Function to replace the Haversine Distance Function. Vincenty gives a much more accurate result.

In [1]:
```
from math import atan, tan, sin, cos, sqrt, radians, atan2

def vincenty_distance(lat1, lon1, lat2, lon2):
    """
    Calculate the great-circle distance between two points on the Earth surface given
    """
    # WGS-84 ellipsiod parameters
    a = 6378137.0  # semi-major axis in meters
    f = 1 / 298.257223563  # flattening
    b = (1 - f) * a  # semi-minor axis

    # convert decimal degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    # calculations
    U1 = atan((1 - f) * tan(lat1))
    U2 = atan((1 - f) * tan(lat2))
    sinU1 = sin(U1)
    cosU1 = cos(U1)
    sinU2 = sin(U2)
    cosU2 = cos(U2)

    lon_diff = lon2 - lon1
    Lambda = lon_diff  # initial approximation for Lambda
    sinLambda = sin(Lambda)
    cosLambda = cos(Lambda)

    # iterate until change is insignificant
    for _ in range(1000):
        sinSigma = sqrt((cosU2 * sin(Lambda)) ** 2 + (cosU1 * sinU2 - sinU1 * cosU2 *
        cosSigma = sinU1 * sinU2 + cosU1 * cosU2 * cos(Lambda)
        sigma = atan2(sinSigma, cosSigma)
        sinAlpha = cosU1 * cosU2 * sin(Lambda) / sinSigma
        cos2Alpha = 1 - sinAlpha ** 2
        cos2SigmaM = cosSigma - 2 * sinU1 * sinU2 / cos2Alpha
        C = f / 16 * cos2Alpha * (4 + f * (4 - 3 * cos2Alpha))
        Lambda_prev = Lambda
        Lambda = lon_diff + (1 - C) * f * sinAlpha * (sigma + C * sinSigma * (cos2Sigm
        # break if change in Lambda is insignificant
        if abs(Lambda - Lambda_prev) < 1e-12:
            break

    # final calculations
    u2 = cos2Alpha * (a ** 2 - b ** 2) / (b ** 2)
    A = 1 + u2 / 16384 * (4096 + u2 * (-768 + u2 * (320 - 175 * u2)))
    B = u2 / 1024 * (256 + u2 * (-128 + u2 * (74 - 47 * u2)))
    deltaSigma = B * sinSigma * (cos2SigmaM + B / 4 * (cosSigma * (-1 + 2 * cos2SigmaM

    # distance in meters
    s = b * A * (sigma - deltaSigma)

    #distance in kilometers
    s = round(s / 1000)
```

```
    return s
```