Map projections and their calculation

The standard projection used in converting latitudes and longitudes on the (roughly) spherical earth to a planar map depends on what you're projecting.

Most people know about the Mercator projection.

The one used extensively for the United States is called the Albers Equal-Area Conic projection.

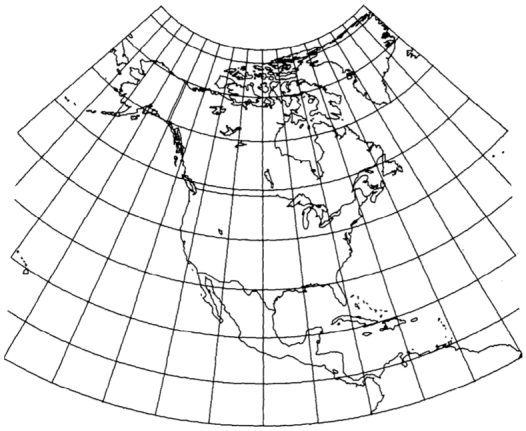

FIGURE 20.—Albers Equal-Area Conic projection, with standard parallels 20° and 60° N. This illustration includes all of North America to show the change in spacing of the parallels. When used for maps of the 48 conterminous States standard parallels are 29.5° and 45.5° N.

`https://en.wikipedia.org/wiki/Albers_projection`

The wikipedia article gives this url

`https://pubs.usgs.gov/pp/1395/report.pdf`

I found the same report referenced in this answer to a question on Stack Exchange.

`https://gis.stackexchange.com/questions/302635/trying-to-implement-albers-projection-302642`

I'll put the code elsewhere (it's always a pain to format for LaTeX). This discussion

is about implementing the math.

The formulas for the sphere are found on page 100, and those for the ellipsoid are on page 102. There is a numerical example worked for each, the spherical case is on page 291, and the ellipsoid on page 292.

spherical case

The idealized spherical earth is a bit easier, so we'll go through that first, and then talk about the earth as an ellipsoid.

What you must do first is to pick two reference latitudes. These are referred to in the following by the letter ϕ. So the reference latitudes are ϕ_1 and ϕ_2.

The standards are: lower 48 states $29.5, 45.5$, Alaska: $55, 65$, and Hawaii $8, 18$.

You also pick a center for the map labeled as longitude ϕ_0 and latitude λ_0.

R is a given constant. For the spherical case it is 1.0.

We use the above constants to calculate

$$n = \frac{1}{2}(\sin\phi_1 + \sin\phi_2)$$
$$C = \cos^2\phi_1 + 2n\sin\phi_1$$
$$\rho_0 = \frac{R}{n}\sqrt{C - 2n\sin\phi_0}$$

n C and ρ_0 are the same for all calculations for a given projection center and reference latitudes.

The next calculation, for ρ, requires each individual ϕ:

$$\rho = \frac{R}{n}\sqrt{C - 2n\sin\phi}$$

θ depends on λ:

$$\theta = n(\lambda - \lambda_0)$$

Given the three values θ, ρ and ρ_0, we can calculate (x, y) according to:

$$x = \rho\sin\theta$$

$$y = \rho_0 - \rho \cos\theta$$

So finally, we can process a series of tuples ϕ, λ to convert them to Cartesian (x, y) coordinates.

Three more values can be calculated: h, k and ω. The first two are the scales of the plot, and the ω is the maximum angular deformation.

This projection preserves areas, but allows deformation of angles.

ellipsoid

There are two additional factors for the ellipsoid, which I'm guessing relate to the semi-major axis and eccentricity: a and e.

q_1 and q_2 are calculated based on the reference latitudes ϕ_1 and ϕ_2.

$$q = (1 - e^2) \left[\frac{\sin\phi}{(1 - e^2 \sin^2\phi)} \right] - \frac{1}{2e} \left[\ln \frac{1 - e\sin\phi}{1 + e\sin\phi} \right]$$

(There was a bit of a challenge interpreting the text as to placement of the brackets. Luckily, there was a numerical example for guidance.)

m depends on each individual ϕ:

$$m = \frac{\cos\phi}{(1 - e^2 \sin^2\phi)^{1/2}}$$

Then

$$n = \frac{m_1^2 - m_2^2}{q_2 - q_1}$$

$$C = m_1^2 + n q_1^2$$

$$\rho_0 = \frac{a\sqrt{C - nq_0)}}{n}$$

As before, n C and ρ_0 are the same for all calculations for a given projection center and reference latitudes.

$$\rho = \frac{a\sqrt{C - nq)}}{n}$$

$$\theta = n(\lambda = \lambda_0)$$

and as before

$$x = \rho \sin\theta$$
$$y = \rho_0 - \rho \cos\theta$$

Inverse formulas are also given for both cases. Let us turn to page 291 of the reference.

numerical example, spherical case

The code is in **spherical.py**. The output is:

```
> python3 sphere.py
p0: 23.0
l0:-96.0
R:  1.00
p1: 29.5
p2: 45.5
n:  0.6028370
C:  1.3512213
R0: 1.5562263
p:  35.0
l: -75.0
R:  1.3473026
t:  12.6595771
x:  0.2952720
y:  0.2416774
>
```

This matches the example on page 291.

ellipsoid

The code is in **ellipsoid.py**. The output is:

```
> python3 ellipsoid.py
test1
p:  23.0
l: -96.0
p1: 29.5
```

```
p2: 45.5
q0: 0.7767080
q1: 0.9792529
q2: 1.4201080
m1: 0.8710708
m2: 0.7021191
n:  0.6029035
C:  1.3491594
R0: 9929079.6
q:  1.1410831
R:  8602328.3
t:  12.6609735
x:  1885472.7
y:  1535925.0
>
```

This matches the example on page 292-293.