```
function d = vincenty_dist(lat1,long1,lat2,long2)
%VINCENTY_DIST Geodetic distance between 2 points on WGS84 ellipsoid
%   Inputs are:
%   lat1  :a scalar origin latitude in degrees
%   long1  :a scalar origin longitdue in degrees
%   lat2  :a numeric array Mx1 destination latitude in degrees
%   long2  :a numeric array Mx1 desination longitdue in degrees
%
%   Output is:
%   d      :a numeric array of Mx1 distance to target in m
%
%   Reference: https://en.wikipedia.org/wiki/Vincenty%27s_formulae

    arguments
        lat1 {mustBeScalarOrEmpty, mustBeNumeric, mustBeReal}
        long1 {mustBeScalarOrEmpty, mustBeNumeric, mustBeReal}
        lat2 (:,1) {mustBeNumeric, mustBeReal}
        long2 (:,1) {mustBeNumeric, mustBeReal}
    end

%   WGS84 properties
    a = 6378137.0; % WGS84 semi-major axis
    b = 6356752.314245; % WGS84 semi-minor axis
    f = 1/298.257223563; % WGS84 flattening

%   Convert degrees to radians
    lat1 = deg2rad(lat1);
    long1 = deg2rad(long1);
    lat2 = deg2rad(lat2);
    long2 = deg2rad(long2);

%   Convert range of longitude to 0->2*pi
    long1 = mod(long1,2*pi);
    long2 = mod(long2,2*pi);

%   Find reduced latitude
    U1 = atan((1-f)*tan(lat1));
    U2 = atan((1-f)*tan(lat2));

%   Calculate difference in longitude
    L = abs(long2-long1);
    if L > pi
        L = 2*pi - L;
    end

%   Iterate to find difference in longitude on auxilliary sphere
    lam = L;
    lam_old = 0;
    ii = 0;
    while ~ii || all(abs(lam-lam_old) > 1e-12)
        ii = ii+1;
```

```
        lam_old = lam;

        sinsigma = sqrt((cos(U2).*sin(lam)).^2+(cos(U1).*sin(U2)-sin(U1)...
            .*cos(U2).*cos(lam)).^2);
        cossigma = sin(U1).*sin(U2)+cos(U1).*cos(U2).*cos(lam);
        sigma = atan2(sinsigma,cossigma);
        alpha = asin(cos(U1).*cos(U2).*sin(lam)./sin(sigma));
        cos2sigma_m = cos(sigma)-2.*sin(U1).*sin(U2)./cos(alpha).^2;

        C = f./16.*cos(alpha).^2.*(4+f.*(4-3.*cos(alpha).^2));
        lam = L+(1-C).*f.*sin(alpha).*(sigma+C.*sin(sigma)...
            .*(cos2sigma_m+C.*cos(sigma).*(-1+2.*cos2sigma_m.^2)));

        if lam > pi
            warning('Points are antipodal');
            lam = pi;
            break
        end
    end

%   Use converged values to find distance
    u_sq = cos(alpha).^2.*(a.^2-b.^2)./b.^2;
    A = 1+u_sq./16384.*(4096+u_sq.*(-768+u_sq.*(320-175.*u_sq)));
    B = u_sq./1024.*(256+u_sq.*(-128+u_sq.*(74-47.*u_sq)));
    delta_sigma = B.*sin(sigma).*(cos2sigma_m+B./4.*(cos(sigma)...
        .*(-1+2.*cos2sigma_m.^2)-B./6.*cos2sigma_m.*...
        (-3+4.*sin(sigma).^2).*(-3+4.*cos2sigma_m.^2)));
    d = b.*A.*(sigma-delta_sigma);

end
```