Satellite Photogrammetry HW3
due Friday 25 Mar.

(1) Create matlab function $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = fi2g(im\#, l, s, h, dp)$ which implements our image to ground projection algorithm. Create a "main" script which loads needed support data, calls fi2g with our test point, and confirm that misclosures $(\delta e, \delta n)$ agree with HW2 results. We will not use arguments im# and dp for now.

(2) Create function $\begin{bmatrix} \phi \\ \lambda \end{bmatrix} = fi2g\text{-}pl(im\#, l, s, h, dp)$ This is just a wrapper for fi2g + conversion from cartesian to geodetic coordinates. You are welcome to use the provided functions xyz2geo.m, plh2enu.m, and (for later) ftmgeo_utmz13.m.

(3) Create function $\begin{bmatrix} \Delta\phi \\ \Delta\lambda \end{bmatrix} = fi2g\text{-}pl\_0(im\#, l, s, h, \phi, \lambda, dp)$ This is just a wrapper for fi2g-pl + a subtraction. The output is just the difference between the input ϕ, λ and the ϕ, λ computed from l, s, h. That is,

$$\begin{bmatrix} \Delta\phi \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \phi \\ \lambda \end{bmatrix} - fi2g\text{-}pl(im\#, l, s, h, dp)$$

(4) Create function $\begin{bmatrix} l \\ s \end{bmatrix} = fg2i(im\#, \phi, \lambda, h, dp)$, this will call fi2g-pl_0 but we switch the knowns and unknowns. Now ϕ, λ, h are known and l, s are to be computed. We do by iterative inversion.

(a) Input ϕ, λ, h, start with initial approximations $(l^\circ, s^\circ) = (0, 0)$.

→ (b) compute $\begin{bmatrix} F_\phi \\ F_\lambda \end{bmatrix} = fi2g\text{-}pl\_0(im\#, l^\circ, s^\circ, h, \phi, \lambda, dp)$

(c) compute partial derivatives $\partial F_\phi/\partial l$, $\partial F_\phi/\partial s$, $\partial F_\lambda/\partial l$, $\partial F_\lambda/\partial s$ numerically:

$$\begin{bmatrix} \partial F_\phi/\partial l \\ \partial F_\lambda/\partial l \end{bmatrix} = \frac{fi2g\text{-}pl\_0(im\#, l^\circ + \Delta l, s^\circ, h, \phi, \lambda, dp) - fi2g\text{-}pl\_0(im\#, l^\circ, s^\circ, h, \phi, \lambda, dp)}{\Delta l}$$

$$\begin{bmatrix} \partial F_\phi/\partial s \\ \partial F_\lambda/\partial s \end{bmatrix} = \frac{fi2g\text{-}pl\_0(im\#, l^\circ, s^\circ + \Delta s, h, \phi, \lambda, dp) - fi2g\text{-}pl\_0(im\#, l^\circ, s^\circ, h, \phi, \lambda, dp)}{\Delta s}$$

(d) Solve for corrections to l°, s°: $\begin{bmatrix} \partial F_\phi/\partial l & \partial F_\phi/\partial s \\ \partial F_\lambda/\partial l & \partial F_\lambda/\partial s \end{bmatrix} \begin{bmatrix} \Delta l \\ \Delta s \end{bmatrix} = \begin{bmatrix} -F_\phi \\ -F_\lambda \end{bmatrix}$

note $\Delta l, \Delta s$ are not the same in steps (c) & (d)!

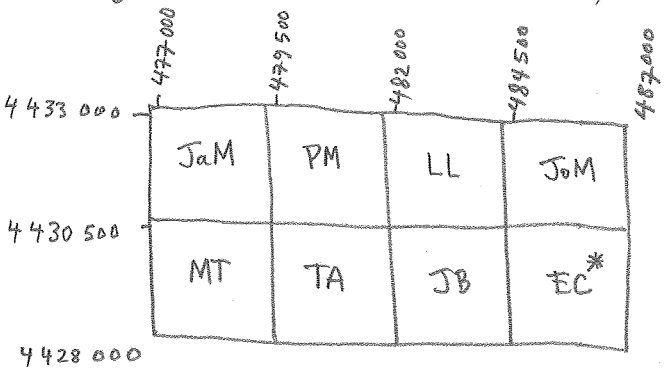
$J \quad \Delta = -F, \quad \Delta = -J^{-1}F$

iteration loop

(e) update ℓ°, s° : $\begin{bmatrix} \ell^\circ_{new} \\ s^\circ_{new} \end{bmatrix} = \begin{bmatrix} \ell^\circ_{old} + \Delta\ell \\ s^\circ_{old} + \Delta s \end{bmatrix}$, iterate until $\Delta\ell, \Delta s$ are small. 3x should do it.

When done with fq2i, verify numerically that it is inverse of fi2q_pl.

(5) use the function created in (4) to make an orthorectification of your assigned tile at 0.5 m GSD, in coordinate system UTM zone 13.



each tile 2500 x 2500 m, 5000 x 5000 pixels. Your top row and left most column will have the shown coordinates (* extra credit)

The DEM is in bc.dem, stored row-wise from N → S

40.1
−105.5 −105.1 721 rows 1441 cols
39.9

read by `fid = fopen('pathname', 'r');`
`D = fread(fid, [1441, 721], 'single');`
`DT = D';` % transpose
`D = DT;`
`clear DT;`

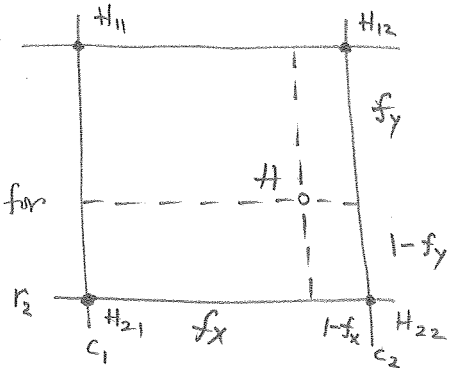
DEM : upper left lat. (dec. deg.) 40.1
upper left long. (dec. deg.) −105.5
interval (dec. deg.) 0.00027777777777778 (1 arc second !)
rows 721
cols 1441

DEM data is orthometric (sea level) height, H, in meters

index arithmetic for height interpolation

$d\phi = \phi_{UL} - \phi$ (deg.)
dsec = dφ / interval
r_1 = fix(dsec) + 1
$r_2 = r_1 + 1$
f_y = dsec − fix(dsec)

Bilinear Interpolation for H at φ, λ



$d\lambda = \lambda - \lambda_{UL}$

$dsec = d\lambda / interval$

$c_1 = fix(dsec) + 1$

$c_2 = c_1 + 1$

$f_x = dsec - fix(dsec)$

$H_{11} = D(r_1, c_1)$;

$H_{12} = D(r_1, c_2)$;

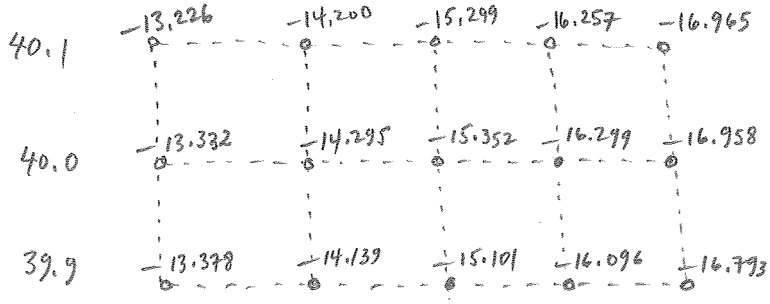
$H_{21} = D(r_2, c_1)$;

$H_{22} = D(r_2, c_2)$;

$$H = (1 - f_x - f_y + f_x f_y) H_{11} + (f_x - f_x f_y) H_{12} + (f_y - f_x f_y) H_{21} + (f_x f_y) H_{22}$$

---

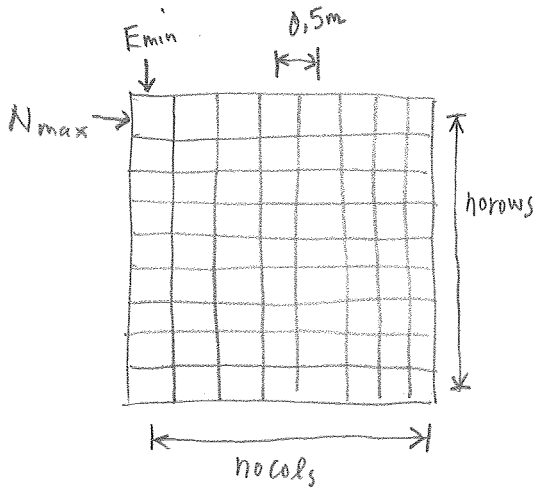
similarly interpolate a value for N (geoid separation) from grid:



from NGS: Geoid12a

Template / Flowchart for orthorectification code

```
in_img = imread('source image', 'TIFF');   or 'JPEG', etc.
[nirows, nicols] = size(in_img);
out_img = zeros(norows, nocols, 'uint8');
for i = norows
    for j = nocols
        E = Emin + (j-1) * 0.5 ;
        N = Nmax - (i-1) * 0.5 ;
        transform to φ, λ
        interpolate H from DEM
        interpolate N from grid
        compute h
        [l; s] = fg2i(im#, φ, λ, h, dp);
```



```
if (l,s) both in range (>2, < max-1)
        interpolate intensity from in_img  = g

else

        g = 128
        end

    out_img (i,j) = g
    end  %  j-loop
end  % i-loop
imwrite (out_img, 'outfile.jpg', 'JPEG');
```

Supplement will be supplied to check by vector overlay

Suggest for debug that you start with larger GSD OR smaller extent, so program executes more quickly. When satisfied run with real numbers.