




Mollweide Projeksiyonu Üzerine Düşünceler

İbrahim Öztuğ Bildirici*¹ 

¹Konya Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Harita Mühendisliği Bölümü, Konya, Türkiye

Kaynak Göster: Bildirici, İ.Ö. (2025). Mollweide Projeksiyonu Üzerine Düşünceler. Geomatik, 10 (2), 175-182.

DOI: 10.29128/geomatik.1596248

Anahtar Kelimeler

Harita Projeksiyonu
Mollweide Projeksiyonu
Alan Koruyan Projeksiyon
Tissot Endikatrişi
Eliptik Projeksiyon

Araştırma Makalesi

Geliş:04.12.2024
Revize:19.12.2024
Kabul:31.01.2025
Yayınlanma:01.08.2025



Öz

Mollweide projeksiyonu, yeryüzünün tamamının gösterimi amaçlı, alan koruyan gerçek anlamda olmayan silindirik bir projeksiyondur. Yeryüzünü bir elips içerisinde temsil eder. Çoğu kaynakta yalnızca projeksiyon bağıntılarına yer verilmiştir. Bu çalışmada projeksiyon geometrik yapısı ayrıntılı olarak irdelenmiş, projeksiyon fonksiyonlarının nasıl elde edildiği gösterilmiştir. Projeksiyonun yol açtığı deformasyonlar tartışılmıştır. Projeksiyonda kullanılan yardımcı değişkenin bulunması için Python SciPy modülü fonksiyonlarının kullanımı ele alınmış, bu amaçla kullanılabilir üç fonksiyonun karşılaştırması yapılmıştır. Son olarak projeksiyon hesaplamalarına yönelik örnek bir sınıf kodu verilmiştir. Burada ele alınan yazılımsal çözümler enleme bağlı yardımcı değişken kullanan diğer projeksiyonlara da uyarlanabilir.

Thoughts on the Mollweide Projection

Keywords

Map Projection
Mollweide Projection
Equal Area Projection
Tissot Indicatrix
Elliptical Projection

Research Article

Received: 04.12.2024
Revised: 19.12.2024
Accepted: 31.01.2025
Published: 01.08.2025

Abstract

The Mollweide projection is an equal-area pseudo-cylindrical projection used to represent the entire Earth. It represents the Earth in an ellipse. In most sources, only projection equations are given. In this study, the geometric structure of the projection is examined in detail, and how projection functions are obtained is shown. The distortions caused by the projection are discussed in detail. The use of Python SciPy module functions to find the auxiliary variable used in the projection is discussed, and three functions that can be used here are compared. Finally, a sample class code for projection calculations is given. The software solutions discussed here can also be adapted to other projections that use latitude-dependent auxiliary variables.

1. Giriş

Harita projeksiyonu, yeryüzü için tanımlanmış bir referans yüzeyinin düzleme dönüşümü için gerekli iki fonksiyon ile tanımlanır (φ, λ coğrafi koordinatlar, x, y projeksiyon düzlemi koordinatları).

$$x = x(\varphi, \lambda) \quad y = y(\varphi, \lambda) \quad (1)$$

(1) fonksiyonlarının bir harita projeksiyonu tanımlaması için Jakobi determinantının sıfırdan farklı olması gerekir (Bildirici, 2023b; Hoschek, 1984; Üstün ve Bildirici, 2021).

Harita projeksiyonları, kartografyanın ana konularından biri olup, her tür coğrafi bilgi sistemi çalışmasında da temel oluşturur (Doğan ve Yakar, 2018; 2019 Şentürk ve Erenler, 2017; Jonuzi ve ark., 2024).

Harita projeksiyonlarında gerçek anlamı olmayan silindirik projeksiyonlar arasında yer alan Mollweide projeksiyonu alan korur ve yeryüzünü 2:1 oranlı bir elips içinde gösterir. Kutuplar nokta ile gösterilir. Meridyenler elips yayları biçimindedir. Bu özelliği nedeniyle eliptik projeksiyonlar olarak da nitelenen projeksiyon grubu içinde yer alır (Bildirici, 2023b; Bugayevskiy ve Snyder, 1995). 1805 yılında Karl Brandan Mollweide (1774–1825) tarafından sunulmuştur. Mollweide Wolfenbüttel/Almanya’da doğmuş, Halle Üniversitesi’nden mezun olmuştur. Leipzig Üniversitesi’nde astronomi ve matematik profesörü olarak görev yapmıştır (Snyder, 1993; Wagner, 1949; Snyder, 1987).

Projeksiyon 1857’de J. Babinet tarafından "homalographic" adı ile yeniden tanıtılmaya kadar çok bilinmiyordu. Bundan sonra bazı atlaslarda kullanılmaya başlandı. Babinet, Homalographic, Homolographic ve eliptik adları ile anılan projeksiyon 19. yy boyunca yeni bir gerçek anlamı olmayan silindirik projeksiyon olarak dikkat çekti (Snyder, 1993).

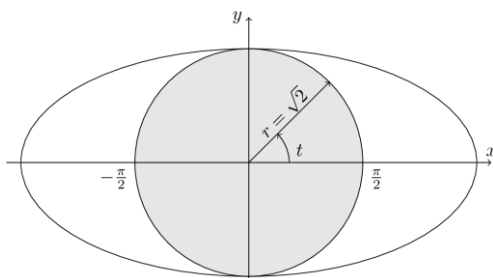
Bu makalede kullanılan notasyon Ek B’de verilmiştir.

2. Materyal ve Metot

2.1. Projeksiyonun Geometrik Yapısı ve Bağlıları

Birim küre varsayımı ile projeksiyon incelenecektir. İlk koşul $-\frac{\pi}{2} \leq \lambda \leq \frac{\pi}{2}$ ile sınırlı yarı kürenin alanı korunarak daire biçiminde gösterilmesidir (Şekil 1’de gri daire). Yarı küre alanı düzlemdeki daire alanına eşitlenir (Hoschek, 1984; Wagner, 1949).

$$2\pi = \pi r^2 \rightarrow r = \sqrt{2} \quad (2)$$



Şekil 1. Mollweide projeksiyonunun tasarımı

$\lambda = \pm \frac{\pi}{2}$ meridyenleri yarım daire, orta meridyen ($\lambda=0$) doğru parçası, diğer meridyenler yarım elipsler biçimindedir. Karşılıklı meridyenler elipsler oluşturur. Bu elipslerin yarıçaplarından biri ortak olup orta meridyenin yarısı kadardır ($\sqrt{2}$). Diğer yarıçap ise boylama bağlı bir fonksiyondur ($f(\lambda)$). Buna göre projeksiyon düzleminde herhangi bir elipsin denklemini yazalım.

$$\frac{x^2}{f^2(\lambda)} + \frac{y^2}{2} \quad (3)$$

Elipsin t parametresine göre parametrik denklemleri (Şekil 1):

$$x = f^2(\lambda) \cos t \quad y = \sqrt{2} \sin t \quad (4)$$

$t = t(\varphi)$, enleme bağlı bir fonksiyondur. $f(\lambda)$ elipslerin x eksenini yöndeki yarıçapıdır. Şekil 1’den,

$$f(0) = 0, \quad f\left(\frac{\pi}{2}\right) = \sqrt{2}$$

olduğu görülür. Buradan,

$$f(\lambda) = \frac{2\sqrt{2}}{\pi} \lambda \quad (5)$$

elde edilir. Projeksiyon eşitlikleri,

$$x = \frac{2\sqrt{2}}{\pi} \lambda \quad y = \sqrt{2} \sin t \quad (6)$$

biçimini alır. $t = t(\varphi)$ fonksiyonunu belirlemek için küre ve düzlemde Gauss temel büyüklükleri ile alan koruma koşulunu yazalım (Bildirici, 2023b; Üstün ve Bildirici, 2021; Hoschek, 1984).

$$\sqrt{EG - F^2} = \sqrt{eg - f^2} \quad (7)$$

E, F ve G küre, e, f ve g projeksiyon düzleminde temel büyüklüklerdir. Birim kürede (Bildirici, 2023b; Üstün ve Bildirici, 2021; Bildirici, 2019):

$$E = 1 \quad F = 0 \quad G = \cos^2 \varphi \quad (8)$$

Projeksiyon düzleminde küre parametreleri (φ, λ) ile temel büyüklükleri bulmak için (6)’dan kısmi türevleri yazalım.

$$\frac{\partial x}{\partial \varphi} = -\frac{2\sqrt{2}}{\pi} \lambda \sin t \quad \frac{dt}{d\varphi} = \frac{2\sqrt{2}}{\pi} \cos t \quad (9)$$

$$\frac{\partial y}{\partial \varphi} = \sqrt{2} \cos t \quad \frac{dt}{d\varphi} = 0$$

Buradan,

$$e = \left(\frac{\partial x}{\partial \varphi}\right)^2 + \left(\frac{\partial y}{\partial \varphi}\right)^2 = \frac{8\lambda^2}{\pi^2} \sin^2 t \left(\frac{dt}{d\varphi}\right)^2 + 2 \cos^2 t \left(\frac{dt}{d\varphi}\right)^2$$

$$= \left(\frac{8\lambda^2}{\pi^2} \sin^2 t + 2 \cos^2 t \right) \left(\frac{dt}{d\varphi} \right)^2$$

$$f = \frac{\partial x}{\partial \varphi} \frac{\partial x}{\partial \lambda} - \frac{\partial y}{\partial \varphi} \frac{\partial y}{\partial \lambda} = -\frac{8\lambda}{\pi^2} \sin t \cos t \frac{dt}{d\varphi}$$

$$g = \left(\frac{\partial x}{\partial \lambda} \right)^2 + \left(\frac{\partial y}{\partial \lambda} \right)^2 = \frac{8}{\pi^2} \cos^2 t \quad (10)$$

elde edilir. (8) ve (10) değerlerini (7)'de yerine koyarsak,

$$\cos \varphi = \frac{4}{\pi} \cos^2 t \frac{dt}{d\varphi} \quad (11)$$

elde edilir. Buradan,

$$\cos \varphi \, d\varphi = \frac{4}{\pi} \cos^2 t \, dt$$

diferansiyel eşitliğini elde ederiz. İki tarafın integrali,

$$\int_0^\varphi \cos \varphi \, d\varphi = \frac{4}{\pi} \int_0^t \cos^2 t \, dt \quad \rightarrow$$

$$\sin \varphi = \frac{4}{\pi} \left(t + \sin t \cos t \right)$$

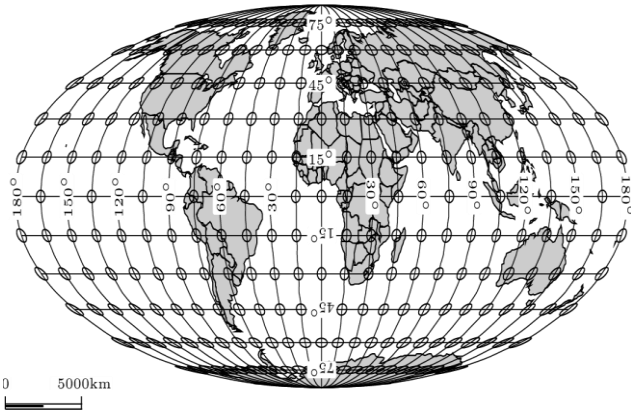
alınarak ve yarım açı bağıntılarından yararlanarak,

$$\pi \sin \varphi = 2t + \sin 2t \quad (12)$$

elde edilir. (12) eşitliği farklı yollardan da elde edilebilir (Bildirici, 2023b; Bugayevskiy ve Snyder, 1995; Hoschek, 1984).

Küre üzerinde bir noktanın koordinatlarını belirlemek için önce (12) eşitliğinden bilinen φ için t değerinin belirlenmesi gereklidir. Ancak t , eşitliğin sol tarafına çekilemediği için nümerik bir kök bulma yöntemi gereklidir. Bu tür problemlerin çözümünde çoğunlukla Newton yönteminden yararlanılır.

Şekil 2'de Mollweide projeksiyonunda bir Dünya haritası endikatrix gösterimleri ile verilmiştir.



Şekil 2. Mollweide projeksiyonunda Dünya

2.2. t Parametresinin Hesabı

Mollweide tarafından projeksiyon sunulduğunda bilişim teknolojisi olmadığından (12) eşitliğinden t

parametresinin hesaplanması kolay değildi. Bu nedenle belli aralıklarla enleme göre düzenlenmiş tablolar kullanılıyordu. Tablo 1'de 10° aralıklarla enlem t değerleri Şekil 3'te ise grafiği gösterilmiştir.

Şekil 3'te görülen eğri $t = t(\varphi)$ biçiminde bir fonksiyonla temsil edilebilirse (12) eşitliğindeki hesaplama güçlüğü aşılabılır. Eğrinin kolay hesaplanabilir polinomlar ile temsili yeterli doğruluğu sağlamaz. Kübik Spline gibi eğriler yeterli doğruluğa ulaşabilir, ancak hesaplamada kolaylık ve/veya hız sağlamaz. Bunun yerine (12) eşitliğinin Newton vb. bir yöntemle iteratif çözümü daha avantajlı olmaktadır.

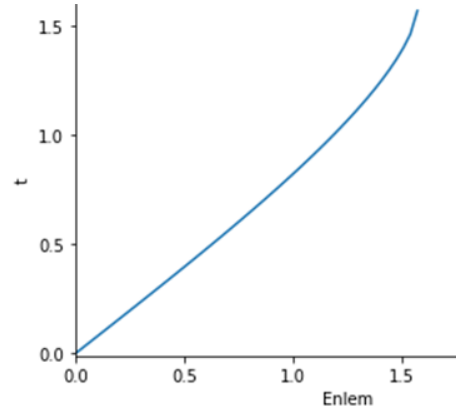
(12) eşitliğinde t bilinmeyen olmak üzere denklem kökü bulmak için sifıra eşit bir fonksiyon yazmak gerekir.

$$f(t) = 2t + \sin 2t - \pi \sin \varphi = 0 \quad (13)$$

Bu durumda problem, $f(t)$ 'yi sıfır yapacak t değerinin bulunmasıdır.

Tablo 1. $\varphi - t$ ilişkisi

φ°	t°
0.0	0.000000
10.0	7.863352
20.0	15.784190
30.0	23.826771
40.0	32.071200
50.0	40.628931
60.0	49.675004
70.0	59.531721
80.0	70.977834
90.0	90.000000



Şekil 3. Enlem- t grafiği (radyan biriminde)

2.2.1. Newton Yöntemi

İlk olarak yaklaşık bir başlangıç değerine gerek vardır. $t_0 = \varphi$ olarak seçilebilir. Hesaplama yinelemeli olarak,

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)} \quad (14)$$

$\frac{f(t_i)}{f'(t_i)}$ yeterince küçük oluncaya kadar devam ettirilir. Bilgisayarlarda reel sayıların temsiline kullanılan çift incelekte 14 basamak doğruluk söz konusu olduğundan olası en küçük değer pratik olarak 10^{-14} 'tür.

(14) için yineleme (iterasyon) sayısı çok fazla değildir. Çoğunlukla en çok 5 adımda yeterli doğrulukta çözüme ulaşılır.

Newton yönteminde birinci türeve gerek vardır. Bazı yazılım dillerinde bulunan hazır kütüphane fonksiyonları (Örneğin Python SciPy modülü fsolve fonksiyonu) türevi nümerik hesaplayarak çözüm sağlamakta, türevin kodlanmasını gerektirmemektedir.

(13) eşitliğinde t hem trigonometrik fonksiyon içinde hem de dışında bulunduğu için hesaplamalar radyan biriminde yapılmalıdır.

Newton yöntemi çok değişkenli olarak da (birden çok fonksiyonun ortak kökleri) uygulanabilir (Bildirici, 2023b; İpbüker, 2002; İpbüker ve Bildirici, 2005).

2.2.2. Halley Yöntemi

Bu yöntemde de bir başlangıç değerine gerek vardır. Yinelemeli hesaplama,

$$t_{i+1} = t_i - \frac{2f(t_i)f''(t_i)}{2(f'(t_i))^2 - f(t_i)f''(t_i)} \quad (15)$$

biçimde olup, ikinci türev gerektirmektedir. Bu nedenle de Newton yöntemine göre adım sayısı biraz daha kısalmaktadır.

2.3. Ters Projeksiyon

Ters projeksiyon, projeksiyon düzlemindeki bir noktanın coğrafi koordinatlarının bulunması problemidir. (6) eşitliklerinde önce t bulunur.

$$t = \arcsin\left(\frac{y}{\sqrt{2}}\right) \quad (16)$$

Buradan (12) ile,

$$\lambda = \frac{x\pi}{2\sqrt{2}\cos t} \quad \varphi = \arcsin\left(\frac{2t + \sin 2t}{\pi}\right) \quad (17)$$

elde edilir (Snyder, 1987).

Mollweide projeksiyonu için ters projeksiyon hesaplamaları basit olup yinelemeli yöntemler gerektirmez.

2.4. Deformasyonlar

Projeksiyon ortogonal olmayan bir coğrafi ağa sahip olduğundan deformasyonların hesaplanması için kısmi türevlere gerek vardır. Tissot Endikatrisinin yönünün belirlenmesinde de ağın ortogonal olmaması dikkate alınmalıdır (Bildirici, 2016; Bildirici, 2023b).

(11)'den,

$$\frac{dt}{d\varphi} = \frac{\pi \cos \varphi}{4 \cos^2 t}$$

yazabiliriz. (9)'da yerine koyarak kısmi türevleri düzenleyelim.

$$\frac{\partial x}{\partial \varphi} = -\frac{2\lambda \tan t \cos \varphi}{2\sqrt{2} \cos t} \quad \frac{\partial x}{\partial \lambda} = \frac{2\sqrt{2}}{\pi} \cos t$$

$$\frac{\partial y}{\partial \varphi} = \frac{\pi \cos \varphi}{2\sqrt{2} \cos t} \quad \frac{\partial y}{\partial \lambda} = 0$$

Projeksiyon düzleminde temel büyüklükler,

$$\begin{aligned} e &= \frac{\cos^2 \varphi}{8 \cos^2 t} (4\lambda^2 \tan^2 t + \pi^2) \\ f &= \frac{-2\lambda \tan t \cos \varphi}{\pi} \\ g &= \frac{8 \cos^2 t}{\pi^2} \end{aligned} \quad (19)$$

Kürede (birim) $E = 1, F = 0, G = \cos^2 \varphi$ olduğu göz önüne alınırsa,

$$ef - f^2 = EG - F^2 = \cos^2 \varphi$$

olduğu, dolayısı ile alan korunduğu görülmektedir.

Temel büyüklükler yardımıyla deformasyon ölçütü (K) ve ortalama deformasyon (H) belirlenir (Hoschek, 1984; Üstün ve Bildirici, 2021). Alan koruma nedeniyle $K=1$ olur.

$$\begin{aligned} 2H^2 &= \frac{Eg - 2Ff + Ge}{eg - f^2} \\ &= \frac{8 \cos^2 t}{\pi^2 \cos^2 \varphi} + \frac{(4\lambda^2 \tan^2 t + \pi^2)}{8 \cos^2 t} \end{aligned} \quad (20)$$

Ana deformasyon yönlerindeki deformasyon oranları ya da kısaca deformasyonlar (Hoschek, 1984; Üstün ve Bildirici, 2021) (maksimum ve minimum deformasyon):

$$a^2 = \frac{1}{H^2 - \sqrt{H^4 - K^2}} \quad b^2 = \frac{1}{H^2 + \sqrt{H^4 - K^2}} \quad (21)$$

Meridyen ve paraleller yönündeki deformasyonlar (Hoschek, 1984; Üstün ve Bildirici, 2021):

$$\begin{aligned} h &= \sqrt{\frac{e}{E}} = \frac{\cos t \sqrt{4\lambda^2 \tan^2 t + \pi^2}}{2\sqrt{2} \cos t} \\ k &= \sqrt{\frac{g}{G}} = \frac{2\sqrt{2} \cos t}{\pi \cos \varphi} \end{aligned} \quad (22)$$

Açı deformasyonu,

$$w = 2 \arcsin\left(\frac{a-b}{a+b}\right) \quad (23)$$

olur. Tissot endikatrisinin yönü (a ekseninin yataydan olan açısı),

$$\gamma = \text{sign}(f) \arcsin \sqrt{\frac{1 - \frac{a^2}{k^2}}{1 - \frac{a^2}{b^2}}} \quad (24)$$

Tablo 2. Deformasyon değerleri

φ	λ	x	y	a	b	h	k	w^o
0	0	0.000	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	30	0.471	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	60	0.943	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	90	1.414	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	120	1.886	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	150	2.357	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
0	180	2.828	0.000	1.1107	0.9003	1.1107	0.9003	12.0111
30	0	0.000	0.571	1.0515	0.9510	1.0515	0.9510	5.7558
30	30	0.431	0.571	1.0965	0.9120	1.0629	0.9510	10.5456
30	60	0.862	0.571	1.1759	0.8504	1.0962	0.9510	18.4875
30	90	1.294	0.571	1.2654	0.7903	1.1495	0.9510	26.7274
30	120	1.725	0.571	1.3617	0.7344	1.2203	0.9510	34.8264
30	150	2.156	0.571	1.4637	0.6832	1.3056	0.9510	42.6340
30	180	2.857	0.571	1.5708	0.6366	1.4030	0.9510	50.0728
60	0	0.000	1.078	1.1652	0.8582	0.8582	1.1652	17.4553
60	30	0.305	1.078	1.2536	0.7977	0.9220	1.1652	25.6826
60	60	0.610	1.078	1.4367	0.6960	1.0913	1.1652	40.6429
60	90	0.915	1.078	1.6593	0.6027	1.3262	1.1652	55.6972
60	120	1.220	1.078	1.9070	0.5244	1.5981	1.1652	69.3123
60	150	1.525	1.078	2.1730	0.4602	1.8910	1.1652	81.1549
60	180	1.830	1.078	2.4530	0.4677	2.1967	1.1652	91.2831

biçiminde bulunur (Bildirici, 2023b, s.152). Endikatrix yönü için Bildirici (2016) kaynağının incelenmesinde de yarar vardır.

Tablo 2'de 30° aralıklı grid noktalarında deformasyon değerleri gösterilmiştir. Tablo değerleri ile Şekil 2'de gösterilen endikatrixler (elipsler) karşılaştırıldığında projeksiyonun deformasyon davranışı daha iyi yorumlanabilir. Projeksiyon deformasyonlarının değerlendirilmesi için düzenli grid oluşturan noktalar kullanmak yerine alternatif yaklaşımlar da önerilmiştir (Yetgin ve ark., 2024; Canters, 2002).

3. Python Dili Uygulaması

Bu başlıkta (12) eşitliğinden t parametresinin bulunuşu ile ilgili Python dili *scipy.optimize* modülünde bulunan *newton*, *fsolve* ve *root* fonksiyonları tanıtılacaktır. Bu tür kök bulma fonksiyonları için kökü bulunacak fonksiyon ile opsiyonel olarak birinci ve ikinci türev fonksiyonların hazırlanması gereklidir. (13) eşitliği ile birinci ve ikinci türevleri için aşağıdaki fonksiyonlar kullanılabilir.

```
import math
from scipy.optimize import fsolve,newton,root

def tfonk(t,u):
    return 2*t[0]+math.sin(2*t[0])-
    math.pi*math.sin(u)

def tprime(t,u):
    return [2+2*math.cos(2*t[0])]

def tprime2(t,u):
    return [-4*math.sin(2*t[0])]
```

Kök bulma fonksiyonları çok değişkenli olarak tasarlandığı için t parametresi liste olarak parametrelerde yer almalıdır. Türevlerin geri dönüş değerlerinin de liste (ya da numpy.array) türünde olması gerekmektedir. u değişkeni enlem için kullanılmıştır.

3.1. fsolve Fonksiyonu

scipy.optimize modülünde yer alır. *fsolve*, MINPACK'in *hybrd* ve *hybrj* algoritmaları ile çalışır (Mor'e ve ark., 1980). Parametreleri, kökü bulunacak fonksiyon ile başlangıç değeri olup, opsiyonel olarak fonksiyon argümanları ve birinci türev fonksiyonu da verilebilir. Birinci türev verilmediğinde numerik türev kullanılır. Çok değişkenli tasarlandığı için birinci türev Jakobiyen matrisi biçimindedir. Tek değişkenli durumda tek elemanlı matristir. Matris yapıları liste ya da *numpy.array* türlerinden biri ile temsil edilebilir. Geri dönüş değeri *numpy.array* türündedir. Fonksiyonun tek kökü var ise tek elemanı doludur.

Yukarıda verilen fonksiyonlar ve başlangıç değerini enleme eşit alarak t değeri, aşağıdaki kodlar ile hesaplanabilir.

```
u=math.radians(50)
t=fsolve(tfonk,[u],args=(u,),fprime=tprime)
print(math.degrees(t[0]))
```

ya da

```
u=math.radians(50)
t=fsolve(tfonk,[u],args=(u,))
print(math.degrees(t[0]))
```

Birinci türevin kullanılmasının performansla belirgin bir etkisi yoktur.

Ayrıntılı bilgi:

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html#fsolve>

3.2. newton Fonksiyonu

scipy.optimize modülünde yer alan *newton* fonksiyonu yukarıda ayrıntısı verilmiş olan Newton ya da Halley yöntemlerine göre çalışır. Parametreleri *fsolve* ile benzerdir. Ek olarak *fprime2* parametresi ile ikinci türev fonksiyonu da kullanılabilir. Üç şekilde kullanılabilir.

- Birinci türev verilmezse numerik olarak belirlenip Newton yöntemi ile çalışır.
- Birinci türev verilirse Newton yöntemine göre çalışır.
- Birinci ve ikinci türevler verilirse Halley yöntemine göre çalışır.

Üç seçeneğe göre örnek kodlar aşağıdaki gibidir:

```
u=math.radians(50)
t=newton(tfonk, [u], args=(u,))
print(math.degrees(t[0]))
```

```
u=math.radians(50)
t=newton(tfonk, [u], args=(u,), fprime=tpime)
print(math.degrees(t[0]))
```

```
u=math.radians(50)
t=newton(tfonk, [u], args=(u,), fprime=tpime, fprime
2=tpime2)
print(math.degrees(t[0]))
```

Ayrıntılı bilgi:

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html#newton>

3.3. root Fonksiyonu

Bu fonksiyon 10 değişik yöntem ile çalışabilir. Varsayılan, modifiye edilmiş Powell yöntemi olan 'hybr' olarak adlandırılan yöntemdir (Mor'e ve ark., 1980). Parametreleri fsolve ile benzerdir. Ancak burada birinci ve/veya ikinci türev seçenekleri yoktur. Geri dönüş değeri ise özel bir nesne olup x niteliğinde fonksiyonun kökü/kökleri (NumPy array veri türünde) yer alır. Varsayılan yöntem ile çözüm için örnek kod:

```
u=math.radians(50)
t=root(tfonk, [u], args=(u,))
print(math.degrees(t.x[0]))
```

Ayrıntılı bilgi:

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html#scipy.optimize.root>

3.4. Fonksiyonların Performansı

Yukarıda ayrıntıları verilen yöntemlerin performansları hakkında bir değerlendirme yapmak için $[-\pi/2, \pi/2]$ aralığında rastgele 500 enlem değeri içeren bir veriden t parametreleri hesaplanmıştır. Program çalışma zamanı arka plan işlemleri nedeniyle değişkenlik göstereceğinden, hesaplama her yöntem için 100 kez tekrar edilip çalışma zamanlarının ortalaması alınmıştır. Sonuçlar Tablo 3'te görülmektedir. Tablodaki değerlerden Newton ve Halley yöntemlerinin daha yavaş olduğu, fsolve fonksiyonunun oldukça iyi sonuç verdiği ve 1. türeve gerek duymadığı anlaşılmaktadır. root fonksiyonunda hybr yöntemi fsolve ile özdeştir. root'daki bazı yöntemlerin ise buradaki probleme uygun olmadığı ve hata verdiği görülmüştür.

Çalışma zamanının işletim sistemi ve donanıma bağlı olduğu açıktır. Burada, yöntemlerin göreceli olarak birbirleri ile karşılaştırılması için ortalama çalışma zamanları belirlenmiştir. Verilen ortalama çalışma zamanlarının mutlak büyüklükler olduğu

düşünülmemelidir. Farklı donanım, Python yorumlayıcısı ve işlem sistemlerinde farklı çalışma zamanları oluşabilir.

3.5. Örnek Sınıf Kodu

fsolve fonksiyonu kullanarak projeksiyon, ters projeksiyon ve deformasyon hesaplarını yapmak üzere örnek sınıf kodu Ek A'da verilmiştir. Kodun devamında Tablo 2'deki değerleri hesaplayan bir kod da yer almaktadır.

Sınıf, aşağıdaki metotları kapsamaktadır:

Tablo 3. Fonksiyon ortalama çalışma zamanları

Fonksiyon	Çalışma Zamanı (sn)
newton	0.087
newton, 1. türev ile	0.065
halley (1. ve 2. türev ile)	0.074
fsolve	0.014
fsolve, 1. türev ile	0.013
root, hybr yöntemi ile	0.013
root, lm yöntemi ile	0.027

- geo2xy (enlem, boylam): Coğrafi koordinatlardan düzlem koordinatların hesabı
- xy2geo (x, y): Düzlem koordinatlardan coğrafi koordinatların hesabı
- partials (enlem, boylam): Kısmi türevler
- tissot (enlem, boylam): Deformasyon değerleri hesabı (a, b, h, k, w)
- u2t (enlem): t parametresinin hesabı (fsolve kullanılarak)
- efg: Kısmi türevler listesinden düzlemde temel büyüklüklerin hesabı
- ft: t parametresinin hesabı için gerekli fonksiyon

4. Sonuç ve Öneriler

Bu makalede Mollweide projeksiyonu kaynaklarda çok bilinmeyen yönleri ile ele alınmış, özellikle projeksiyon hesaplamalarındaki kök bulma problemi mercek altına alınmıştır. Mollweide, yeryüzünü 2:1 oranlı elips içinde gösteren projeksiyon ailesinin bir üyesidir. Bu makalede yeryüzünü bir elips içinde gösterme yaklaşımı da incelenmiştir.

Hesaplamalar için kullanılacak hazır Python modül fonksiyonları tanıtılmış ve karşılaştırılmıştır. Ele alınan probleme yazılımsal çözümler de sunulmuştur. Yazılımsal çözümler, enleme bağlı parametre kullanılan diğer projeksiyonlara da (Eckert IV, Eckert VI vb.) kolaylıkla uyarlanabilir.

Kök bulma genel bir optimizasyon problemi olup burada ele alınmayan ancak eldeki probleme uygun olabilecek yöntem ve yazılımsal çözümler şüphesiz mevcuttur. Ele alınan yöntem ve yazılımsal çözümler uygun sonuçlar verdiği için daha fazla alternatif arayışına girilmemiştir.

Ekler**A) Örnek Sınıf Kodu**

```

class mollweide:
    'Mollweide projeksiyonu sınıfı'
    def __init__(self,r=1):
        self.r=r
        self.cx=r**2**0.5/math.pi
        self.cy=r**2**0.5
    def geo2xy(self,u,v):
        'Coğrafi-düzlem koordinat hesabı, açı
        birimi radyan'
        if abs(abs(u)-math.pi/2)<=1.e-14:
            return 0,np.sign(u)*self.cy
            t=self.u2t(u)
            x=self.cx*v*math.cos(t)
            y=self.cy*v*math.sin(t)
            return x,y
    def xy2geo(self,x,y):
        'Düzlem-coğrafi koordinat hesabı, açı
        birimi radyan'
        t=math.asin(y/self.cy)
        v=x/(self.cx*math.cos(t))
        u=math.asin((2*t-math.sin(2*t))/math.pi)
        return u,v
    def partials(self,u,v):
        'Kısmi türevler, list biçiminde'
        t=self.u2t(u)
        d=[0.,0.,0.,0.]
        d[0]=-
        2**0.5*v*math.tan(t)*math.cos(u)/(2*math.cos(t))
        #dx/du
        d[1]=2**0.5*math.cos(t)/math.pi
        #dx/dv
        d[2]=math.pi*math.cos(u)/(2**0.5*math.cos(t))
        #dy/du, dy/dv=0
        return d
    def tissot(self,u,v):
        'Tissot Endikatrik elemanları, açı def.
        derece'
        t=self.u2t(u)
        h2=4*math.cos(t)**2/(math.cos(u)**2*math.pi**2)+\
        math.cos(u)**2*(4*v**2**math.tan(t)**2+math.pi**2)
        /(16*math.cos(t)**2)
        b=(h2+(h2**2-1)**0.5)**(-0.5)
        a=(h2-(h2**2-1)**0.5)**(-0.5)
        h=math.cos(u)*4*v**2**math.tan(t)**2+math.pi**2)*
        *0.5/(8**0.5*math.cos(t))
        k=8**0.5*math.cos(t)/(math.pi*math.cos(u))
        w=2*math.asin((a-b)/(a+b))
        return a,b,h,k,math.degrees(w)
    def u2t(self,u):
        'Enlem-t parametresi hesabı'
        return fsolve(self.ft,[u],args=(u,))[0]
    @staticmethod
    def efg(d):
        'Temel büyüklükler'
        e=d[0]**2+d[2]**2
        g=d[1]**2+d[3]**2
        f=d[0]*d[1]+d[2]*d[3]
        return e,f,g
    @staticmethod
    def ft(t,u):
        'f(t)=0 fonksiyonu, fsolve için
        kullanılıyor'
        return
        math.pi*math.sin(u)
#Sınıfın örneklenmesi
proj=mollweide(r=1)
#Koordinat ve deformasyonlar
for u in range(0,90,30):
    for v in range(0,185,30):
        print(f"{u:>3d} {v:>3d}",end=' ',)
x,y=proj.geo2xy(math.radians(u),math.radians(v))
print(f"{x:>8.3f}{y:>8.3f}",end=' ')
d=proj.tissot(math.radians(u),math.radians(v))
txt="{:>8.4f}"*len(d)
print(txt.format(*d))

```

B) Notasyon

φ, λ Enlem, boylam
 x, y Projeksiyon düzlemi koordinatları (x sağa, y yukarı)
 h Meridyen yönünde uzunluk deformasyonu
 k Paralel yönünde uzunluk deformasyonu
 a, b Maksimum ve minimum uzunluk deformasyonları
 γ Tissot endikatriksi büyük yarıçapının yataydan olan açısı
 E, F, G Referans yüzeyinde temel büyüklükler
 e, f, g Projeksiyon düzleminde temel büyüklükler
 w Açı deformasyonu
 K Deformasyon ölçütü
 H Ortalama deformasyon
 R Küre yarıçapı

Araştırmacıların katkı oranı

İbrahim Öztuğ Bildirici: Literatür taraması, Bulgular, Tartışma, Sonuçlar, Makale yazımı, Düzenleme

Çatışma Beyanı

Herhangi bir çıkar çatışması bulunmamaktadır.

Kaynakça

- Bildirici, İ. Ö. (2016). Ortogonal olmayan coğrafi ağı harita projeksiyonlarında Tissot endikatrik elemanlarının belirlenmesi. *Harita Dergisi*, 156: 13-22.
- Bildirici, İ. Ö. (2017). An iterative approach for inverse transformation of map projections. *Cartography and Geographic Informaiton Science*, 44(5):463-471.
- Bildirici, İ. Ö. (2019). Harita projeksiyonları ve nümerik analiz. *Geomatik*, 4(2):160-169.
- Bildirici, İ. Ö. (2023a). Alan koruyan projeksiyonlar her zaman alan korur mu? *Geomatik*, 8(3):306-311.
- Bildirici, İ. Ö. (2023b). Kartografya: Harita tasarımı ve kullanımı için gerekli bilim, sanat ve teknik. Atlas Akademi Yayınevi, Konya, 3. Baskı.
- Bugayevskiy, L. M. & Snyder, J. (1995). *Map projections: A reference manual*. CRC Press, Philadelphia.
- Canter, F. (2002). *Small-Scale Map Projection Design*. Taylor and Francis, London.
- Doğan, Y. & Yakar, M. (2018). Gis and three-dimensional modeling for cultural heritages. *International Journal of Engineering and Geosciences*, 3(2):50-55.
- Hoschek, J. (1984). *Mathematische Grundlagen der Kartographie*. Bibliographisches Institut, Mannheim.
- Ipbuker, C. (2002). An inverse solution to the winkel tripel projection using partial derivatives. *Cartography and Geographic Information Science*, 29(1):37-42.
- Ipbuker, C. & Bildirici, İ. Ö. (2005). Computer program for the inverse transformation of the winkel projection. *Journal of Surveying Engineering*, 131(4):125-129.
- Jonuzi, E., Alkan, T., Durduran, S. S., Selvi, H. Z. (2024). Using gis-supported mcda method for appropriate site selection of parking lots: The case study of the city of tetovo, north macedonia. *International Journal of Engineering and Geosciences*, 9(1):86-98.

- Mor'e, J. J., Garbow, B. S., Hillstrom, K. E. (1980). User guide for minpack-1. Teknik rapor:, CM-P00068642.
- Şentürk, E. ve Erener, A. (2017). Determination of temporary shelter areas in natural disasters by gis: A case study, g'olc'uk/turkey. International Journal of Engineering and Geosciences, 2(3):84–90.
- Snyder, J. P. (1987). Map projections-A working manual, volume 1395. US Government Printing Office, Washington, D.C.
- Snyder, J. P. (1993). Flattening the Earth. University of Chicago Press, Chicago.
- Üstün, A. & Bildirici, İ. Ö. (2021). Harita Mühendisliğinde Diferansiyel Geometri. <https://www.iobildirici.com/docs/hmdg.pdf>, 01.11.2024.
- Wagner, K. (1949). Kartographische Netzentwürfe. Bibliographisches Institut Leipzig.
- Yakar, M., & Dogan, Y. (2019). 3D Reconstruction of Residential Areas with SfM Photogrammetry. In Advances in Remote Sensing and Geo Informatics Applications: Proceedings of the 1st Springer Conference of the Arabian Journal of Geosciences (CAJG-1), Tunisia 2018 (pp. 73-75). Springer International Publishing.
- Yetgin, M., Kırtılođlu, O. S., Koçak, G. (2024). Fibonacci kafesi ile harita projeksiyon deformasyonlarının deđerlendirilmesi. Geomatik, 10(1):47–53.



© Author(s) 2024. This work is distributed under <https://creativecommons.org/licenses/by-sa/4.0/>