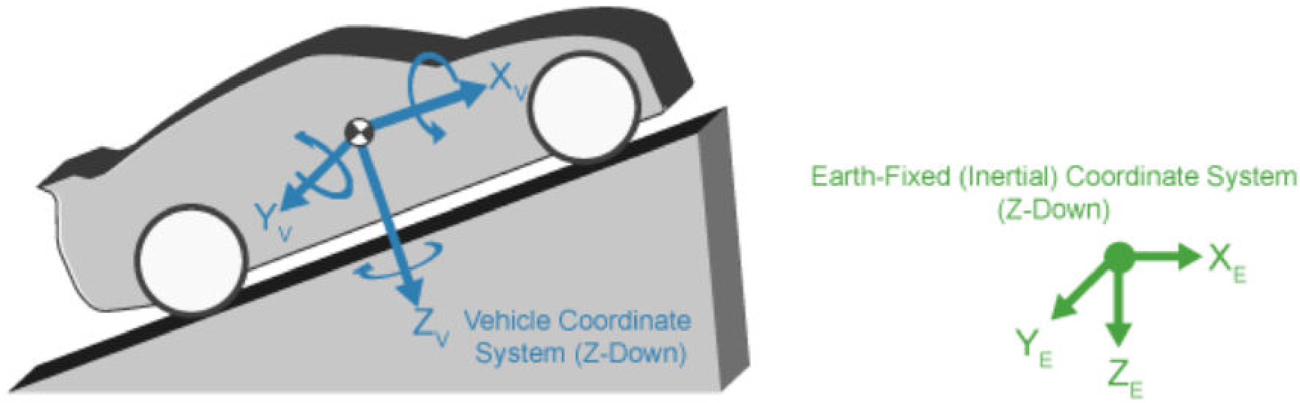
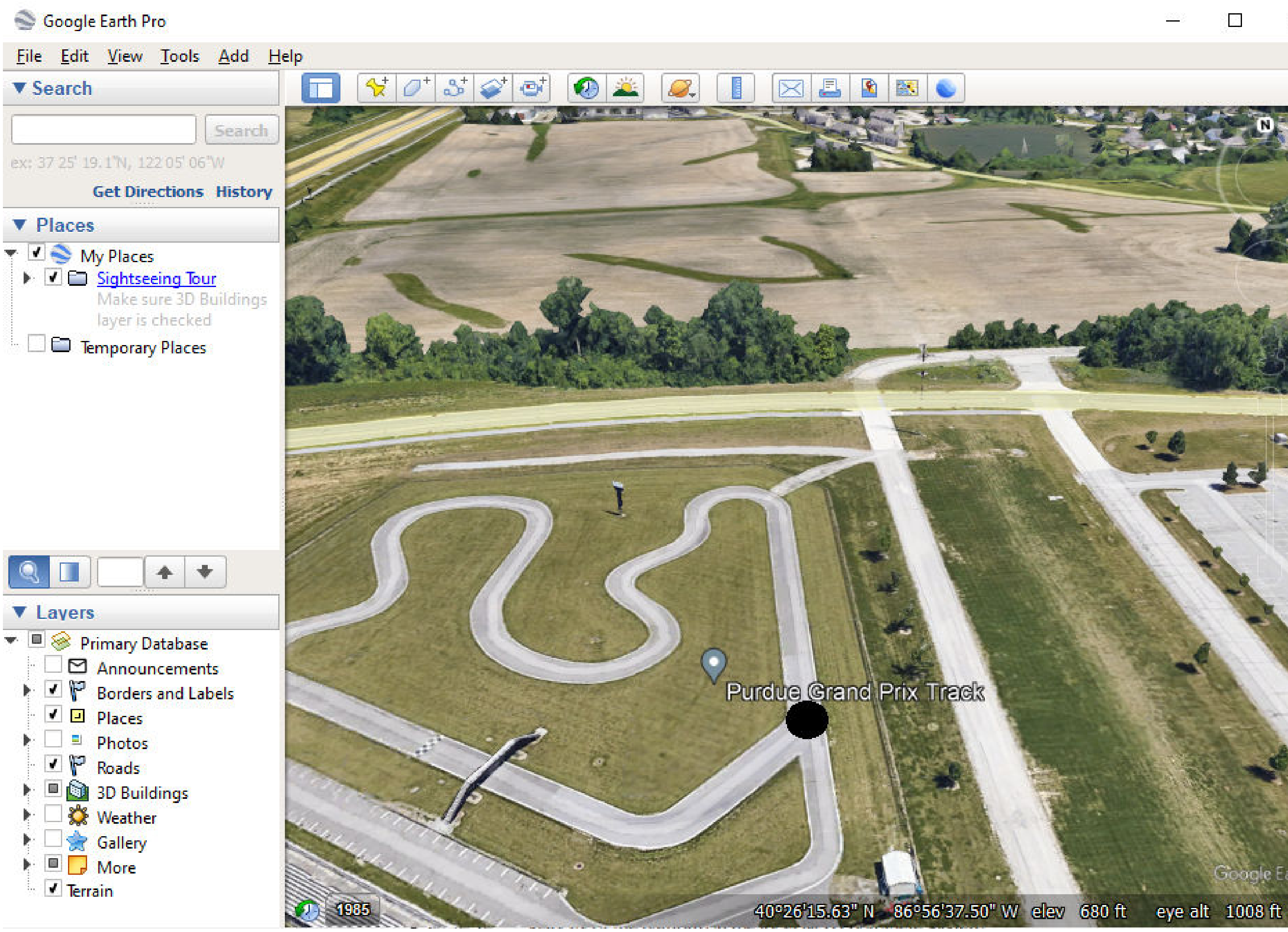
```
%% Extract Raw Data
Digital_Signals = out.Digital_Signals;
Vehicle_Signals = out.Vehicle_Signals;
```




```
% Orientation using NED (Earth-Fixed Coordinate System)
Orientation = Digital_Signals.Navigation_Sensors.ang.Data(:,:);
Orientation = quaternion(Orientation, 'euler', 'XYZ', 'frame');

% Angular Velocity using Vehicle Coordinate System
AngularVelocity = Digital_Signals.Navigation_Sensors.ang_vel.Data(:,:);

% Position using NED (Earth-Fixed Coordinate System)
Position = Digital_Signals.Navigation_Sensors.pos.Data(:,:);

% Velocity using NED (Earth-Fixed Coordinate System)
VelocityX = Vehicle_Signals.veh.InertFrm.Cg.Vel.Xdot.Data(:);
VelocityY = Vehicle_Signals.veh.InertFrm.Cg.Vel.Ydot.Data(:);
VelocityZ = Vehicle_Signals.veh.InertFrm.Cg.Vel.Zdot.Data(:);

% Acceleration using VNED (Vehicle Coordinate System)
AccelerationX = Vehicle_Signals.veh.BdyFrm.Cg.Acc.xddot.Data(:);
AccelerationY = Vehicle_Signals.veh.BdyFrm.Cg.Acc.yddot.Data(:);
AccelerationZ = Vehicle_Signals.veh.BdyFrm.Cg.Acc.zddot.Data(:);

%% Process Data
num_samples = 1001;
count = 1;
orient = zeros(num_samples,1,'quaternion');
vel = zeros(num_samples,3);
acc = vel;
angVel = vel;
pos = vel;

trajectory = waypointTrajectory(Position, 'TimeOfArrival',(0:t:15), 'Orientation',Orientation,
tInfo = waypointInfo(trajectory);
```

```
while count < num_samples+1
   [pos(count,:),orient(count),vel(count,:),acc(count,:),angVel(count,:)] = trajectory();

   count = count + 1;
end

%% Create Trajectory Struct (Sample rate of Simulation)
PER22_Ground_Truth.Fs = 1/t;
PER22_Ground_Truth.gpsFs = 1/t;
```

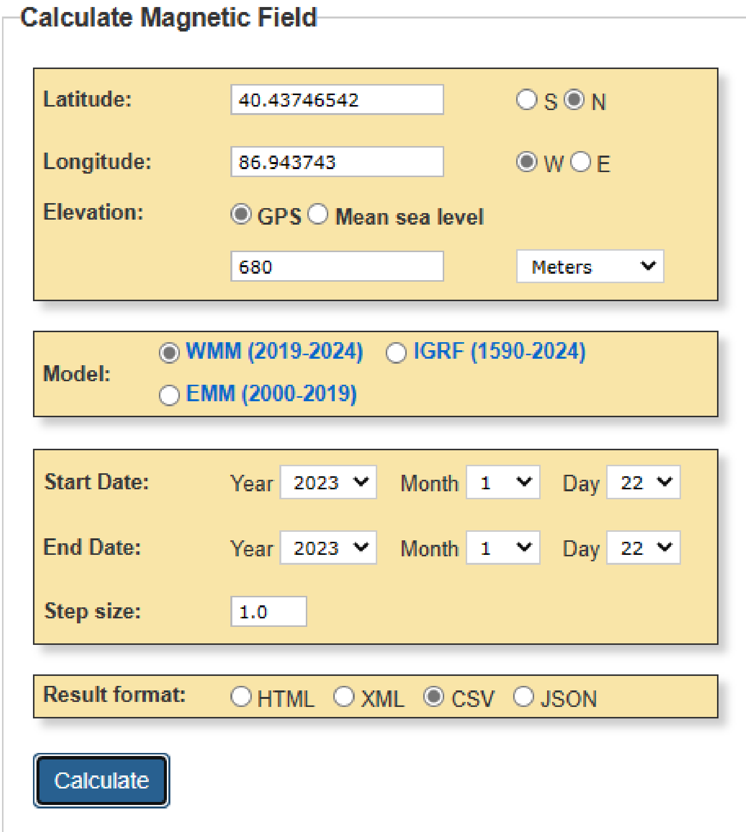
Degree to Decimal LLA Calculator




```
% LLA of grand prix track
PER22_Ground_Truth.refloc = [40.437675, -86.943750, 680]; % [deg deg m]
```

Magnetic Field Calculator

More Information About Geomagnetic Fields | NCEI (noaa.gov)




```
% XYZ magnetic field components at grand prix track
% note that magnetic north and true north are not equal
PER22_Ground_Truth.magField = [19.9577 -1.5449 48.4109]; % [uT uT uT]
```

- q_0, q_1, q_2, q_3 -- Parts of orientation quaternion. The orientation quaternion represents a frame rotation from the platform's current orientation to the loca NED coordinate system.
- $angVel_X, angVel_Y, angVel_Z$ -- Angular velocity relative to the platform's body frame.
- $position_N, position_E, position_D$ -- Position of the platform in the local NED coordinate system.
- ν_N, ν_E, ν_D -- Velocity of the platform in the local NED coordinate system.
- $accel_N, accel_E, accel_D$ -- Acceleration of the platform in the local NED coordinate system.
- $accelbias_X, accelbias_Y, accelbias_Z$ -- Bias in the accelerometer reading.
- $gyrobias_X, gyrobias_Y, gyrobias_Z$ -- Bias in the gyroscope reading.
- $geomagneticFieldVector_N, geomagneticFieldVector_E, geomagneticFieldVector_D$ -- Estimate of the geomagnetic field vector at the reference location.
- $magbias_X, magbias_Y, magbias_Z$ -- Bias in the magnetometer readings.

```
% Orientation using NED (Earth-Fixed Coordinate System)
PER22_Ground_Truth.trajData.Orientation = Orientation;

% Position using NED (Earth-Fixed Coordinate System)
PER22_Ground_Truth.trajData.Position = Position;

% Velocity using NED (Earth-Fixed Coordinate System)
PER22_Ground_Truth.trajData.Velocity = vel;

% Angular Velocity using VNED (Vehicle Coordinate System)
```

```
PER22_Ground_Truth.trajData.AngularVelocity = AngularVelocity;

% Acceleration using NED (Earth-Fixed Coordinate System)
PER22_Ground_Truth.trajData.Acceleration = acc;

% plot((0:t:15), acc(:,2))
% plot((0:t:15), angVel(:,3))
% hold on
% plot(out.yaw_error.Time, PER22_Ground_Truth.trajData.AngularVelocity(:,3))

clearvars -except PER22_Ground_Truth
save("ground_truth_data.mat");
```