궤도차량의 실시간 시뮬레이션을 위한 동운동 모델 개발

안재준* · 오중석* · 윤석준**

Development of a Dynamic Model for Real-Time Simulation of Tracked Vehicle

Jae-joon Ahn, Jung-seok Oh and Sug-joon Yoon

Abstract

궤도차량 운동의 실시간 시뮬레이션을 위해서 실시간 시간제약 조건을 만족하기 위한 모델링을 시도하였다. 단위 블록 열차로 연결된 전체 차체를 질점으로 가정하여 모델을 단순화함으로써 차량 설계 시 사용되는 궤도 차량 모델보다 실시간성을 향상시키고자 하였다. 당 모델에서 궤도차량의 운동은 트랙의 형상에 구속되는데, 차량의 운동 궤적은 궤도의 형상에 의해 결정된다. 단, 매 단위 시간마다에서 궤도 차량의 위치는 뉴턴의 제2법칙으로 결정된다. 본 모델링은 몇 개의 단위 트랙을 설정하여 이들의 조합으로 전체트랙을 설계할 수 있도록 사용자가 임의로 설계한 전체트랙에서 바로 실시간으로 시뮬레이션할 수 있는 특징을 갖는다. 당 궤도차량 운동 모델은 전동차, 철도, 롤러코스터의 궤도 설계 등에 사용될 수 있으며, 주 용도는 롤러코스터 게임 시뮬레이터에서 동 특성을 비교적 엄밀하게 시뮬레이션 하는데 있다.

Key Words : Modeling, Simulation, Tracked vehicle

1. 서론

자연 현상을 이해하는 방법으로 수학적 표현을 이용하여 기술하고, 이러한 수식의 해를 구하여 물리적 현상을 해석한다. 이처럼 자연 현상을 수학이나 기타 다른 언어를 사용하여 표현하는 것을 모델링(Modeling)이라 하고, 표현된 그 자체를 모델이라 한다. 모델을 가상적으로 모의 구현하는 것을 시뮬레이션(Simulation)이라 한다. 시뮬레이션 구현 환경을 HILS(Hardware In-the-Loop Simulation)로 구축한다면 실시간(Real-Time) 시뮬레이션을 적용할 수 있다.

그림 1과 같이 전체적인 시스템의 관할은 실시간 스케줄러가 담당하게 된다. 스케줄러에서

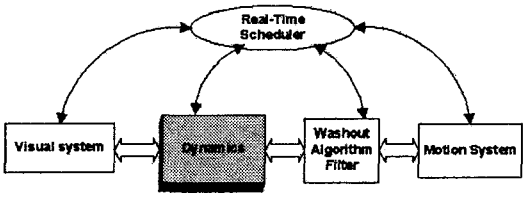


그림 1. HILS환경 시뮬레이션의 개념도

는 동운동 모델에서 연산된 결과들을 일정시간에 맞게 비쥬얼 시스템, 워시아웃 필터와 모션 시스템 에 전달해준다. 이 때 중요한 것이 바로 실시간 시간 제약 조건이다. 한 모듈에서 시간 지연은 전체 시스템을 이루는 다른 모듈들과 시간 지연이 있게되고, 이 시간 오차는 누적되어 전체 시뮬레이션 시간에 적지 않은 시간 지연으로 나타나게 된다. 곧, 실시간 시뮬레이션을 위해서는 가능한 운동 모듈의 연산량을 줄여야 한

* 세종대학교 대학원 항공우주공학과
** 세종대학교 공과대학 기계항공우주공학부

다. 본 논문은 감성공학 기술을 이용한 롤러코스터 게임용 시뮬레이터 개발과정에서 동적 모델링 영역에서 그림 1과 같이 전체 실시간 시스템의 요구사항을 만족하도록 모델링하였다.

기존의 궤도 차량의 모델은 대부분 차체를 강체(Rigid Body)로 가정하여 대상의 운동을 자세히 묘사한 반면에 매우 연산량이 많아서 시뮬레이션을 위해서는 연산 부하가 높아진다. 곧, 연산에 필요한 시스템의 하드웨어 성능의 요구사항이 높아지고 이는 전체 시스템의 개발가격을 높아지게 해준다.

본 연구에서는 동운동 모델을 롤러코스터 운동에서 블록 열차들로 연결된 전체 차량을 하나의 질점(Particle)으로 가정하여 모델링하였다. 따라서, 단순화된 운동 모델은 일반 PC에서도 실시간 시뮬레이션을 허용하며, 또한 궤도차량의 훈련용이나 오락용 시뮬레이터에서 요구되는 정도의 정확성을 제시할 수 있었다.

2. 단위 트랙 모델

2.1 단위 트랙의 구분

감성 공학을 적용한 롤러코스터 게임용 시뮬레이터 개발의 일부분인 이 운동 모델은 게임 시뮬레이터의 특성상 사용자가 몇 개의 정해진 단위 트랙들을 조합해서 전체 트랙을 완성하고 그 시나리오 상에서 시뮬레이터가 시뮬레이션 할 수 있도록 구성되어있다. 따라서 동운동 모듈도 그에 맞추어 단위 트랙 별로 운동 연산을 할 수 있도록 구성되었다.

열차의 운동을 단위 트랙별로 계산하기 위해서 모두 7가지의 단위 트랙을 설정하였다. 직선구간, 터널 구간, 오르막 언덕, 내리막 언덕, 왼쪽 커브, 오른쪽 커브, 루프마다의 트랙의 형상 데이터는 그림 2와 같이 동운동 모델의 입력 값으로 정해진다.

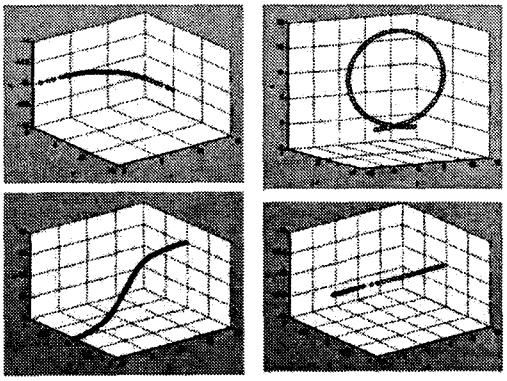


그림 2. 형상별 단위 트랙의 궤적

이러한 작업은 전체 시스템 상에서 비쥬얼 시스템에서 먼저 형성된 값들이다. 각 단위 트랙을 그래픽으로 표현하기 위해서 샘플링된 각 점마다의 위치 좌표 x, y, z와 각 축에서의 회전각도 ϕ, θ, ψ를 계산하여, 모두 6개의 값들을 각 단위 트랙마다 데이터 테이블로 형성하여 동운동 모델의 입력부로 넘겨준다. 이 데이터 테이블은 각 단위 트랙마다 존재하며, 동운동 모델의 연산은 이 데이터들을 근간으로 계산된다. 이 때에 그래픽 좌표계와 동운동 모델의 좌표계가 서로 다르므로 좌표를 변환해서 같은 좌표계 상에서 계산해 주어야한다. 기본적으로 운동 모델의 기준 좌표계는 그림 3과 같다[1].

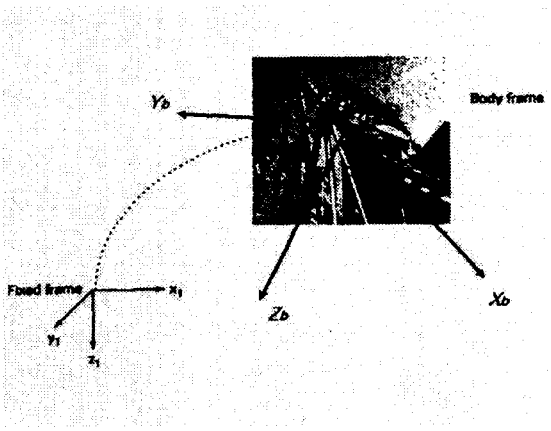


그림 3. 절대기준좌표계와 동체좌표계

2.2 단위 트랙의 모델링

롤러코스터의 운동에서 차제에 작용하는 힘은 중력과 마찰력뿐이고 열차는 트랙의 형상에 구속된 운동을 하기 때문에 트랙의 형상은 곧 열차의 운동을 결정한다. 열차의 병진운동과 회전운동은 바로 트랙의 형상으로 결정된다 할 수 있다. 열차의 운동은 트랙에 구속된 1자유도 운동으로 볼 수 있기 때문에 매 시간마다의 열차의 위치와 자세는 트랙의 형상에 의해서 결정된다. 기본적으로 중력이 작용하는 계(system)는 절대기준좌표계이고 이를 열차에 작용하는 힘으로 환산하려면 다음의 식(1)과 같은 오일러 회전 변환의 정의를 이용하여 동체좌표계로 변환할 수 있다.

$$[Reference_{Frame}] = [\phi][\theta][\phi][Body_{Frame}] \quad (1)$$

위의 식에서 회전 행렬의 역변환을 이용하면 동체좌표계에서 열차에 작용하는 힘을 계산할 수 있다. 관성계에서의 질량체에 걸리는 힘은 뉴턴의 제2법칙으로 식(2)와 같이 정의되므로 여기서 힘과 질량을 안다면 가속도를 구할 수 있다[2].

$$F = m \times a \quad (2)$$

식(2)에서 구하는 가속도는 동체좌표계에서 열차에 작용하는 가속도이다. 이 경우 열차가 트랙 위를 진행 시에 트랙과 열차 사이의 마찰력을 생각할 수 있다. 마찰력은 동체좌표계에서 열차의 진행방향의 반대로 작용하는 힘이므로 식(3)과 같이 표현할 수 있다. 마찰계수 μ는 트랙의 재질에 따라 달라질 수 있는 값이다.

$$F_{x_{friction}} = F_{x_{body}} - \mu N \quad (3)$$

곧, 식(2)에서의 힘의 항은 식(1)의 역변환으로 구할 수 있는데 그 중에서 진행방향 성분은 식(3)과 같이 마찰력을 적용하여 더욱 엄밀한 모델을 설정한다. 열차의 움직임은 트랙에 구속되므로 진행방향이외에는 유효한 힘이 작용하지 않는다. 결국 식(2)에서의 힘 성분은 식(4)와 같이 나타내지는데, 열차 전체를 질점으로 생각했으므로 식(2)에 식(4)를 대입하고 양변을 질량으로 나누어주면 가속도항만 남는다.

$$F = [\, F_{x_{friction}} \; 0 \; 0 \,]^{T} \quad (4)$$

이는 열차에 미치는 가속도 성분이 되는데, 열차의 동체 좌표에서의 값이므로 이를 절대기준좌표계로 환산하면 열차의 병진 가속도를 알 수 있다. 이를 적분하여 열차의 속도와 위치를 매 기준 시간마다 알 수 있다.

2.3 MATLAB/SIMULINK code

전체 트랙이 단위 트랙들의 조합으로 이루어지므로 동운동 모듈자체도 각 단위 트랙의 계산 유형에 따라 몇 가지 유형으로 나뉘어진다. 오르막 언덕, 내리막 언덕, 루프 구간에서의 계산 유형은 동일하므로 하나의 계산 모듈로 연산 처리하였다. 그림 4에서처럼 하나의 질점에 작용하는 가속도는 절대기준좌표계에서는 접선방향 가속도가 된다. n번째와 n+1번째 점 사이의 거리를 s라하고 각 점의 좌표와 미소구간에서의 초기속도를 알면 식(5)와 같이 만들 수 있다.

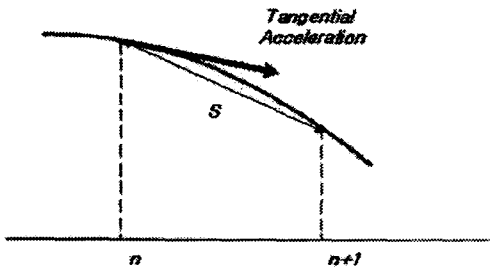


그림 4. 미소구간에서의 접선가속도

$$s = v_0 t_{e+} \frac{1}{2} a\, t_e^{\,2} \tag{5}$$

식(5)의 해를 구하면 식(6)과 같다.

$$t_e = \frac{-v_0 \pm \sqrt{v_0^{\,2} + 2as}}{a}, (t_e > 0) \tag{6}$$

가속도는 식(2)와 식(4)로부터 알 수 있으므로 알고 있으므로 식(7)과 같이 미소구간에서의 열차의 진행방향 속도를 알 수 있다.

$$V = a \times t_e \tag{7}$$

속도 역시 접선방향 속도이므로 이를 절대기준좌표에서의 각 성분으로 나눈다면, 다음과 같다.

$$v_x = V\cos\theta\sin\phi \tag{8}$$

$$v_y = V\sin\phi \tag{9}$$

$$v_z = V\sin\theta \tag{10}$$

식(8),(9),(10)의 속도 성분들을 식(6)에서 구한 소요시간과 곱해주면 x, y, z축 방향으로의 위치를 알 수 있다. 이 모델을 MATLAB/SIMULINK로 만들어 그림 5와 같이 나타내었다[3][4][5]. 직선 구간, 터널 구간도 연산 과정이 같지만 그림 5에서와 x축 방향의 위치를 나타내는 적분과정이 다르다[6]. 왼쪽 커브, 오른쪽 커브도 연산 과정은 x축은 직선에서의 연산과 같고 y축 방향으로는 속도성분을 분해하여 계산하는 과정은 그림 5와 같다. 그러나 커브 단위 트랙에서는 열차가 운동할 때는 Yawing과 Rolling 운동이 함께 일어나고, 원심력, 중력과 트랙의 열차에 대한 수직 항력이 힘 평형을 이루어야 하므로 식(11)과 같은 관계가 성립된다.

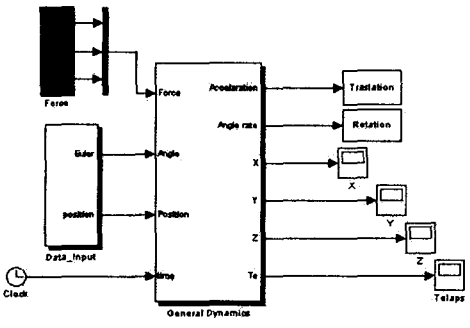


그림 5. 단위트랙의 SIMULINK 모델

$$\phi = \tan^{-1}\left(\frac{v^2}{\rho g}\right) \tag{11}$$

v는 커브 구간의 진입속도, ρ는 곡률반경, g는 중력가속도이다. 식(11)을 이용하여 커브구간의 선회각(Bank angle)을 결정할 수 있다. 곧, 단위 트랙의 계산 유형을 모두 3가지로 나누었고 그에 따른 계산 모듈도 따로 구성하였다.

3. 단위 트랙 통합

3.1 단위 트랙의 통합

지금까지 완성된 단위 트랙 모듈들에 대하여 MATLAB/SIMULINK상에서 통합을 시도하였으나 프로그램 언어상의 한계와 동운동 모델의 요구조건 때문에 단위트랙 통합 개발 환경을 바꾸지 않을 수 없었다. MATLAB/SIMULINK는 모델링 과정에서 간편함을 제공해 주었지만 실시간 시뮬레이션을 위한 코드 개발 환경은 제공해주지 못했다. 그래서 트랙의 통합 환경은 ANSI-C를 적용하였다. 앞에서 단위 트랙의 모델을 C-code를 적용하여 새로운 원시코드를 작성하였고, 컴파일러 및 작업환경은 Microsoft Visual C++ 6.0을 사용하였다[7][8]. 각각의 단위 트랙은 사용자가 단위 트랙을 조합할 수 있는 형태로 통합을 시도하였다. 사용자는 지정된 단위 트랙의 번호를 입력하여 전체 트랙을 기호에 맞게 구성할 수 있는 것이다.

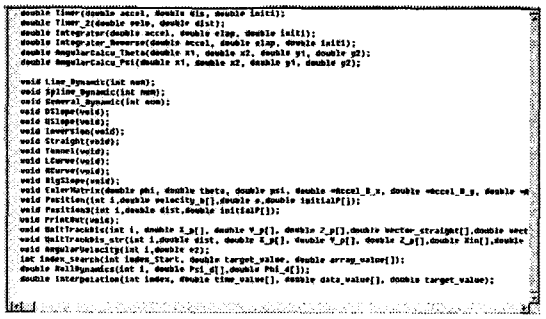
3.2 통합 모델링의 요소 기술

단위 트랙별로 모델링된 모듈들을 이용하여 전체 모델링으로 통합할 때 중요한 것은 단위 트랙 사이의 데이터들의 상호연결이다. 앞의 단위 트랙의 마지막 위치는 뒤의 단위 트랙의 초기 위치가 되므로 이러한 값들은 서로간에 연결이 이루어져야 한다. 2개의 단위 트랙을 서로 연결하기 위해서는 열차의 위치, 자세, 회전 방향, 진행 소요시간 등을 고려해주어야 한다.

열차의 자세는 트랙의 형상으로 결정이 된다고 했으나 자세 값을 매 단위 시간당 결정하려면 결국은 위치 좌표를 이용하여 자세각(Attitude angle) ϕ, θ, ψ를 새롭게 연산해주어야 했다. 자세각을 계산해주는 과정에서 단위 트랙 루프(Loop)는 pitch 각이 90° 가 넘게되면 삼각함수의 특성상 기준 좌표계가 90° 회전하는 현상이 나타났다. 이 현상을 해결하기 위해서 θ값에 대해서 별도로 연산하는 함수를 만들었다. 또한 커브 단위 트랙에서는 Rolling과 Yawing이 함께 일어나는 현상을 열차의 자세로 나타내기 위해서도 역시 별도로 ϕ, ϕ를 연산하였다. 자세각을 연산하는 방법은 벡터의 내적을 이용하였다. 두 벡터의 사이각을 계산하여 이를 누적해주는 방법을 사용하였다. 자세각에 대한 확인을 위해서 롤러코스터 테스트용 프로그램을 제작하였다. OpenGL 그래픽 라이브러리를 사용하였기 때문에 그래픽 좌표계에서의 회전 순서와 동운동 모델에서의 회전 순서가 서로 달라서 의도하지 않은 회전이 추가적으로 생겼다. 이를 보완하기 위해서 그래픽 좌표계는 객체(Object) 좌표계를 사용하고 동운동 모델에서 행렬의 순서를 맞추어 주었다. 그 결과 그림 6과 같은 가시적인 테스트를 시행할 수 있었다.

동운동 모델에서 열차의 위치 좌표 x, y, z, 열차의 자세각 ϕ, θ, ψ, 좌표변환을 이용한 열차의 병진 가속도 a_x, a_y, a_z, 열차의 자세각의 변화율인 각속도 p, q, r 과 진행시간 t의 총 13개의 변수 값들이 연산을 통하여 결과 값으로 나온다.

그림6. 커브구간에서의 열차운동



```
double Timer(double accel, double dis, double init[]);
double Timer_2(double velo, double dist);
double Integrator(double accel, double elap, double init[]);
double Integrator_Reverse(double accel, double elap, double init[]);
double AngularCalcu_Theta(double x1, double x2, double y1, double y2);
double AngularCalcu_Psi(double x1, double x2, double y1, double y2);

void Line_Dynamic(int num);
void Spline_Dynamic(int num);
void General_Dynamic(int num);
void DSlope(void);
void USlope(void);
void LowerPlane(void);
void Straight(void);
void Tunnel(void);
void LCurve(void);
void RCurve(void);
void BigSlope(void);
void ColorMatrix(double phi, double theta, double psi, double xAccel_B_x, double xAccel_B_y, double xA
void Position(int i,double velocity_b[],double p,double initialP[]);
void Position2(int i,double dist,double initialP[]);
void PrintOut(void);
void WaitTrack1(int i, double X_p[], double Y_p[], double Z_p[],double Vector_straight[],double Vect
void WaitTrack1_str(int i,double dist, double X_p[], double Y_p[], double Z_p[],double Rio[],double
void AngularVelocity(int i,double *Z);
int Index_Search(int Index_Start, double target_value, double array_value[]);
double RollDynamics(int i, double Psi_d[],double Phi_d[]);
double Interpolation(int Index, double time_value[], double data_value[], double target_value);
```

그림 7. Dynamics 모듈의 헤더 파일

이러한 계산 방법은 미소구간의 길이를 알고 그 구간을 지나갈 때의 소요시간을 구했으나, 시간 간격이 모두 일정치 않으므로 선형 보간(Interpolation)을 통하여 일정한 샘플링 간격으로 보간해주어야 했다. 동운동 모듈에서 계산된 데이터를 일정한 시간 간격을 기준으로 다른 시스템에 전달해주어야 하기 때문에 보간 루틴도 필요하다. 이러한 요소 기술들을 함수로 표현하여 그림 7에서처럼 원시 코드를 작성하였다.

4. 수치실험

본 논문은 롤러코스터의 운동을 단위 트랙 별로 모델링 하여 사용자가 단위 트랙의 조합을 이용하여 전체 트랙에서 시뮬레이션 할 수 있는 동운동 모듈을 개발과정을 기술하였다. 그러나 시뮬레이션 결과가 과연 얼마나 정확한지, 얼마나 믿을 만한지를 알고 있어야 이 동운동 모델

의 충실도(fidelity)를 부여할 수 있다. 이 때문에 정량적(quantitative)인 면과 정성적(qualitative)인 면에서 결과 데이터를 비교, 검증하였다. 정량적인 면에서는 트랙과 열차사이의 마찰력을 고려하지 않았을 경우에 생각해보았다. 처음의 큰 언덕(BIG SLOPE)에서 지면까지의 낙하 운동에서 에너지 보존 법칙을 생각해보면, BIG SLOPE의 높이가 210m이므로 위치에너지와 운동에너지의 보존 법칙을 생각해보면, 지면까지 열차가 내려왔을 때 속도는 $64.189(m/s^2)$가 되야 한다. 시뮬레이션 결과를 데이터 테이블로 추출하면 지면에서의 속도는 $64.289(m/s^2)$가 되어 정량적으로 0.156(%)의 오차를 가진다는 것을 확인하였다. 정성적으로는 트랙에 대한 그래픽스 데이터가 존재함으로 시뮬레이션 데이터와 비교하여 그림 8과 같이 그 경향성이 같음을 알 수 있었다.

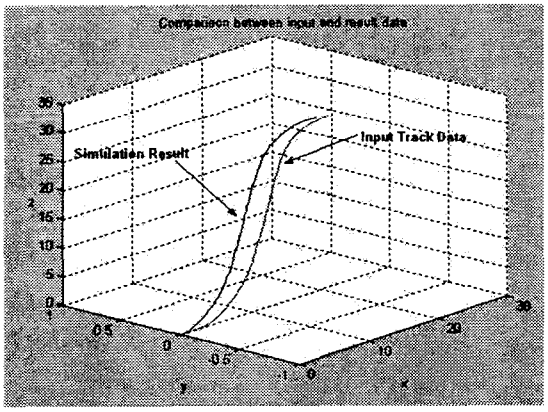


그림 8. 시뮬레이션 결과의 검증

이처럼 롤러코스터의 동운동 모델이 어느 정도의 엄밀한 시뮬레이션을 할 수 있다는 것을 알 수 있었다. 다음은 마찰력을 고려했을 때 열차의 운동의 궤적을 그림 9에 나타내었고, 그 운동에서의 가속도, 자세각, 각속도를 각각 그림 10, 그림 11, 그림 12와 같이 나타내진다는 것을 알 수 있었다.

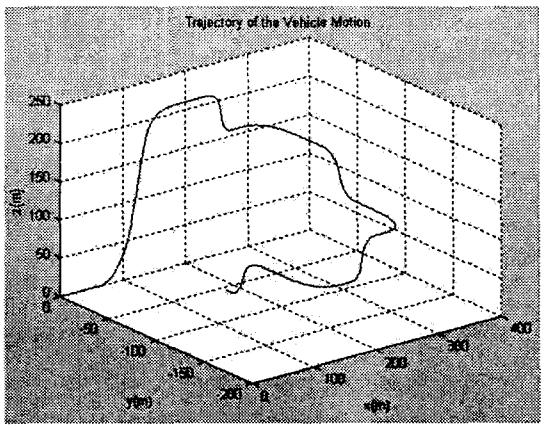


그림 9. 열차의 운동 궤적

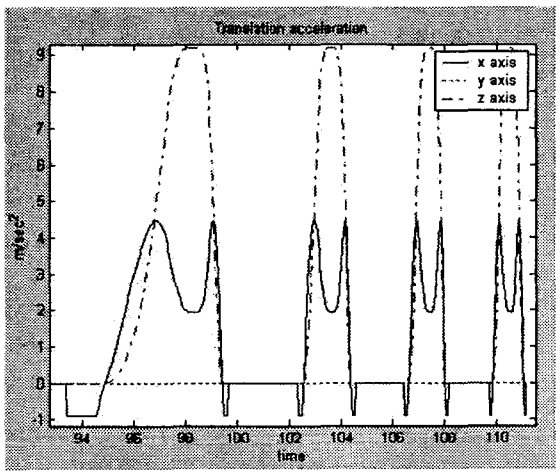


그림 10. 열차의 병진 가속도

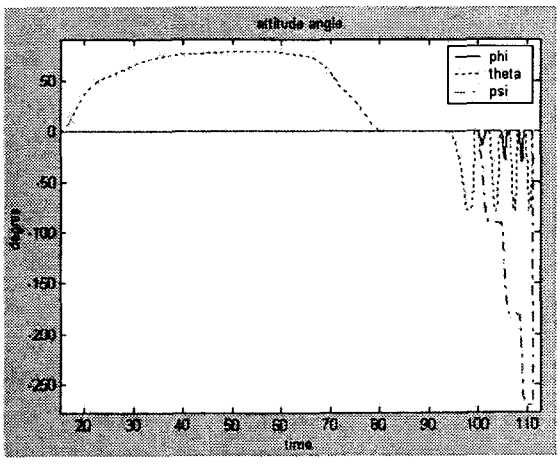


그림 11. 열차의 자세각

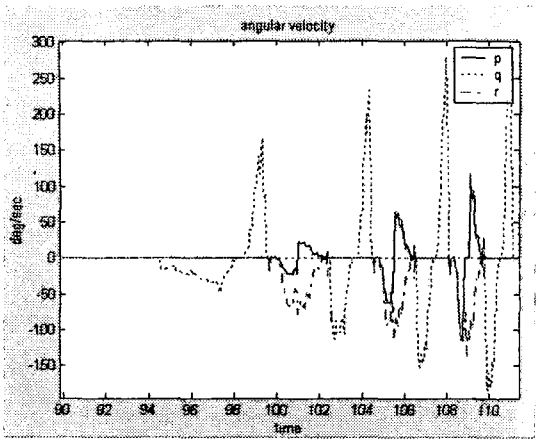


그림 12. 열차 운동의 각속도

5. 결론 및 향후과제

일반적인 궤도차량의 동운동 모델은 열차를 강체로 가정하여 모델링하였다. 본 논문에서는 실시간 시뮬레이션을 위한 궤도 차량의 동운동을 열차 전체를 하나의 질점을 가정하여 모델링하였다. 그러므로 본 모델은 시뮬레이션에서 소요되는 연산 시간이 기존의 궤도차량 모델보다 매우 적게 소요되며, 이러한 이유에서 HILS 환경하에서 실시간으로 시뮬레이션하기에 적합한 형태라 할 수 있다.

롤러코스터의 동운동 모델에서 좀더 발전시켜 할 점은 ANSI-C를 기반으로 프로그래밍된 모델을 객체 지향형식인 C++를 사용하여 변환 해주어야 하고, MATLAB/SIMULINK 환경에서 시험평가 할 수 있도록 C 작성된 S-function을 사용하여 동운동 모델의 평가 환경을 구축해야 한다.

후기

본 연구는 KISTEP에서 주관하는 G7과제 중 감성공학 기반 기술 개발 사업의 지원으로 수행되었으며, 지원에 감사드립니다.

참고문헌

1) Robert C. Nelson, "Flight Stability and Automatic Control", 2nd Edition, McGraw-Hill, 1998
2) Ferdinand P. Beer, "Vector Mechanics for Engineers : DYNAMICS", 3rd SI Metric Edition, McGraw-Hill, 1999
3) Duane Hanselman, Bruce Littlefield, "Mastering MATLAB 5", Prentice-Hall, 1998
4) MATLAB/SIMULINK Reference Manual version 6.1 , Release 12
5) SIMULINK version 3 reference manual, the MathWorks, 1999
6) Robert W. Hornbeck, "Numerical Method", Prentice-Hall/Quantum, 1997
7) Haruhiko Hayashi, "C 언어 입문(기초편)", 영진출판사, 2001.
8) MSDN Library April, 2000