Projectile Trajectory Maker

With Air Resistance & Spin Resistance

최선우

프로그램 설명

우선 이 프로그램은 2차 혹은 3차원에서 공기 저항 및 회전에 의한 마그누스 힘을 고려한 투사체의 궤적을 그리고 그 투사체의 주어진 정보와 투사체가 이동한 평면상 거리, 소요된 시간, 가장 높이 올라간 높이, 평면상 평균 속력을 구하는 프로그램이다.

프로그램 사용법 설명

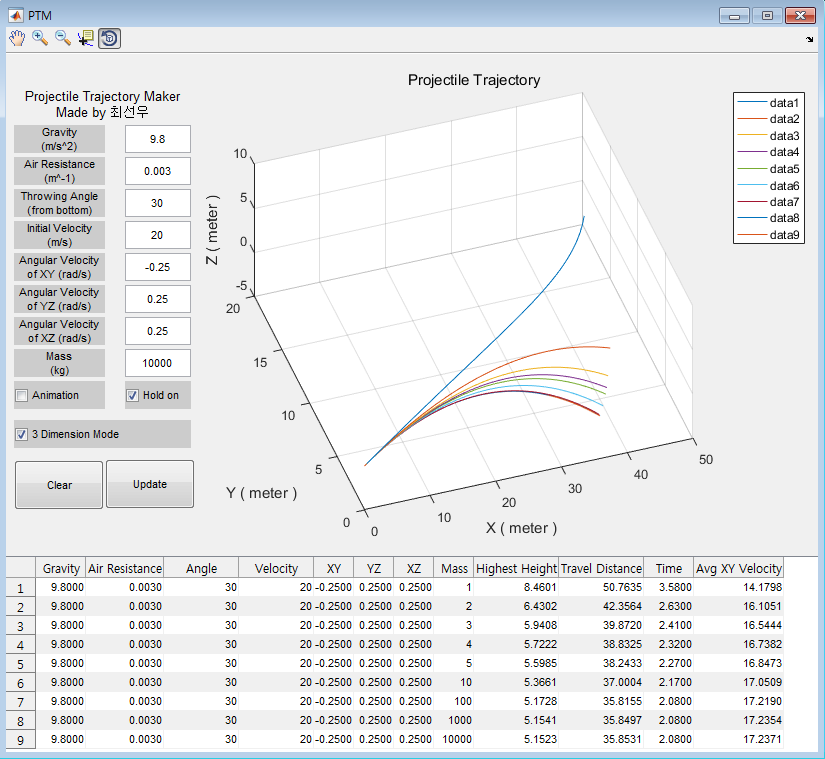
초기에 화면에 많은 흰색 칸들이 있는데 전부 수정 가능하다, 단 각도를 조절하는 칸은 보다 직관적인 사용을 위하여 Z 축에서 내려오는 각도가 아닌 땅에서부터 몇도 벌어지느냐 이다. 체크 박스의 경우 애니메이션 체크 박스, 홀드 온 체크 박스, 3 dimension 체크 박스가 있는데 애니메이션 박스의 경우 체크 될 경우 업데이트시 궤적의 경로를 실시간으로 보여주며 애니메이션이 끝나기 전까진 클리어나 업데이트 버튼을 누르지 못한다. 홀드온 체크 박스의 경우 모든 업데이트에 대하여 홀드 온 하며 아래 데이터 히스토리에 기록이 추가된다. 순서는 오래된 데이터가 1번 이며 추가하면 2번, 3번에 들어간다. 3 dimension 체크 박스의 경우 체크 할 경우 XZ 축 회전뿐만이 아닌 YZ 와 XY 회전을 추가해서 계산 하며 플롯창의 앵글을 바꿔서 보여준다. 3 dimension 체크 박스를 풀어서 업데이트 할 경우 XZ 축 회전을 제외한 회전 데이터는 무시되며 플롯창도 X축과 Z축을 기준으로 2차원으로 보게 된다.

알고리즘 설명

우선 이 프로그램은 객체 지향 프로그래밍으로 제작하였으며 MATLAB 의 기본 제공되는 guide 기능을 이용하여 객체 지향 프로그래밍 경험을 기억 하여 완성하였다. 메인 함수는 PTM ( Projectile Trajectory Maker ) 이며 drawer 은 2차원 플롯 함수, drawer3d 는 3차원 플롯 함수이다. drawer 함수는 입력 인수로 중력, 공기 저항 계수, 발사 각도, 초기 속력, XZ축 각속도, 질량, 애니메이션 여부 총 7개를 받는다. drawer3d 함수는 입력 인수로 중력, 공기 저항 계수, 발사 각도, 초기 속력, XZ축 각속도, YZ축 각속도, XY축 각속도, 질량, 애니메이션 여부 총 9개를 받는다. drawer 함수는 XZ 를 제외한 축의 각속도는 데이터를 쓰더라도 0으로 무시해서 저장하지만 3차원 플롯모드에서는 무시하지 않는다.

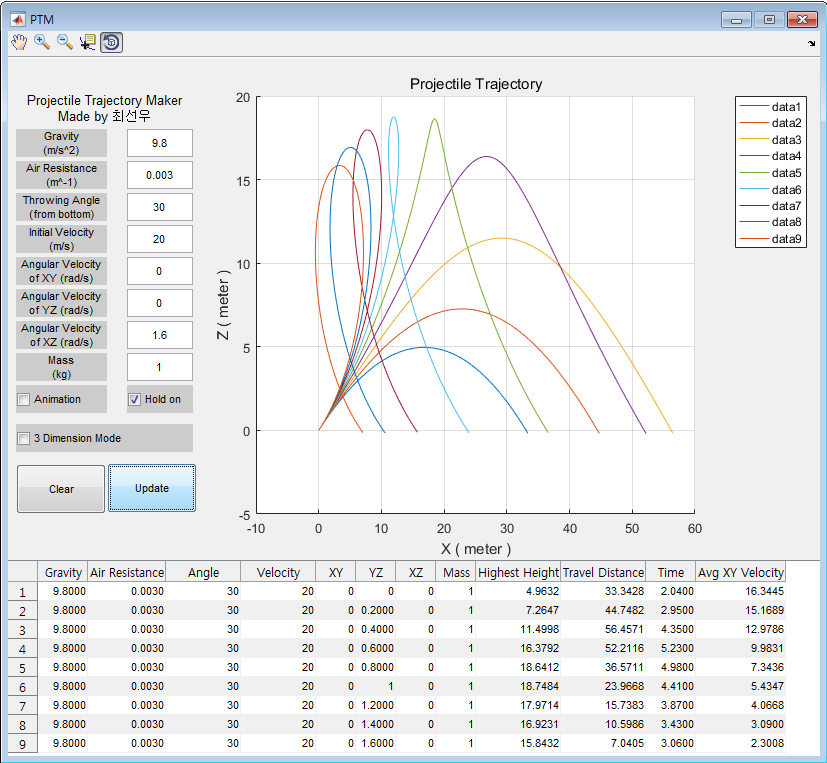
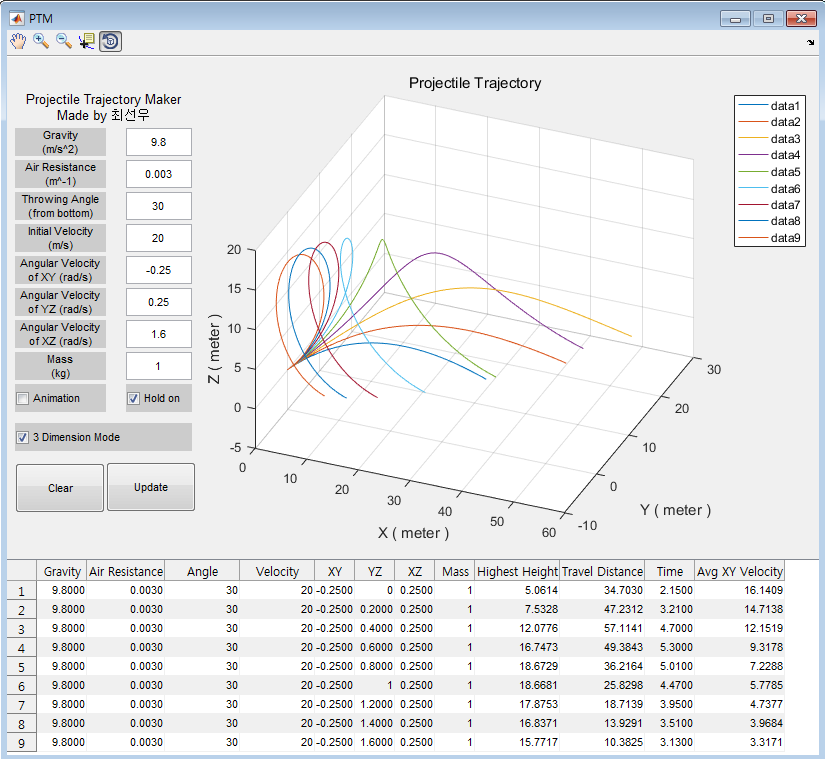
실제 활용해서 얻은 결과

우선 실험 데이터를 기록할 수 있다는 장점을 활용하여 다양한 조건에서 궤적을 그리고 데이터를 확보해 보았다.

그 결과 모든 조건을 동일시 하였을 때 질량이 증가하여도 일정 질량 이상이 되면 궤적은 별로 변하지 않았다. 이는 질량이 증가하여도 초기 속도는 같고 받는 중력 가속도 또한 같기 때문이다.

(x,y,z) = (0,0,0) 에서 출발해서 +x 방향 +z 방향으로 날아갈 때 XY 축 각속도는 얼마나 옆으로 휠지를 결정하고 XZ축 각속도는 얼마나 아래로 내려갈지 혹은 위로 솟구칠지를 결정하며 YZ 축 각속도는 투사체의 XY 축 각속도에 의한 영향만큼은 아니지만 옆으로 휘는 것에 조금 영향을 주었다.

위 세가지 회전축을 잘 조합해본 결과 야구 선수들이 던지는 변화구를 재현해 볼 수 있었고 축구 선수 들이 차는 공을 재현 하는 놀라운 경험을 할 수 있었다.

다른 조건들이 동일 할 때 XZ 축 각속도에 변화를 주었을 때 투사체가 공중에서 한 바퀴 돌고 내려오는 현상도 일어났다. 단 이는 회전이 지속적으로 이루어지는 이상적 환경이기 때문에 현실과는 차이가 있다.

Travel distance 즉 높이가 아닌 이동한 거리를 보여주는데 이를 통해 어떤 회전을 주고 어떤 무게, 초기 속도 와 각도를 조절해서 가장 효율적 조건에 대한 가늠이 가능했다. 가령 축구공처럼 붕 떠서 계속 날아가는 것을 예로 들면 XZ 회전축에서 진행방향에 대해 윗 공기와 같은 방향으로 회전을 주면 굉장히 멀리 날아간다. 즉 축구공을 찬다고 생각하면 축구공의 아래를 찬다고 생각하면 된다.

알고리즘 및 프로그램의 개선 여지

투사체의 애니메이션을 실제 계산된 시간과 같은 시간만큼 흐르도록 만들고 싶었으나 프로그램이 구동되는 컴퓨터 마다 다른 성능을 갖기 때문에 일관적이지 못할 것이라는 생각을 하였고 또한 시간이 10초 이상만 되어도 굉장히 지루한 투사체의 모습을 보고 있어야 하므로 실제로 누가 10초 이상 실제 시간으로 애니메이션을 보고 있을까 의문이 들어서 구현하지 않기로 하였다.

데이터를 기록해 둘 수 있는 프로그램의 특징을 활용하여 두 개 이상의 투사체가 동시다발적으로 날아 가는 궤적 애니메이션을 구현해 보고자 하였는데 조건이 다를 경우 각각의 배열의 사이즈가 달라지고 이로 인해 차원이 달라 오류가 난다. 이 부분은 배열 사이즈가 가장 큰 배열 기준으로 나머지 배열에 빈칸에 NaN 을 넣어서 배열 차원을 맞춰보려 하였으나 아직 구현에 성공하지 못 하였고, thread 개념을 이용해 보려 하였으나 매트랩에는 그런 기능이 잘 나와있지 않았다.