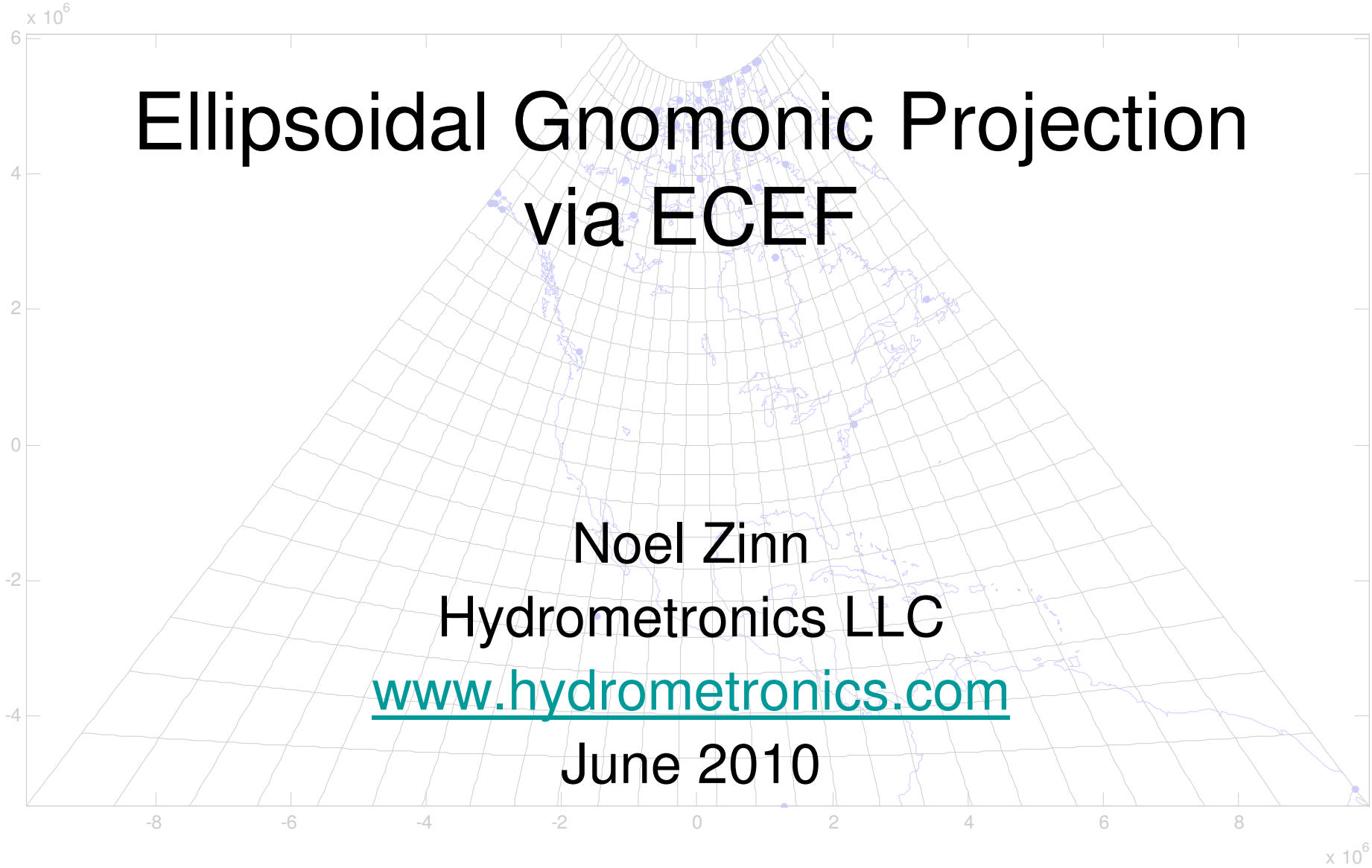


Ellipsoidal Gnomonic Projection via ECEF

Noel Zinn
Hydrometronics LLC
www.hydrometronics.com

June 2010

Algorithm

The following algorithm for an ellipsoidal gnomonic is a direct perspective from the geocenter through the ellipsoid onto the tangential plane via ECEF, topocentric coordinates and a little matrix algebra. No intermediate mappings. No truncations. No approximations.

Choose an oblique point of tangency (lat, lon, hgt=0). Compute the ECEF XYZ as is widely described (see EPSG link below). Define that as (0,0,0) topocentric UVW. Chose a point to the east (100000,0,0) and a point to the north (0,100000,0). Convert these points from topocentric UVW to ECEF XYZ. EPSG Guidance Note 7-2 Section 2.2.2 will do (www.epsg.org) or use the equations on page 3 of Bugayevskiy and Snyder (less elegant). You now have three points in ECEF XYZ that define the tangential plane. Any point on the ellipsoid can be converted to ECEF XYZ. The geocenter is (0,0,0) in ECEF XYZ. Those two points define the perspective ray. Now, we need the point of intersection of the ray with the plane. Wikipedia (http://en.wikipedia.org/wiki/Line-plane_intersection) provides the answer, which requires the inversion of a 3x3 matrix, but we've just computed all the required xyz points. Multiply the parameter 't' (throw away 'u' and 'v') by the ECEF XYZ coordinates of the point on the ellipsoid to get the ECEF XYZ coordinates on the tangential plane. Convert those XYZ coordinates to topocentric UVW. The W will be zero (because we're on the plane). The U and V coordinates are our ellipsoidal gnomonic Easting and Northing.

Matlab code that accomplishes the essentials follows.

Essential Matlab Code – Page 1

```
% Load coastline culture in Matlab format from NOAA
load namer.dat
data = namer;
len = length(data);
dataGnom = zeros(len,3);
% Useful constant for converting radians to degrees
% pi is an environment constant in Matlab
d2r = pi / 180;
A = 6378137;
RF = 298.257223563;
% Solve for the parameters of the ellipsoid
% Solve for flattening
F = 1 / RF;
% Solve for semi-minor axis
B = A - F*A;
% Solve for eccentricity squared
E2 = (A^2 - B^2) / A^2;

% Local Horizontal point
lat0 = 40;
lon0 = -100;
hgt = 0;
% Get ECEF of tangent point X0Y0Z0
[X0 Y0 Z0] = ECEF03(lat*d2r, lon*d2r, hgt, A, E2);
P0 = [X0 Y0 Z0];
```

Essential Matlab Code – Page 2

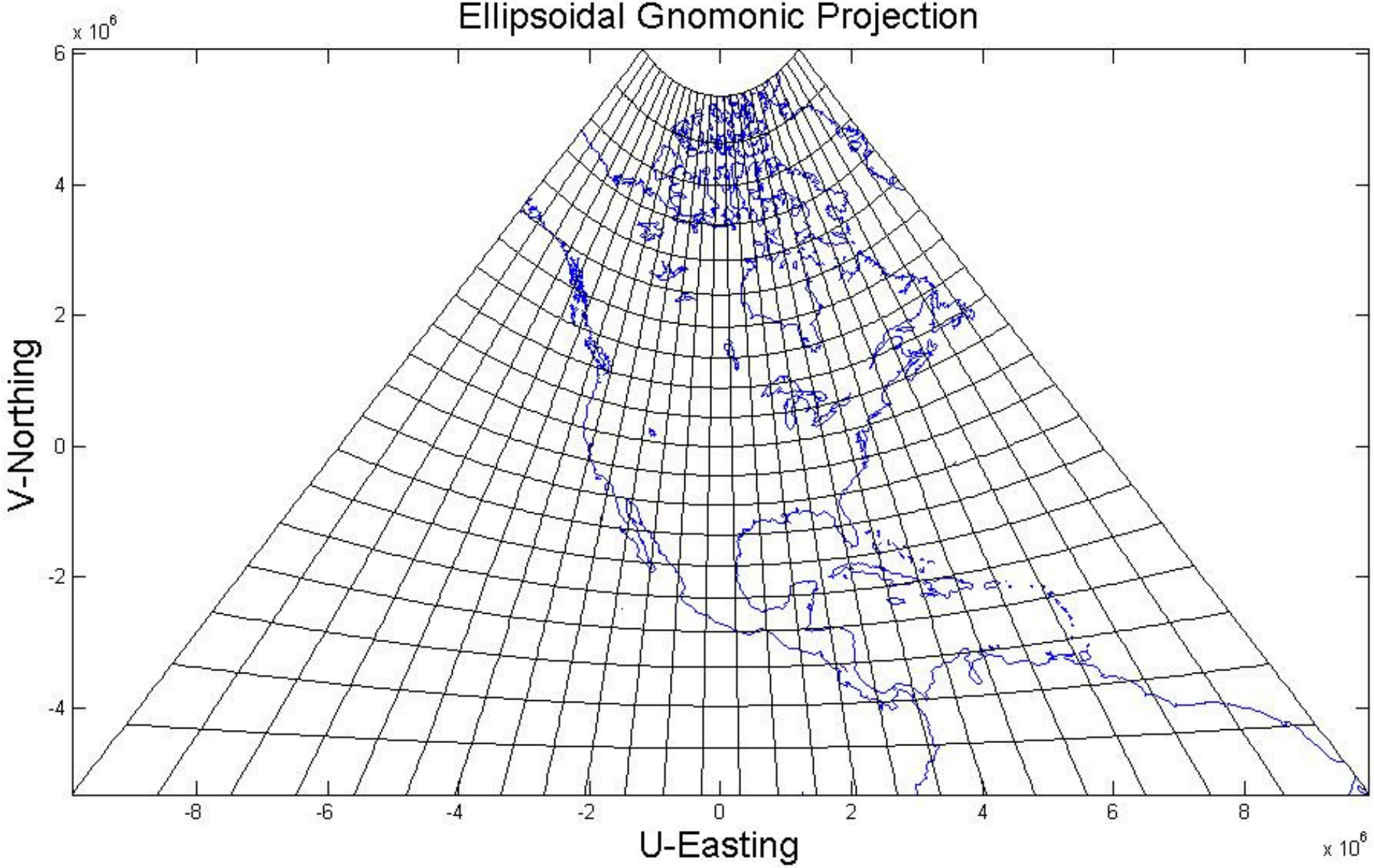
```
% Define topocentric rotation into UVW
lat0rad = lat0*d2r;
lon0rad = lon0*d2r;
CC = [ -sin(lon0rad)                cos(lon0rad)                0
       -sin(lat0rad)*cos(lon0rad)   -sin(lat0rad)*sin(lon0rad)  cos(lat0rad)
        cos(lat0rad)*cos(lon0rad)   cos(lat0rad)*sin(lon0rad)   sin(lat0rad)];
invCC = inv(CC);

% Get ECEF for P1 (100000,0,0) and P2 (0,100000,0) in UVW
P1 = invCC*[100000; 0; 0] + P0';
P2 = invCC*[0; 100000; 0] + P0';
% P0, P1 & P2 define the tangent plane
```

Essential Matlab Code – Page 3

```
% Convert lat/lon/hgt to Gnomonic
tic  % Start timer
% Coastlines
for dex = 1:len
    % Point on the ellipsoid in ECEF
    [X Y Z] = ECEF03(data(dex,2)*d2r, data(dex,1)*d2r, 0, A, E2);
    if isnan(X)
        dataGnom(dex,:) = [nan nan nan]';
    else
        % Define the intersection matrix CUT and its inverse
        CUT = [-X P1(1)-P0(1) P2(1)-P0(1)
            -Y P1(2)-P0(2) P2(2)-P0(2)
            -Z P1(3)-P0(3) P2(3)-P0(3)];
        invCUT = inv(CUT);
        % Solve for the intersection parameter
        param = invCUT*[-P0(1); -P0(2); -P0(3)];
        % param
        % Solve for ECEF XYZ on the plane
        thisECEF = [X Y Z]*param(1);
        % Convert to topocentric (gnomonic)
        gnom = CC*(thisECEF-[X0 Y0 Z0])';
        dataGnom(dex,:) = gnom;
    end
end
```

ECEF Function

```
function [X, Y, Z] = ECEF03(LATRAD, LONRAD, HGT, A, E2)
% Out variables in bracket to the left
% Input variables in parentheses to the right
% LATRAD, LONRAD are lat/lon in radians
% HGT is ellipsoidal height
% Note: "^" in Matlab (and BASIC) is "**" in FORTRAN
% A is semi-major axis
% RF is reciprocal of flattening
% Change the input geographical coordinates to ECEF
V = A / sqrt(1 - E2*sin(LATRAD)^2);
X = (V + HGT)*cos(LATRAD)*cos(LONRAD);
Y = (V + HGT)*cos(LATRAD)*sin(LONRAD);
Z = (V*(1 - E2) + HGT)*sin(LATRAD);
```


Ellipsoidal Gnomonic Projection
x 10^6
6
4
2
0
-2
-4
V-Northing
-8
-6
-4
-2
0
2
4
6
8
U-Easting
x 10^6