' \\ z_2' \\ z_3' \\ z_4' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -\frac{K_s}{m_2} & -\frac{b}{m_2} & 0 & \frac{b}{m_2} \\ 0 & 0 & 0 & 1 \\ \frac{K_s}{m_1} & \frac{b}{m_1} & -\frac{K_u}{m_1} & -\frac{b}{m_1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} r'$$

(그림 3 상태 방정식)

이때 general solution은 $y = y(h) + y(p)$ 이다. $Y(p)$ 는 입력에 대한 solution이므로 $y(h)$ 에 대한 eigenvalue problem을 풀어보자. $Ax = \lambda x$ 로 나타내고 $(A - \lambda I)x = 0$ 를 이용해 det값을 구해 특성방정식으로부터 구할 수 있다.

$$\det \left(\begin{bmatrix} 0 & 1 & 0 & -1 \\ -\frac{k_s}{m_2} & -\frac{b}{m_2} & 0 & \frac{b}{m_2} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{m_1} & \frac{b}{m_1} & -\frac{k_w}{m_1} & -\frac{b}{m_1} \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

(그림 4 det 식)

이 과정은 MATLAB을 사용했다. λ 대신 x 를 사용했다.

```

1 - syms x;
2 - syms b;
3 - syms ks;
4 - syms kw;
5 - syms ma;
6 - syms mb;
7
8 - A = [-x 1 0 -1;
9        -ks/mb -b/mb-x 0 b/mb;
10       0 0 -x 1;
11       ks/ma b/ma -kw/ma -b/ma-x];
12
13 - d = det(A);

```

$$d = (ks*kw + b*ma*x^3 + b*mb*x^3 + ks*ma*x^2 + ks*mb*x^2 + kw*mb*x^2 + ma*mb*x^4 + b*kw*x)/(ma*mb)$$

식을 다시 x 를 λ 로 정리하면

$$\lambda^4 + \left(\frac{b}{m_1} + \frac{b}{m_2} \right) \lambda^3 + \left(\frac{k_s}{m_1} + \frac{k_s}{m_2} + \frac{k_w}{m_1} \right) \lambda^2 + \left(\frac{k_w b}{m_1 m_2} \right) \lambda + \frac{k_w k_s}{m_1 m_2}$$

(특성 방정식)

이런 4차 특성방정식을 구할 수 있다.

이때 자동차의 총 질량이 1580kg이고 바퀴 1개의 무게가 20kg일 때 서스펜션 모델에 적용해 보자. 각 변수는 $ma (= m_1) = 20\text{kg}$, $mb (= m_2) = 375\text{kg}$, $K_s = 130000 \text{ N/m}$, $K_w = 1000000 \text{ N/m}$, $b = 5000 \text{ Nsec/m}$ 로 세팅한다. (단, 초기값은 모두 0이고 중력은 없다고 가정한다.)

```

1 - m2 = 375;
2 - m1 = 20;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;
6
7 - A = [0 1 0 -1;
8       -ks/m2 -b/m2 0 b/m2;
9       0 0 0 1;
10      ks/m1 b/m1 -kw/m1 -b/m1];
11
12 - [Evector, Evalue] = eig(A);
13

```

(MATLAB 코드 eig 사용)

eigenvalue λ 는 아래와 같다.

Evalue =

1.0e+02 *

```

-1.2622 + 1.9454i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
0.0000 + 0.0000i   -1.2622 - 1.9454i    0.0000 + 0.0000i    0.0000 + 0.0000i
0.0000 + 0.0000i    0.0000 + 0.0000i   -0.0544 + 0.1711i    0.0000 + 0.0000i
0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i   -0.0544 - 0.1711i

```

그리고 λ 에 eigenvalue 값을 넣어 eigenvector를 구한다. 위에서부터 $\lambda_1 \sim 4$ 이렇게 설정했다.

MATLAB을 통해 eigenvector를 구하면, 아래와 같다.

Evector =

$V_1 =$ $V_2 =$ $V_3 =$ $V_4 =$

$$\begin{bmatrix} -0.0023 - 0.0038i \\ 0.0331 + 0.0452i \\ 0.0023 + 0.0036i \\ -0.9984 + 0.0000i \end{bmatrix} \begin{bmatrix} -0.0023 + 0.0038i \\ 0.0331 - 0.0452i \\ 0.0023 - 0.0036i \\ -0.9984 + 0.0000i \end{bmatrix} \begin{bmatrix} 0.0188 + 0.0462i \\ -0.9914 + 0.0000i \\ -0.0021 + 0.0064i \\ -0.0981 - 0.0700i \end{bmatrix} \begin{bmatrix} 0.0188 - 0.0462i \\ -0.9914 + 0.0000i \\ -0.0021 - 0.0064i \\ -0.0981 + 0.0700i \end{bmatrix}$$

이를 통해 general solution의 $y(h)$ 부분을 구할 수 있다. v 는 위에서부터 1~4이렇게 됐다.

$$y(h) = c_1 v_1 e^{\lambda_1 t} + c_2 v_2 e^{\lambda_2 t} + c_3 v_3 e^{\lambda_3 t} + c_4 v_4 e^{\lambda_4 t}$$

또, 이 식을 오일러 공식을 이용해 아래와 같이 표현할 수도 있다.

$$e^{ix} = \cos x + i \sin x \quad (\text{오일러 공식})$$

$$\begin{aligned}
y(h) = & C_1 V_1 e^{-126.22} (\cos(194.54) + i \sin(194.54)) \\
& + C_2 V_2 e^{-126.22} (\cos(-194.54) + i \sin(-194.54)) \\
& + C_3 V_3 e^{-5.44} (\cos(17.11) + i \sin(17.11)) \\
& + C_4 V_4 e^{-5.44} (\cos(-17.11) + i \sin(-17.11))
\end{aligned}$$

(y(h) 결과 값)

이제 미정계수법을 바탕으로 y(p)를 구한 뒤 y(h)와 더해 general solution을 구한다.

미정계수법은 basic rule, modification rule, sum rule을 바탕으로 y(p)를 찾게 된다. 간단히 과정을 설명하면 y(p)의 형태를 y(h)와 비교해 겹치지 않는 solution 형태를 갖도록 $C \cdot x^n \cdot e^{0.01t}$ 형태로 나타낸다. 여기서 x의 제곱을 나타내는 n은 $e^{0.01t}$ 와 겹치는 solution이 y(h)에 없으므로 0으로 설정할 수 있다. 이렇게 기본 y(p)의 형태를 $C \cdot e^{0.01t}$ 로 두고 y', y'', y''', y'''' 를 구하고 원래의 4차 방정식에 각 값을 대입해 상수 C를 구했다. 이 과정은 MATLAB을 사용해 특정 지었다. 과정은 아래와 같다.

$y_p = C_5 e^{0.01t}$ 를 가정한다.

$$\begin{cases}
y_p' = C_5 \cdot 10^1 \cdot e^{0.01t} \\
y_p'' = C_5 \cdot 10^4 \cdot e^{0.01t} \\
y_p''' = -C_5 \cdot 10^6 \cdot e^{-0.01t} \\
y_p'''' = C_5 \cdot 10^8 \cdot e^{-0.01t}
\end{cases}$$
이 값을 주어진 방정식에

$$r(x) = y'''' + \left(\frac{b}{m_1} + \frac{b}{m_2}\right) y''' + \left(\frac{k_s}{m_1} + \frac{k_s}{m_2} + \frac{k_w}{m_1}\right) y'' + \left(\frac{k_w}{m_1 m_2}\right) y' + \frac{k_w k_s}{m_1 m_2} y$$
의 좌변 $y^{(n)}$ 에 맞게 대입하면, C_5 를 구해주면
 $\therefore y_p = \underline{C_5 e^{-0.01t}}$ 가 된다.
그러므로 최종 $y(t) = y(h) + C_5 e^{-0.01t}$ 이다.

- 1 - $m_1 = 20;$
- 2 - $m_2 = 375;$
- 3 - $b = 5000;$
- 4 - $k_s = 130000;$
- 5 - $k_w = 1000000;$

```
input = [100000000 + (-1000000*(b/m2)+(b/m1)) + 10000*((kw/m1)+(ks/m2)+(ks/m1))-100*(b+kw)/(m1+m2)+(ks+kw)/(m1+m2) -1];
```

`yp = roots(input);` (입력에 대한 C구하는 MATLAB 코드)

`yp =`

`1.6507e-09` (C 결과 값)

위 결과 값을 바탕으로 최종적인 general solution을 아래와 같이 구할 수 있다.

MATLAB에서 각 값을 대입해 C 를 풀면, (roots 함수 사용)

$C = 1.6507 \cdot 10^{-9}$

$$y(t) = y_h + 1.6507 \cdot 10^{-9} e^{-0.01t}$$

general solution.

2-2) Laplace transform

위의 모델링 과정에서 도출할 수 있었던 두 식에 대해 라플라스 변환을 한 뒤, $X(s)$ 를 $Y(s)$ 에 관한 식으로 소거한 뒤 라플라스 역변환을 취하면 general solution을 얻을 수 있다. 전개 과정은 아래와 같다.

위와 동일한 과정을 통해

$$1) m_1 x'' = b(y - x') + k_s(y - x) - k_w(x - r)$$

$$2) m_2 y'' = -k_s(y - x) - b(y' - x')$$

식을 세울 수 있다.

이식을 변형해 라플라스 변환하여 정리한다.

$$*1) x'' + \frac{b}{m_1}(x' - y') + \frac{k_s}{m_1}(x - y) + \frac{k_w}{m_1}x = \frac{k_w}{m_1}r$$

$$*2) y'' + \frac{b}{m_2}(y' - x') + \frac{k_s}{m_2}(y - x) = 0$$

이제 위식을 라플라스 변환해준다.

$$\mathcal{L}(*1) \Rightarrow s^2 X(s) + s \frac{b}{m_1}(X(s) - Y(s)) + \frac{k_s}{m_1}(X(s) - Y(s)) + \frac{k_w}{m_1}X(s) = \frac{k_w}{m_1}R(s)$$

$$\mathcal{L}(*2) \Rightarrow s^2 Y(s) + s \frac{b}{m_2}(Y(s) - X(s)) + \frac{k_s}{m_2}(Y(s) - X(s)) = 0$$

이 식을 전개해 $Y(s)$ 에 대한 식으로 나타내보자.

$$s^2 Y(s) + s \frac{b}{m_2} Y(s) - s \frac{b}{m_2} X(s) + \frac{k_s}{m_2} Y(s) - \frac{k_s}{m_2} X(s) = 0$$

$$(s^2 + \frac{b}{m_2}s + \frac{k_s}{m_2})Y(s) = (s \frac{b}{m_2} + \frac{k_s}{m_2})X(s)$$

$$\therefore X(s) = \frac{(s^2 + \frac{b}{m_2}s + \frac{k_s}{m_2})}{(s \frac{b}{m_2} + \frac{k_s}{m_2})} Y(s)$$

이식을 $\mathcal{L}(*1)$ 식에 대입해
 $Y(s)$ 와 $R(s)$ 로 정리하자.

$$\frac{k_w}{m_1} \cdot s^2 X(s) + s \frac{b}{m_1} (X(s) - Y(s)) + \frac{k_s}{m_1} (Y(s) - X(s)) + \frac{k_w}{m_1} X(s)$$

$$X(s) = \frac{(s^2 + \frac{b}{m_2} s + \frac{k_s}{m_2}) Y(s)}{(s \frac{b}{m_2} + \frac{k_s}{m_2})} \quad \uparrow \text{대입}$$

$$\Rightarrow s^2 \frac{(s^2 + \frac{b}{m_2} s + \frac{k_s}{m_2})}{(s \frac{b}{m_2} + \frac{k_s}{m_2})} Y(s) + s \frac{b}{m_1} \cdot s^2 Y(s) + \frac{k_s}{m_1} \cdot s^2 X(s) + \frac{k_w}{m_1} \cdot \frac{(s^2 + \frac{b}{m_2} s + \frac{k_s}{m_2})}{(s \frac{b}{m_2} + \frac{k_s}{m_2})} Y(s)$$

$$= \frac{k_w}{m_1} R(s)$$

이 식을 전라하면

$$\frac{Y(s)}{R(s)} = \frac{\frac{k_w b}{m_1 m_2} (s + \frac{k_s}{b})}{s^4 + (\frac{b}{m_1} + \frac{b}{m_2}) s^3 + (\frac{k_s}{m_1} + \frac{k_s}{m_2} + \frac{k_w}{m_1}) s^2 + (\frac{k_w b}{m_1 m_2}) s + \frac{k_w k_s}{m_1 m_2}}$$

이때 $R(s)$ 은 impulse로 $\delta(t)$ 라고 쓴 $\delta(t)$ 의 라플라스 변환 $\mathcal{L}(\delta(t)) = 1$ 이므로
 이를 대입해준다.

\therefore 전달함수는

$$\frac{\frac{k_w b}{m_1 m_2} s + \frac{k_w k_s}{m_1 m_2}}{s^4 + (\frac{b}{m_1} + \frac{b}{m_2}) s^3 + (\frac{k_s}{m_1} + \frac{k_s}{m_2} + \frac{k_w}{m_1}) s^2 + (\frac{k_w b}{m_1 m_2}) s + \frac{k_w k_s}{m_1 m_2}} \quad \text{이것}$$

이렇게 라플라스 변환을 해 주었다. 그리고 위와 같이 전달함수를 도출했다. 이제 전달함수의 분모에 대한 인수분해를 MATLAB의 root를 통해 하고, 라플라스 역변환이 쉽게 하기 위해 추가적으로 부분 분수를 MATLAB의 residue를 통해 전개한다.

전달함수의 분모의 식을 아래와 같이 코드를 작성하여 den이라는 변수에 저장하고, MATLAB에 roots 함수를 사용해 인수분해를 구했다.

```
1 - m1 = 20;
2 - m2 = 375;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;

den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b*kw)/(m1*m2) (ks*kw)/(m1*m2)];

r = roots(den);
```

```
r =

1.0e+02 *

-1.2622 + 1.9454i
-1.2622 - 1.9454i
-0.0544 + 0.1711i
-0.0544 - 0.1711i
```

(MATLAB 코드와 도출된 인수분해해)

$$\frac{\frac{k_w b}{m_1 m_2} \left(s + \frac{k_s}{b}\right)}{(s - (-126.22 + 194.54i))(s - (-126.22 - 194.54i))(s - (-5.44 + 17.11i))(s - (-5.44 - 17.11i))}$$

이렇게 MATLAB을 통해 분모를 주어진 r값을 이용해 인수 분해할 수 있다.

그리고 이 식을 부분분수를 통해 형태를 바꿔줄 수 있다.

부분 분수의 정확한 값을 유도하기 위해 MATLAB의 residue 함수를 사용했다.

```
num = [0 0 (kw+b)/(m1*m2) (ks*kw)/(m1*m2)];
den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b*kw)/(m1*m2) (ks*kw)/(m1*m2)];

[r,p,k] = residue(num,den);
```

```
r =
a -5.7480 + 4.2845i
b -5.7480 - 4.2845i
c 5.7480 - 8.1376i
d 5.7480 + 8.1376i

p =
1.0e+02 *
1 -1.2622 + 1.9454i
2 -1.2622 - 1.9454i
3 -0.0544 + 0.1711i
4 -0.0544 - 0.1711i
```

(residue의 코드 및 결과값)

이 값을 토대로 식을 전개해 보면 아래와 같다.

$$\frac{k_w b}{m_1 m_2} \left(s + \frac{k_s}{b} \right)$$

$$\frac{(s - (-126.22 + 194.54i)) (s - (-126.22 - 194.54i)) (s - (-5.44 + 17.11i)) (s - (-5.44 - 17.11i))}{\begin{matrix} \hookrightarrow \lambda_1 & \hookrightarrow \lambda_2 & \hookrightarrow \lambda_3 & \hookrightarrow \lambda_4 \end{matrix}}$$

특성 $\lambda_1 \sim \lambda_4$ 3 값을 아래에 나타내겠다.

부분분수를 사용해,

$$= \frac{a}{(s - \lambda_1)} + \frac{b}{(s - \lambda_2)} + \frac{c}{(s - \lambda_3)} + \frac{d}{(s - \lambda_4)}$$

라플라스 역변환

$$\Rightarrow y(t) = a e^{\lambda_1 t} + b e^{\lambda_2 t} + c e^{\lambda_3 t} + d e^{\lambda_4 t}$$

를 구할 수 있다. 이 식 또한 2개의 공식을 통해
 $e^{ix} = (\cos x + i \sin x)$ 형태로 표현할 수 있다.

a, b, c, d의 값은 Matlab에서 `residue` 함수로 구하면,
 값을 대입하면 된다.

위처럼 라플라스 변환을 사용해 solution 을 구하면 초기값을 따로 구할 필요 없이 바로 general solution 이 도출됨을 알 수 있다.

2-3) MATLAB을 이용한 Response plot (입력은 step을 가정)

```

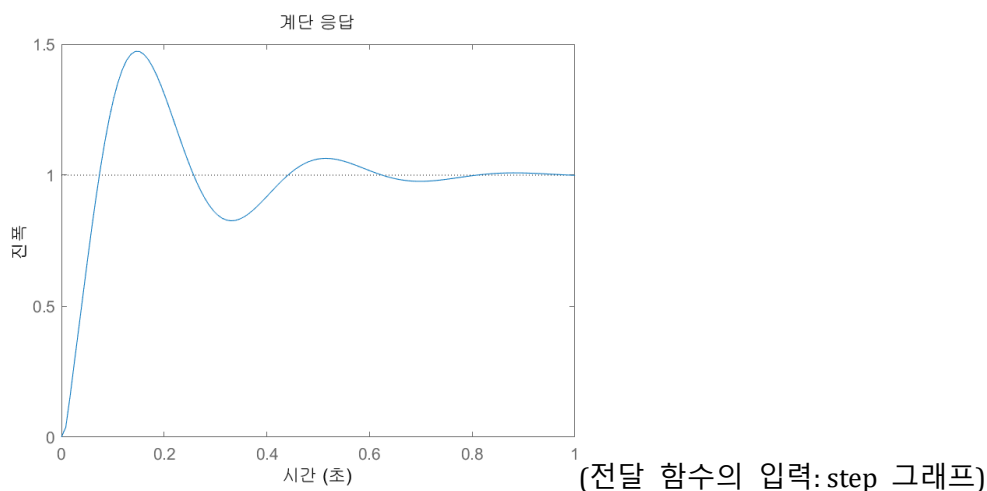
1 - m1 = 20;
2 - m2 = 375;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;

14
15 - num = [0 0 (kw+b)/(m1*m2) (ks+kw)/(m1*m2)];
16 - den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b+kw)/(m1*m2) (ks+kw)/(m1*m2)];
17
18 - figure(1),step(num,den);
19

```

(MATLAB 코드)

위와 같이 전달 함수의 분모와 분자를 나누어 num과 den의 변수에 각각 계수를 나타낸다. 그리고 figure함수와 step 함수를 통해 입력이 step일 때 서스펜션 시스템의 response가 아래와 같이 감쇠 진동의 형태로 나타남을 알 수 있다.



```

num = [0 0 (kw+b)/(m1*m2) (ks+kw)/(m1*m2)];
den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b+kw)/(m1*m2) (ks+kw)/(m1*m2)];

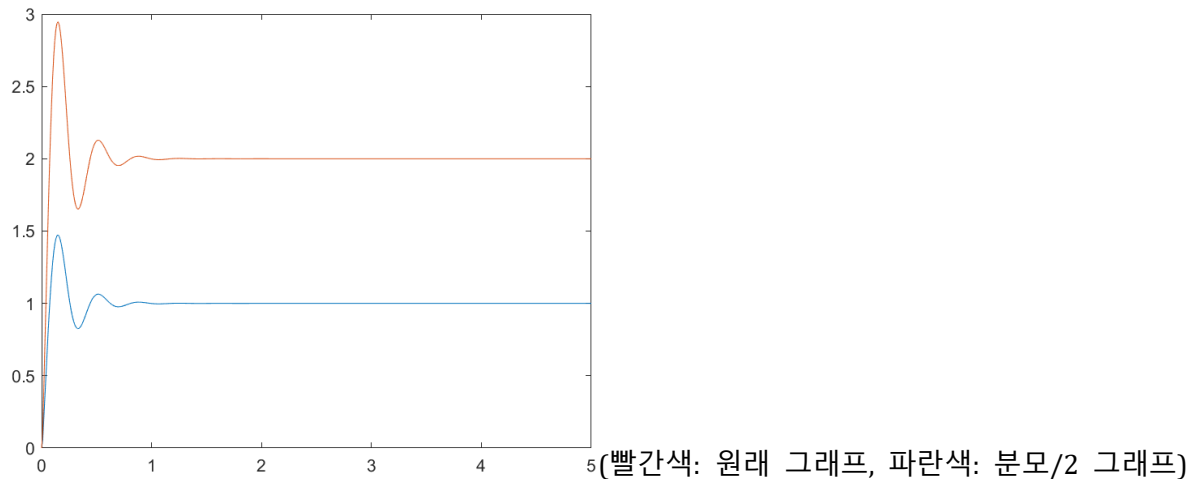
figure(1),step(num,den);

t=0:0.01:5;
out1 = step(num,den,t);
out2 = step(num,den/2,t);
figure(2),plot(t,out1,t,out2)

```

(분모를 2로 나누었을 때의 원래의 그래프와 비교 그래프 코드)

이번엔 분모의 계수를 2로 나눈 뒤 시스템의 response가 어떻게 달라지는 지 확인하기 위해 추가적으로 그래프를 MATLAB으로 그려보았다. 그 결과 아래와 같이 전체적인 response 값이 원래의 그래프와 비교했을 때 딱 절반이 줄어든 것을 살펴볼 수 있다.



3. 본문 2의 1, 2번 풀이 방법 비교

3-1) 미분방정식의 해법과 라플라스 변환의 관계

라플라스 변환은 미분방정식을 대수방정식 꼴로 변환하는 과정을 통해 보다 쉬운 방정식으로 미분방정식의 해를 구할 수 있도록 한 방법이다. 미분방정식은 미분개념과 적분개념이 모두 포함되어 있는 방정식인데, 이러한 변화율은 지수함수나 삼각함수와 같은 초월함수가 포함될 경우 이해하기 어려우며 쉽게 해를 구할 수 없다는 단점이 있다. 반면에 대수방정식 꼴은 인수분해 또는 근의 공식을 통해 쉽게 해를 구할 수 있다는 장점이 있다. 또 대수방정식의 해를 구하는 과정에서 자연스럽게 초기값이 사용되므로 해의 형태가 일반해가 아닌 특수해 형태로 나온다는 장점이 있다. 즉 미지상수가 없다. 결과적으로 라플라스 변환의 방법은 미분방정식을 대수방정식으로 바꾸고, 대수방정식의 해를 구한 뒤 다시 라플라스 역변환을 통해 원래 미분방정식의 해를 얻는 방식으로 전개된다. 기본적으로 라플라스 변환에서는 기본 변환표와 함께 미분공식, 적분공식을 사용한다. 정의는 아래와 같다.

$$\mathcal{L}(f) = F(s) = \int_0^{\infty} e^{-st} f(t) dt$$

라플라스 변환의 정의



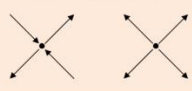



이 정의를 이용해 시간에 대한 함수 f 를 s 라는 새로운 변수에 대한 함수 F 로 변환시킬 수 있다.

3-2) 일반적인 방법과 라플라스 변환을 이용하는 방법에 대한 비교

미분방정식을 구하는 일반적인 방법과 라플라스 변환은 모두 시스템의 일반해를 구하기 위한 방법이지만, 해를 구하는 방법에 있어서 많은 차이를 보인다.

우선 비제차 상미분방정식을 푸는데 있어서 차이를 보인다. 일반적인 방법에서는 제차 상미분방정식에 대한 해를 구하고 입력에 대한 해를 구해 더하는 방식으로 해를 구한다. 또 해를 이루는 미지상수를 구하기 위해 초기값 문제를 풀어야한다. 반면에 라플라스 변환을 사용하면 제차 상미분방정식을 먼저 푸는 것이 요구되지 않는다. 또 초기값은 자동적으로 처리되므로 따로 초기값 문제를 풀 필요가 없다. 이것 외에도 라플라스 변환은 변화과 그 역변환의 관계를 명확히 밝혀주는 기본적 특성과 그 결과로 얻어지는 기법들을 갖고 있다. 또 단위계단함수 (step function)와 Dirac의 델타 함수의 입력에 대한 솔루션을 간단하고 광범위하게 적용할 수 있는 장점이 있다.

3-3) Relationship between eigenvalue and systems

	Stable $\text{Re}(\lambda_d) < 0$	Lyapunov stable* $\text{Re}(\lambda_d) = 0$	Unstable $\text{Re}(\lambda_d) > 0$
Real eigenvalues	Stable point $\text{Re}(\lambda_1) < 0, \text{Re}(\lambda_2) < 0$ 	Neutral point $\text{Re}(\lambda_1) = 0, \text{Re}(\lambda_2) < 0$ $\text{Re}(\lambda_1) = 0, \text{Re}(\lambda_2) = 0$ 	Saddle point Unstable point $\text{Re}(\lambda_1) > 0, \text{Re}(\lambda_2) < 0$ $\text{Re}(\lambda_1) > 0, \text{Re}(\lambda_2) > 0$ 
Complex conjugate eigenvalues	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) < 0$ Stable spiral focus 	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) = 0$ Neutral center 	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) > 0$ Unstable spiral focus 

Eigenvalue 는 기본적인 spring-damper-system 뿐만 아니라 대부분의 역학 시스템의 출력과 매우 밀접한 관계를 갖는다. Eigenvalue 의 실수부가 0 보다 작으면 시스템에서의 진폭은 감소하여 특정 값으로 수렴한다. 수렴하면서 시스템은 stable 하게 되는데 이렇듯 eigenvalue 의 허수부는 시스템에서의 stability, 즉 주파수 등의 속성 값을 결정하는 매우 중요한 요소이며, 이를 이용해 system 에서 우리가 원하고자 하는 출력 값을 유도하는 것에 있어 매우 유용하다.

4. response 특성을 개선하기 위한 솔루션

서스펜션 시스템에서의 response 는 충격으로 인한 진동을 최대한 빠르게 상쇄하여 차가 안정적으로 달릴 수 있도록 유도해야 한다. 여러 변수 중 조작성 용이한 서스펜션 댐핑과 스프링 상수를 조작하여 서스펜션 시스템 목적에 최적화되도록 response 를 조작해볼 것이다.

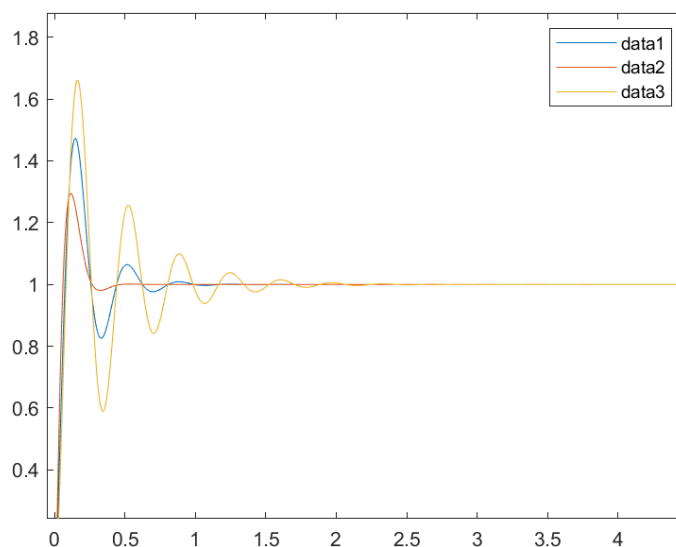
- 서스펜션 댐핑 (b) 2 배 증가 & 1/2 감소

```

1 - m1 = 20;
2 - m2 = 375;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;
6
7 - num = [0 0 (kw*b)/(m1*m2) (ks*kw)/(m1*m2)];
8 - den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b*kw)/(m1*m2) (ks*kw)/(m1*m2)];
9 - b1 = 10000;
10 - num_bplus = [0 0 (kw*b1)/(m1*m2) (ks*kw)/(m1*m2)];
11 - den_bplus = [1 (b1/m2)+(b1/m1) (kw/m1)+(ks/m2)+(ks/m1) (b1*kw)/(m1*m2) (ks*kw)/(m1*m2)];
12 - b2 = 2500;
13 - num_bminus = [0 0 (kw*b2)/(m1*m2) (ks*kw)/(m1*m2)];
14 - den_bminus = [1 (b2/m2)+(b2/m1) (kw/m1)+(ks/m2)+(ks/m1) (b2*kw)/(m1*m2) (ks*kw)/(m1*m2)];
15
16 - out1 = step(num,den,t);
17 - out2 = step(num_bplus,den_bplus,t);
18 - out3 = step(num_bminus,den_bminus,t);
19
20 - figure(1),plot(t,out1,t,out2,t,out3);

```

(MATLAB 코드)



(도출된 그래프, data 1: 원래 data2: b 2 배 증가 data3: b 1/2 감소)

MATLAB 을 통해 직접 b 의 값의 두배와 1/2 배의 그래프를 그리고 원래 시스템과 비교해 보았다. 원래 그래프와 비교했을 때 b 의 값이 2 배 증가했을 때 시스템이 더욱 빨리 안정되는 것을 볼 수 있다. 그러므로 b 의 댐핑 값은 증가하는 방향으로 서스펜션을 조작해야 더욱 최적화할 수 있다.

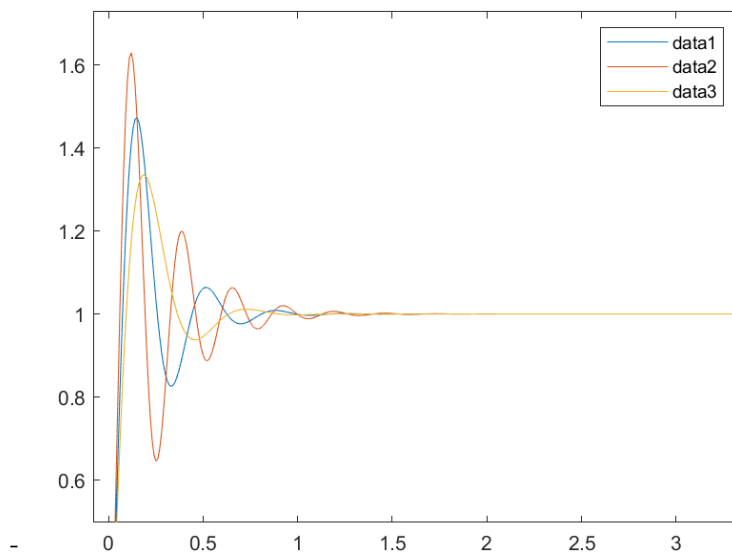
- 서스펜션 스프링 상수(K_s) 2 배 증가 & 1/2 감소

```

1 - m1 = 20;
2 - m2 = 375;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;
6
7 - num = [0 0 (kw+b)/(m1+m2) (ks*kw)/(m1+m2)];
8 - den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b*kw)/(m1+m2) (ks*kw)/(m1+m2)];
9
10 - ks1= 260000;
11 - num_kplus = [0 0 (kw+b)/(m1+m2) (ks1*kw)/(m1+m2)];
12 - den_kplus = [1 (b/m2)+(b/m1) (kw/m1)+(ks1/m2)+(ks1/m1) (b*kw)/(m1+m2) (ks1*kw)/(m1+m2)];
13
14 - ks2= 65000;
15 - num_kminus = [0 0 (kw+b)/(m1+m2) (ks2*kw)/(m1+m2)];
16 - den_kminus = [1 (b/m2)+(b/m1) (kw/m1)+(ks2/m2)+(ks2/m1) (b*kw)/(m1+m2) (ks2*kw)/(m1+m2)];
17
18 - out1 = step(num,den,t);
19 - out2 = step(num_kplus,den_kplus,t);
20 - out3 = step(num_kminus,den_kminus,t);
21
22 - figure(1),plot(t,out1,t,out2,t,out3);

```

(MATLAB 코드)



- (도출된 그래프, data 1: 원래 data2: ks 2 배 증가 data3: Ks 1/2 감소)

이번에는 서스펜션 스프링 상수 K_s 를 2 배 1/2 배씩 조작했다. 원래 그래프와 비교했을 때 스프링 댐퍼 상수와는 달리 ks 가 1/2 만큼 감소했을 때 시스템이 더욱 빨리 안정되는 것을 볼 수 있다. 즉, 서스펜션 시스템의 최적화를 위해 스프링 상수 K_s 는 감소되는 방향이 되어야한다.

- 서스펜션 댐핑(b) 2 배 증가 & 스프링 상수(K_s) 1/2 감소

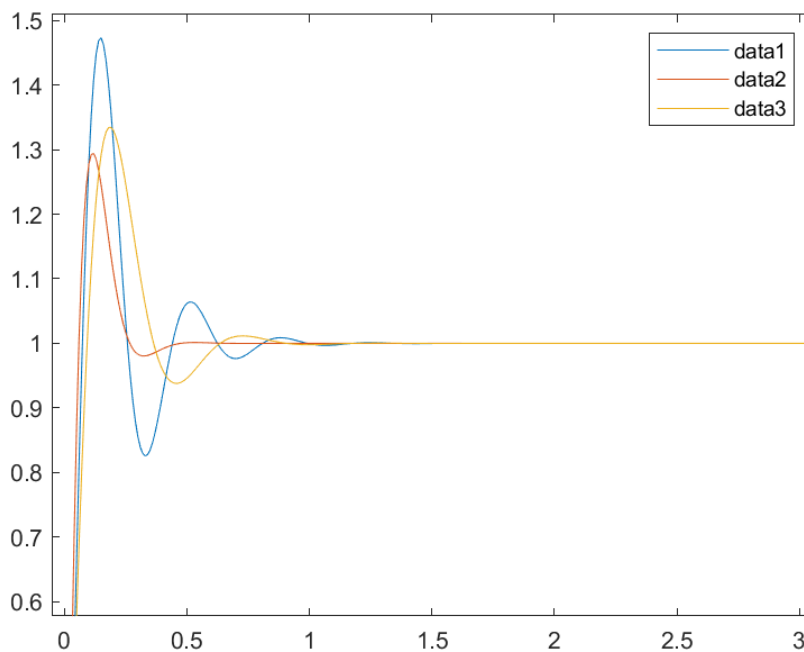
이제 위의 서스펜션 시스템의 최적화를 위해 각 변수가 증가, 감소하는 방향으로 조작되어야 한다는 사실을 알았다. 이제 두 상수 중 어떤 상수의 조작이 효율적인지 MATLAB 을 통해 알아보겠다.


```

1 - m1 = 20;
2 - m2 = 375;
3 - b = 5000;
4 - ks = 130000;
5 - kw = 1000000;
6
7 - num = [0 0 (kw+b)/(m1+m2) (ks+kw)/(m1+m2)];
8 - den = [1 (b/m2)+(b/m1) (kw/m1)+(ks/m2)+(ks/m1) (b+kw)/(m1+m2) (ks+kw)/(m1+m2)];
9 - b1 = 10000;
10 - num_bplus = [0 0 (kw+b1)/(m1+m2) (ks+kw)/(m1+m2)];
11 - den_bplus = [1 (b1/m2)+(b1/m1) (kw/m1)+(ks/m2)+(ks/m1) (b1+kw)/(m1+m2) (ks+kw)/(m1+m2)];
12 - ks1 = 65000;
13 - num_kminus = [0 0 (kw+b)/(m1+m2) (ks1+kw)/(m1+m2)];
14 - den_kminus = [1 (b/m2)+(b/m1) (kw/m1)+(ks1/m2)+(ks1/m1) (b+kw)/(m1+m2) (ks1+kw)/(m1+m2)];
15
16 - out1 = step(num,den,t);
17 - out2 = step(num_bplus,den_bplus,t);
18 - out3 = step(num_kminus,den_kminus,t);
19
20 - figure(1),plot(t,out1,t,out2,t,out3);

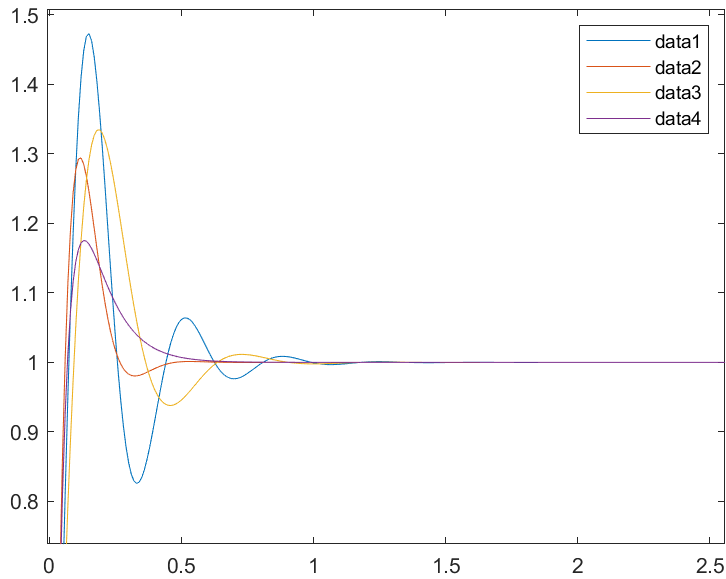
```

(MATLAB 코드)



(도출된 그래프, data 1: 원래 data2: b 2 배 증가 data3: Ks 1/2 감소)

결과적으로 그래프만 보면, 스프링 댐퍼 상수 b 의 2 배 증가가 스프링 상수 K_s 의 1/2 감소보다 더 빨리 시스템의 안정화를 이끌어낸다는 것을 알아 낼 수 있다. 원래 그래프와 비교했을 때 b 와 K_s 조작 모두 서스펜션 시스템의 안정화를 이끌어 냈다. 한 가지 변수만 조작한다면 댐퍼 상수의 조작이 효율적이지만, 두 가지 값을 모두 조작한다면 서스펜션 시스템이 더욱 안정화되는 것을 아래의 그래프를 통해 알 수 있다.

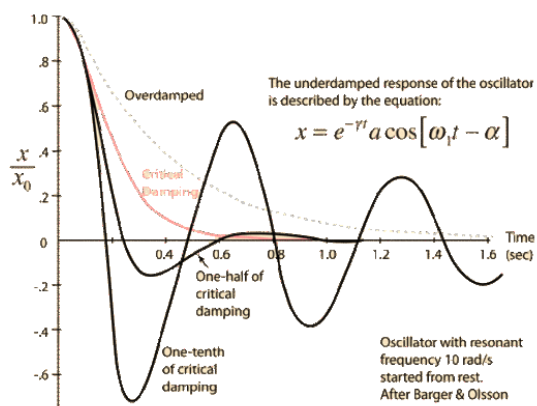


(도출된 그래프, data 1: 원래 data2: b 2 배 증가 data3: Ks 1/2 감소 data4: b 2 배 증가, Ks 1/2 감소)

결론

실제 서스펜션 시스템의 최적화

서스펜션 시스템의 최적화를 위해서는 적절하게 댐핑 상수를 증가하고 스프링 상수를 감소해야 한다. 하지만 차량을 안정화하는 것 이외에도 탑승자의 승차감도 고려해줘야 한다. 그러므로 충격이 좀 더 부드럽게 사라져야 탑승자에게 가해지는 충격력이 적다. 그러므로 비록 본문에서 빨리 안정화하는 방향으로 시스템의 최적화를 진행했지만, 실제 서스펜션 시스템에서는 충격이 끝나는데 걸리는 시간을 늘려 진동 제거 성능을 조금 낮추더라도 승차감을 확보하는 방향으로 차량을 제작한다. 그러므로 실제로는 지금까지 본문에서 유도했던 critical damping 이 아닌 underdamping 하도록 세팅 되어 있어 차량이 다소 출렁이는 것을 볼 수 있다.



배운 점 및 느낀 점

이번 Learning journal을 통해 자동차의 서스펜션 시스템의 원리에 대해 자세히 알 수 있었다. 또 실제로 사용되는 시스템의 모델링을 통해 미분방정식을 구축해보는 과정이 신기했다. 미분방정식은 미분방정식을 미정계수법과 eigenvalue를 사용한 방법과, 라플라스 변환을 통한 방법 이렇게 총 두 가지 방법을 통해 일반해를 직접 구해보면서 각 방법의 장단점과 차이점을 좀 더 확실히 느낄 수 있었다. 추가적으로 MATLAB의 여러 함수 (roots, relidue, ilaplace, step, figure 등)를 사용하면서 MATLAB과 좀 친해진 기분이 들어서 뿌듯했다. 계산 실수가 많이 나올 수 있는 부분을 MATLAB이 해줘서 훨씬 수월하게 일반해를 구할 수 있었다. 이번 러닝저널의 분량이 많은 만큼 굉장히 의미 있는 활동을 많이 해서 미래에 도움이 확실히 많이 될 것 같다. 동역학이 벌써부터 기대 반 걱정 반으로 다가왔다는 것에 실감이 되었다.

참고문헌

<출력귀환제어에 의한 차량 능동 현가시스템의 성능 분석 시뮬레이션>

- 김재열 김영석 박기형 장종훈 유신
- 한국공작기계기술학회 '96년도 춘계학술대회 논문집

< Robust Vehicle Suspension System by Converting Active & Passive Control of a Vehicle to Semi-Active Control System Analytically >

- Hassan Elahi, Asif Israr, M. Zubair Khan, and Shamraiz Ahmad
- Journal of Automation and Control Engineering Vol. 4, No. 4, August 2016

<Optimization of Vehicle Suspension System Using Genetic Algorithm >

- Sikandar Khan , Mamon M. Horoub , Saifullah Shafiq , Sajid Ali , Umar Nawaz Bhatti
- 2019 IEEE 10th international Conference on Mechanical and Aerospace Engineering