Bölüm 8

SageMath ile Uygulamalar

Bu bölümde farklı alanlarda SageMath uygulamaları verilmiştir. Alt bölümler birbirinden bağımsızdır. İlginizi çekmeyen ya da ön bilgi gerektirdiğini düşündüğünüz uygulamaları atlayabilirsiniz. Bu bölümün amacı zor problemler çözmek değil, çeşitli problemleri çözmemize yardımcı olacak araçları basit problemlerle tanıtmaktır.

8.1 Fermat Asalları

n negatif olmayan bir tam sayı olmak üzere,

$$F_n = 2^{2^n} + 1$$

şeklindeki sayıları düşünelim. Pierre de Fermat 1650'de yukarıdaki F_n formülünün sadece asal sayı veren bir formül olduğu sanısını ortaya atmış. Seksen yıldan fazla zaman aksini iddia eden olmamış; ta ki İsviçreli matematikçi Leonhard Euler 1732'de bu formülün $n = 5$ için asal olmadığını ve aşağıdaki gibi çarpanlarına ayrılabileceğini gösterene kadar:

$$2^{2^5} + 1 = 641 \times 6700417$$

Fermat bu sanısını bugün ortaya atsaydı SageMath bir saniyeden daha kısa zamanda Euler'in bulduğunu bulurdu:

```
is_prime(2^2^5+1)
```

```
False
```

```
factor(2^2^5+1)
```

```
641 * 6700417
```

F_n formundaki bir sayıya *Fermat sayısı*, eğer sayı asal ise *Fermat asalı* denir. Günümüzde bilinen sadece 5 tane Fermat asalı var; bunlar da $n = 0, 1, 2, 3, 4$ için elde edilenler. SageMath yardımıyla $n = 0, 1, 2, \ldots, 14, 15$ durumlarına bakalım:

```
[ (i,is_prime(2^2^i+1)) for i in [0..15] ]
```

```
[(0, True),
 (1, True),
 (2, True),
 (3, True),
 (4, True),
 (5, False),
 (6, False),
 (7, False),
 (8, False),
 (9, False),
 (10, False),
 (11, False),
 (12, False),
 (13, False),
 (14, False),
 (15, False)]
```

Alıştırma 8.1.1 *p asal sayı olmak üzere $M_p = 2^p - 1$ formundaki asal sayılara Mersenne asalı denir.*

a. İlk 100 tane p asal sayısı için M_p sayısının Mersenne asalı olup olmadığını bulun.

b. En küçük 15 Mersenne asalını listeleyin.

Not 8.1.1 Bu satırlar yazılıyorken (Eylül 2022), son bulunanı Aralık 2018'de olmak üzere sadece 51 tane Mersenne asalı bilinmekteydi (`https://www.mersenne.org/primes/`).

8.2 Bir Denklem

Aşağıdaki denklemin 1'den büyük bir kökünün olmadığını iddia ediyoruz:

$$x^7 - 2x^5 + 10x^2 - 1 = 0 \tag{8.1}$$

Bunu göstermek için yukarıdaki denklemde $x = y + 1$ dönüşümü yapalım. Yani x gördüğümüz yere $y + 1$ yazalım. Sonra da gerekli genişletmeleri yapıp ifadeyi sadeleştirelim. Tabii ki SageMath ile:

```
(x^7-2*x^5+10*x^2-1).subs(x=y+1).expand()
```

```
y^7 + 7*y^6 + 19*y^5 + 25*y^4 + 15*y^3 + 11*y^2 + 17*y + 8
```

İşlemimiz tamam. Aşağıdaki denklemi elde ettik:

$$y^7 + 7y^6 + 19y^5 + 25y^4 + 15y^3 + 11y^2 + 17y + 8 = 0 \tag{8.2}$$

x ve y arasındaki ilişki $y = x - 1$ şeklinde olduğundan dolayı, 8.1 numaralı denklemin 1'den büyük kökünün olmadığını söylemek, 8.2 numaralı denklemin 0'dan büyük (yani pozitif) kökünün olmadığını söylemekle aynı şey. 8.2 denkleminin bütün katsayıları pozitif olduğundan pozitif bir y değeri, 8.2 denkleminin sol tarafını pozitif yapacaktır; yani hiçbir pozitif y değeri denklemi sağlamayacaktır. Böylece 1'den büyük hiçbir x değeri de 8.1 denklemini sağlamayacaktır.

Not 8.2.1 Bu küçük problem Howard Eves'in *Fundamentals of Modern Elementary Geometry* [5] adlı kitabında, dönüşümlerin işleri kolaylaştırdığı örneklerden biri olarak veriliyor.

Alıştırma 8.2.1 *$f : \mathbb{R} \to \mathbb{R}$ bir fonksiyon olmak üzere, $f(x) = 0$ denkleminin kökleri $y = f(x)$ fonksiyonunun grafiğinin x-ekseniyle kesiştiği noktalardır. 8.1 numaralı denklemin 1'den büyük kökünün olmadığı iddiasını grafikle destekleyin. Ayrıca, söz konusu dönüşümün grafiksel karşılığını yorumlayın.*

8.3 Fahrenhayttan Santigrata Çeviri

Sıcaklık birimleri olan *fahrenhayt* (°F) ve *santigrat* (°C) arasında aşağıdaki gibi bir ilişki var:

$$C = \frac{5}{9}(F - 32)$$

Şimdi bizim için bu işlemi yapacak küçük bir SageMath programı yazıp, 80 °F'ın kaç derece olduğunu hesaplayalım:

```
F=80
C=5/9*(F-32)
round(C,2)
```

```
26.67
```

Buradaki `round(C,2)` komutunun noktadan sonra 2 basamak kullanarak yuvarlama yaptığını 2.6 numaralı alt bölümden biliyoruz. Farklı bir fahrenhayt değeri için benzer şekilde sonucu bulma şansımız olsa da, bir fonksiyon kullanmak daha kullanışlı olacaktır. Aşağıda `FdenCye` adında bir fonksiyon oluşturuyoruz:

```
def FdenCye(F):
    C=5/9*(F-32)
    print(round(C,2))
```

Fonksiyon tamam; hemen kullanalım:

```
FdenCye(80)
```

```
26.67
```

Şimdi biraz daha ileri gidip etkileşimli bir araç yaratalım. Böylece girdi kutusuna girilen bir fahrenhayt ifadesini hemen santigrat ifadesine çevirmiş olacağız.

```
@interact
def _( F=input_box(label="Fahrenhayt:",default=80,width=15) ):
    C=5/9*(F-32)
    print(round(C,2))
```

Alıştırma 8.3.1 *Santigrattan fahrenhayta çeviri yapan,* `CdenFye` *adlı bir fonksiyon oluşturun ve bu fonksiyonu interaktif bir araç oluşturmak için kullanın.*

Alıştırma 8.3.2 *Kullanıcının buton kullanarak C'de F'ye ve F'den C'ye seçeneklerinden birini seçerek çeviri yapacağı bir interaktif araç oluşturun. 6.6 numaralı alt bölüm buton oluşturmak için yardımcı olabilir.*

Not 8.3.1 Bu alt bölümdekine benzer bir şekilde uzunluk, alan, hacim ve ağırlık gibi birimler için çeviri aracı oluşturulabilirsiniz.

Alıştırma 8.3.3 (Sayısal Loto) *6/49 biçimindeki loto, 49 sayıdan rastgele seçilecek 6 sayıyı tutturmaya dayalı bir oyun. Daha genel olarak, r/n biçimindeki loto da n tane sayıdan seçilecek r tane sayıyı tutturmaya dayalı.*

r tane sayının tamamını tutturma olasılığı $1/C(n,r)$. Sadece k tanesini tutturma olasılığı da

$$\frac{C(r,k)\,C(n-r,r-k)}{C(n,r)}.$$

Girilen n, r ve k doğal sayıları için n sayıdan $r \leq n$ tanesini seçerek sadece $k \leq r$ tanesini tutturma olasılığı nümerik olarak veren bir fonksiyon oluşturun.

8.4 Bir Nümerik Çözüm Örneği: $x^2 = 2^x$

Bu alt bölümde matematik yazılımlarının çok yaygın kullanıldığı ve gerçekten de iyi iş çıkardıkları bir uygulama olan denkem çözümünü basit bir örnek ile inceleyeceğiz. Bilindiği üzere bazı denklemlerin bazı çözümlerini ancak nümerik olarak bulabiliriz. Örneğin

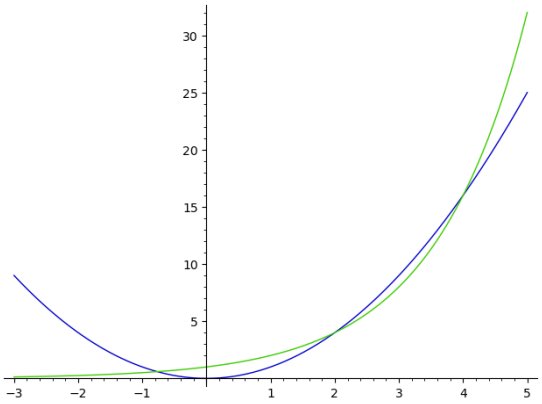
$$x^2 = 2^x$$

böyle bir denkem. Bu denklemin bütün reel sayı çözümlerini bulmaya çalışalım. 2 ve 4 değerleri, verilen denklemi sağlar ve bu değerlerin denklemin birer çözümü oldukları açıktır. Acaba başka bir çözüm var mı? Bunu görmek için denklemin iki tarafındaki

fonksiyonların grafiklerini aynı koordinat sisteminde SageMath'e çizdirip ortak bir x değerinde aynı değeri alıp almadıklarına bakalım. Aslında bunun anlamı grafiklerin kesişip kesişmedikleridir. Her bir kesişim noktasının apsisi, verilen denklem için bir çözümdür.

Aşağıdaki komut ile grafikleri çizdiriyoruz:

```
plot( [x^2 , 2^x] , (x,-3,5) )
```



Yukarıdaki komut SageMath'e $y = x^2$ ve $y = 2^x$ fonksiyonlarının grafiklerini $x = -3$'ten $x = 5$'e kadar olan kısmını çizdiriyor. Üç kesişim noktası görüyoruz. Bunlardan iki tanesi, tahmin ettiğimiz gibi $x = 2$ ve $x = 4$ değerleridir. Üçüncü kesişim noktasından anladığımıza göre bu çözüm negatif bir sayıdır.

4.5 numaralı alt bölümden bildiğimiz üzere kökü bulmak için `find_root` komutunu kullanabiliriz. Bir de kökün, içinde bulunduğu tahmini bir aralık girmemiz gerekiyor. Bu konuda grafikten destek alırsak $(-1, 0)$ veya $(-2, 0)$ aralıklarından biri uygun olacaktır. $(-2, 0)$ aralığını kullanalım.

```
find_root( 2^x == x^2, -2, 0 )
```

```
-0.7666646959621225
```

Böylece $x^2 = 2^x$ denkleminin, ikisi tam sayı olmak üzere üç tane reel sayı çözümü olduğunu görüyoruz. Tam sayı olmayan çözümün de yaklaşık değeri hemen yukarıda.

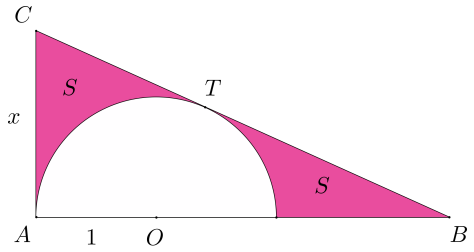
Not 8.4.1 Grafiklerle ilgili renk, çizgi stili, çizgi kalınlığı vb. gibi pek çok seçeneği 3 numaralı bölümde bulabilrsiniz.

Alıştırma 8.4.1 *$3^x = x^3$ denkleminin bütün reel sayı çözümlerinin mümkünse tam sonuçlarını, değilse yaklaşık değerlerini bulun.*

Alıştırma 8.4.2 *$x > 0$ olmak üzere $e^x = x^e$ denkleminin $x = e$ çözümünden başka bir çözümü var mı? Bu denklemi SageMath ile inceleyin.*

Alıştırma 8.4.3 *$f(x) = \sin x$ fonksiyonunun grafiğini 1 birim sağa kaydırıp $g(x)$ fonksiyonunun grafiğini elde ediyoruz. $y = f(x)$ ve $y = g(x)$ fonksiyonlarının grafiklerinin tüm kesim noktalarını bulun.*

Alıştırma 8.4.4 *Şekilde ABC dik üçgeni ve üçgenin iki kenarına A ve T noktalarında teğet olan O merkezli 1 birim yarıçaplı yarım çember görülmektedir. Taralı bölgelerin alanları eşit olduğuna göre $|AC| = x$ kaçtır?*



Alıştırma 8.4.5 *$x \geq 1$ olmak üzere aşağıdaki ifadenin alabileceği en büyük değeri bulun:*

$$(x-1)^{1/x}$$

Alıştırma 8.4.6 (En Büyük Eğim)

$$y = \frac{\sin x}{x}$$

fonksiyonunun, eğimi en büyük olan teğet doğrusunun denklemini bulun. Verilen fonksiyonun ve söz konusu teğetin grafiğini aynı koordinat düzleminde çizdirin.

8.5 Viviani Eğrisi

Viviani eğrisi, r pozitif bir sayı olmak üzere

$$x^2 + y^2 + z^2 = r^2$$

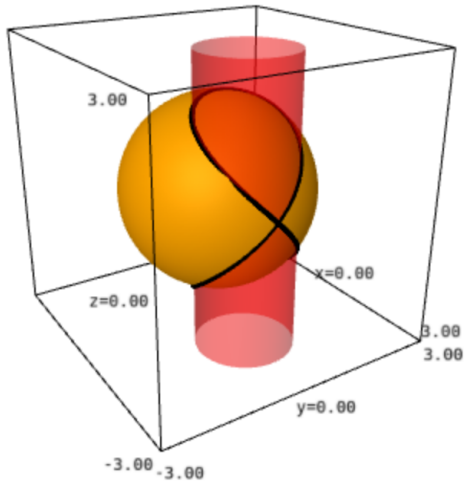
denklemiyle verilen küre yüzeyiyle

$$x^2 + y^2 = rx$$

silindir yüzeyinin kesişimiyle oluşan bir eğridir. Bu alt bölümde söz konusu küre ve silindirle birlikte, onların kesişimleri olan Viviani eğrisini aynı koordinat sisteminde çizdireceğiz. Özel olarak $r = 2$ alalım. Siz farklı bir pozitif değer seçebilirsiniz.

Komutlar aşağıda. Grafiği fare kullanarak çevirip inceleyiniz. Üç boyutlu grafikler için detaylar 5.13 numaralı alt bölümde.

```
var("x y z t")
r=2
kure=implicit_plot3d( x^2+y^2+z^2==r^2, (x,-3,3), (y,-3,3) , (z,-3,3),
    color="orange", opacity=1 , plot_points=150 )
silindir=implicit_plot3d( x^2+y^2==r*x, (x,-3,3), (y,-3,3) , (z,-3,3) ,
    color="red" , opacity=.5 , plot_points=150 )
viviani= parametric_plot3d(( r*cos(t)*cos(t), r*cos(t)*sin(t), r*sin(t)
    ), (t,-pi,pi), color="black", thickness=4, plot_points=1000)
show(kure+silindir+viviani)
```



Yukarıda önce sırasıyla küre ve silindiri `implicit_plot3d` komutuyla çizdiriyoruz. Sonra da Viviani eğrisini bu yüzeylerden bağımsız bir şeklide parametrik olarak `parametric_plot3d` ile oluşturuyoruz. Son olarak da bu üç grafiği aynı koordinat sisteminde gösteriyoruz. Beklediğimiz üzere eğri tam da yüzeylerin kesişiminden geçiyor.

Alıştırma 8.5.1 (Simit Kesitleri) *$R = 2$ ve $r = 1$ olmak üzere, üç boyutlu uzayda*

$$(x^2 + y^2 + z^2 + R^2 - r^2)^2 = 4R^2(x^2 + y^2)$$

simitini (torus) ve $y = 2$ düzlemini ele alalım.

a. *Verilen simit ve düzlemin grafiklerini aynı koordinat sisteminde çizdirin.*

b. *Grafiklerin kesim noktaları bir Bernoulli lemniskatı verecektir. Bu arakesiti de aynı koordinat sisteminde gösterin.*

c. *Farklı R ve r değerleri için bu durumu grafiklerle inceleyin.*

8.6 Vücut Kitle İndeksi

Quetelet İndeksi ya da daha yaygın deyimiyle *Vücut Kitle İndeksi* (*VKİ*) yetişkin bir kişinin ağırlığının boyuna göre normal olup olmadığını belirlemekte kullanılan bir parametredir. Bunun için doku kütlesi (kas, yağ ve kemik) ve boy temel alınır ve bu parametre vücut ağırlığının (kilogram), boy uzunluğunun (metre) karesine bölünmesi ile hesaplanır. Yani m ağırlık, h boy olmak üzere *VKİ* aşağıdaki gibi tanımlanır:

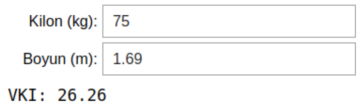
$$VKİ = \frac{m}{h^2}$$

VKİ, insanları kabaca zayıf, normal kilolu, fazla kilolu ve obez kategorilerine ayırmak için evrensel olarak kullanılmaktadır. Yaygın olarak kullanılan aralıklar aşağıdaki tabloda verilmiştir:

Kategori	Aralık
Zayıf	$VKİ < 18.50$
Normal Kilolu	$18.50 \leq VKİ < 25$
Fazla Kilolu	$25 \leq VKİ < 30$
Obez	$VKİ \geq 30$

Kullanıcının, kilosunu ve boyunu girip *VKİ* değerini elde edeceği interaktif bir araç yaratalım.

```
@interact
def _(
    m=input_box(label="Kilon (kg):", default=75),
    h=input_box(label="Boyun (m):", default=round(1.69,2) )
     ):
    print( "VKI:", round(m/h^2,2) )
```



Görüldüğü üzere kilo ve boy bilgileri, `input_box` (girdi kutucuğu) komutu kullanılarak girdi olarak isteniyor ve bu girdiler sırasıyla m ve h değişkenlerine atanıyor. Daha sonra *VKİ* formülününden elde edilen değer `print` komutuyla ekrana yazdırılıyor. Bunu yaparken metin ve değişkenleri virgül ile ayırdığımıza dikkat edin. Daha önce de yaptığımız gibi `round` komutuyla noktadan sonra iki basamak görmek istediğimizi belirttik. Varsayılan değerleri de kilo ve boy için sırasıyla 75 ve 1.69 olarak aldık.

Koşullu Önermeler ile Kategori Bilgisi

Oluşturduğumuz interaktif araç bize *VKİ* için sadece bir sayı değeri veriyor, biz de tablodan bakarak kategoriyi buluyoruz. İstersek bu kodları biraz geliştirip programın ilgili kategoriyi vermesini sağlayabiliriz. Aşağıdaki programı inceleyin:

```
@interact
def _(
    m=input_box(label="Kilon (kg):", default=75),
    h=input_box(label="Boyun (m):", default=round(1.69,2) )
     ):
    vki = m/h^2
    if vki < 18.5:
        kategori="Zayıf"
    if 18.5 <= vki < 25:
        kategori="Normal"
    if 25 <= vki < 30:
        kategori="Fazla Kilolu"
    if vki >= 30:
        kategori="Obez"
    print( "VKI:", round(vki,2) )
    print( "Kategori:", kategori )
```

Yukarıdaki programda bu problem için `if-elif-else` yapısının daha uygun olacağını tahmin etmiş olabilirsiniz. Aşağıdaki program tam da o yapıyı kullanıyor ve tamamıyla aynı işlevi görüyor. Programı çalıştırıp yorumlamayı size bırakıyorum. Koşullu önermeler için ilgili bölüm: 6.3.

```
@interact
def _(
    m=input_box(label="Kilon (kg):", default=75),
    h=input_box(label="Boyun (m):", default=round(1.69,2) )
     ):
    vki = m/h^2
    if vki < 18.5:
        kategori="Zayıf"
    elif vki < 25:
        kategori="Normal"
    elif vki < 30:
        kategori="Fazla Kilolu"
    else:
        kategori="Obez"
    print( "VKI:", round(vki,2) )
    print( "Kategori:", kategori )
```

Kategori Sınırları ve Seviye Eğrileri

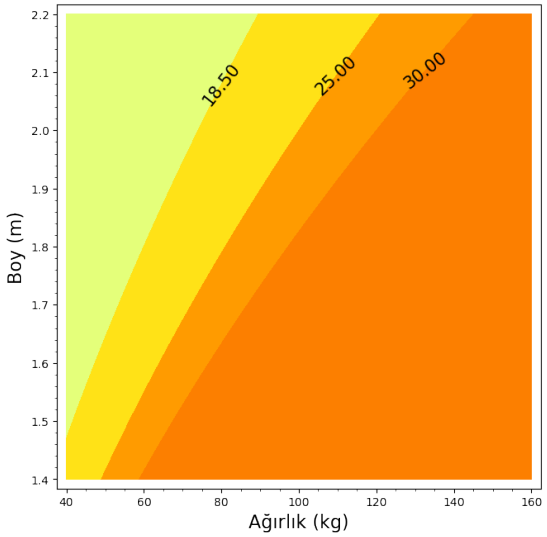
Şimdi de *VKİ* foksiyonuna biraz geometrik açıdan bakalım. Örneğin Ağırlık-Boy düzleminde hangi noktalar *VKİ* değerini 18.5 yapar? Bu noktaların kümesi, "zayıf" ile "normal kilo" kategorilerinin sınırını oluşturur. Bu bir eğri, hatta tam olarak $\frac{m}{h^2} = 18.5$ ya da daha açık bir ifadeyle $m = 18.5h^2$ parabolü olacaktır. Benzer şekilde 25 ve 30 değerleri için sınırları belirleyen eğriler de bulunabilir ve hepsi aynı düzlemde çizilebilir. Bu şekilde düzlemde üç tane parabol yayı ve dört farklı kategori için birer bölge oluşturabiliriz.

Ancak şu yöntem daha pratik olabilir: $VK\dot{I} = \frac{m}{h^2}$ iki değişkenli bir fonksiyon ve üç boyutlu uzayda bir yüzey ifade eder. Bu yüzeyin de C bir sabit olmak üzere,

$$\frac{m}{h^2} = C$$

biçiminde seviye eğrileri vardır. Örneğin $\frac{m}{h^2} = 18.5$ seviye eğrilerinden biridir. Yani daha önce incelediğimiz `contour_plot` komutuyla çok daha pratik bir şekilde tüm sınırları elde edebiliriz. Aşağıda `contour_plot` komutunu girdikten sonra pek çok seçenek giriyoruz. Bu sebeple kalabalık görünüyor. 5.7 numaralı bölümdeki seçeneklere bakılabilir.

```
var("m h")
contour_plot( m/h^2 , (m,40,160), (h,1.4,2.2), contours=[ 18.5, 25 ,
    30], cmap="Wistia", label_fontsize=15, linestyles='-',
    linewidths=0, figsize=10, label_inline=True, fill=True,
    aspect_ratio=150, axes_labels=[ "Ağırlık (kg)","Boy (m)" ],
    labels=True, label_colors='black' )
```



Yukarıdaki kodlar sınırları veren üç seviye eğrisi ve oluşturdukları bölgeleri çizdiriyor. Kilo ve boy için uygun aralıklar, bu aralıkların anlaşılır sonuç vermesi için oranlamalar (`aspect_ratio` komutu) ve biraz da makyaj ile istediğimiz sonucu elde ediyoruz. Deneyip siz de görün.

Hangi değerler için kontur çizdireceğimizi `contours` komutu ile giriyoruz. Seçeneklerde `True`'ları `False`'a çevirerek keşifler yapın. `cmaps` komutu ile renk grubu belirliyoruz; bu eğlenceli bir komut. Eksen etiketleme `axes_labels` ile yapılıyor. Detaylar 5.7 numaralı alt bölümde.

Son olarak kontur grafiği, *VKİ* ve kategoriyi aynı çıktıda veren bir araç oluşturalım. Yani önceki kodları birleştirelim. Ayrıca girilen *m* ve *h* bilgilerini aynı grafikte bir nokta olarak gösterelim. Aşağıdaki programı çalıştırıp görün.

```
@interact
def _(
    m=input_box(label="Kilon (kg):", default=75),
    h=input_box(label="Boyun (m):", default=round(1.69,2) )
     ):
    vki = m/h^2
    if vki < 18.5:
        kategori="Zayif"
    elif vki < 25:
        kategori="Normal"
    elif vki < 30:
        kategori="Fazla Kilolu"
    else:
        kategori="Obez"
    var("mm hh")
    cp = contour_plot( mm/hh^2 , (mm,40,160),(hh,1.4,2.2), contours=[
         18.5, 25 , 30],
         label_fontsize=15,linestyles='-',linewidths=0,figsize=10,
         cmap="Wistia",
         label_inline=True,fill=True,aspect_ratio=150,colorbar=False,
         axes_labels=[ "Ağırlık (kg)","Boy (m)" ], labels=True,
         label_colors='black' )
    pp=point( (m,h),pointsize=50 , zorder=5, color="black" )
    show(cp+pp)
    print( "VKI:", round(vki,2) )
    print( "Kategori:", kategori )
```

Alıştırma 8.6.1 *Boy ile kıyaslayınca, kilo yetişkin biri için daha müdahale edilebilir bir değişken. Yani uygun aralıkta olmak için boyumuzu değil de kilomuzu değiştirmeyi mantıklı buluruz. Şöyle bir araç oluşturun: Kullanıcı sadece boyunu girsin; boyuna bağlı olarak kilo sınır değerlerini veya aralıklarını çıktı olarak elde etsin.*

Alıştırma 8.6.2 *Yukarıda çizmiş olduğumuz kontur grafik dört farklı bölgeye ayrılmış durumda. Ancak hangi bölgenin hangi gruba ait olduğu grafikte görünmüyor. Bu bölgelere ilgili metin etiketlerini yerleştirin. Detaylar 5.10 numaralı alt bölümde.*

Alıştırma 8.6.3 (Cobb-Douglas Fonksiyonu) *Cobb-Douglas fonksiyonu ekonomide sıkça kullanılan bir üretim fonksiyonudur ve aşağıdaki gibi tanımlanır:*

$$Q = AK^{\alpha}L^{\beta}$$

Burada K ve L girdi faktörleri (genellikle, sırasıyla sermaye ve işgücü), Q ise bu girdilere bağlı olarak elde edilen üretim seviyesidir. α ve β parametreleri üretimin sırasıyla sermaye ve iş gücüne göre elastikliği olarak tanımlanır. A ise sabittir.

Kullanıcının α, β ve A değerlerini girip, K-L koordinat sisteminde bazı seviye eğrilerini görüp üretim seviyesi hakkında bilgi edineceği interaktif bir araç yaratın.

8.7 Brahmagupta Formülü

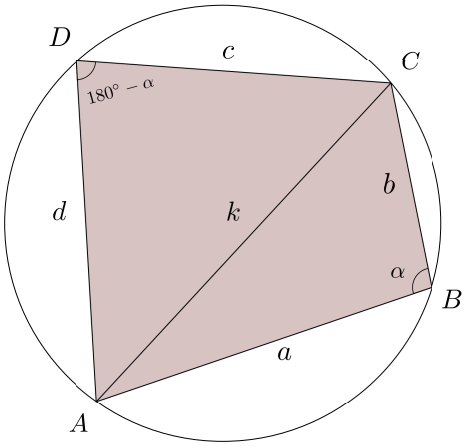
Brahmagupta Formülü kenar uzunlukları verilen bir kirişler dörtgeninin alanını veren formüldür. Diyelim ki kenar uzunlukları a, b, c, d olsun. s sayısı kirişler dörtgeninin çevre uzunluğunun yarısı, yani

$$s = \frac{a+b+c+d}{2}$$

olmak üzere, dörtgenin alanını aşağıdaki formülle bulabiliriz:

$$A = \sqrt{(s-a)(s-b)(s-c)(s-d)}$$

Brahmagupta Formülü'nün çeşitli ispatları var. En popüler olanı trigonometrik ispattır. Bu ispat uzun sadeleştirmeler ve işlemler içerdiğinden, sembolik hesaplamalar için SageMath'in kullanılabileceği güzel bir örnek.



Şekilde kenarları a, b, c, d olan bir kirişler dörtgeni görüyoruz. Amacımız, bu dörtgenin alanını a, b, c, d cinsinden ifade etmek ve bu ifadenin yukarıda verilen A'ya özdeş olduğunu göstermek. Dörtgenin alanına S diyelim. $[AC]$ köşegeninin uzunluğu k,

ABC açısı α derece olsun; böylece ADC açısı $(180° - \alpha)$ olacaktır. Dörtgenin alanı ABC ve ADC üçgenlerinin alanlarının toplamıdır. Üçgenlerin alanları için Sinüs Alan Teoremi'ni kullanırsak:

$$\begin{aligned} S &= A(ABC) + A(ADC) \\ &= \frac{1}{2}ab\sin\alpha + \frac{1}{2}cd\sin\left(180° - \alpha\right) \end{aligned}$$

Buradan, aşağıdaki sonuç elde edilir:

$$S = \frac{1}{2}(ab + cd)\sin\alpha \tag{8.3}$$

S'yi bulduk, ama henüz tam olarak kenar uzunluklarıyla ifade edilmiş değil. α açısından kurtulmamız lazım. Bunu Kosinüs Teoremi yardımıyla yapıyoruz. ABC ve ADC üçgenlerinde, B ve D açılarıyla Kosinüs Teoremi'ni kullanıp k'yı elimine edersek:

$$\begin{aligned} a^2 + b^2 - 2ab\cos\alpha &= c^2 + d^2 - 2cd\cos\left(180° - \alpha\right) \\ a^2 + b^2 - c^2 - d^2 &= 2(ab + cd)\cos\alpha \end{aligned}$$

Her iki tarafı 4'e bölelim:

$$\frac{1}{4}\left(a^2 + b^2 - c^2 - d^2\right) = \frac{1}{2}(ab + cd)\cos\alpha \tag{8.4}$$

Şimdi de (8.3) ve (8.4) denklemlerinin her iki tarafının karelerini alarak taraf tarafa toplayıp S^2'yi çekersek aşağıdaki ifadeyi elde ederiz:

$$S^2 = -\frac{1}{16}\left(a^2 + b^2 - c^2 - d^2\right)^2 + \frac{1}{4}(ab + cd)^2 \tag{8.5}$$

Amacımız S'nın A'ya özdeş olduğunu göstermek. İkisi de pozitif sayılar olduğundan S^2'nin A^2'ye eşit olduğunu göstermek yeterli olacaktır.

Formüldeki s ifadesini kenarlar cinsinden yazarsak A^2 aşağıdaki gibi olur:

$$A^2 = -\frac{1}{16}(a + b + c - d)(a + b - c + d)(a - b + c + d)(a - b - c - d) \tag{8.6}$$

Bu iki karmaşık ifadenin, yani (8.5) ve (8.6) denklemlerinin sağ taraflarının birbirine özdeş olduğunu göstermek istiyoruz. Bunu SageMath yapacak. Aşağıdaki kodlarda ifadelerin birine `alan1` diğerine de `alan2` diyoruz. `bool` komutu ile bu iki ifadenin özdeş olup olmadıklarını sorguluyoruz; ve `True` yanıtını alıyoruz:

```
var(" s a b c d ")
alan1=-1/16*(a^2+b^2-c^2-d^2)^2 + 1/4*(a*b+c*d)^2
alan2=((s-a)*(s-b)*(s-c)*(s-d)).subs(s=(a+b+c+d)/2)
bool( alan1 == alan2 )%
```

```
True
```

Not 8.7.1 Bu sadeleştirmeyi `bool` komutuna alternatif olarak aşağıdaki şekilde de yapabiliriz:

```
(alan1-alan2).simplify_full()
```

```
0
```

Alıştırma 8.7.1 (Heron Formülü) *Brahmagupta formülünün $d = 0$ için özel durumu üçgenin alanını bulmak için kullanılabilir. Kenar uzunlukları a, b, c birim ve*

$$s = \frac{a+b+c}{2}$$

olmak üzere, üçgenin alanını veren aşağıdaki formüle Heron formülü denir:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Üçgenin kenar uzunlukları girildiğinde alanını hesaplayan bir araç oluşturun. Ayrıca, girilen kenar uzunlukları bir üçgen oluşturmuyorsa bu aracın, kullanıcıyı uyarmasını istiyoruz. `if-else` *yapısı ve üçgen eşitsizliği konusunda yardımcı olabilecek detaylar 6.3 numaralı alt bölümde.*

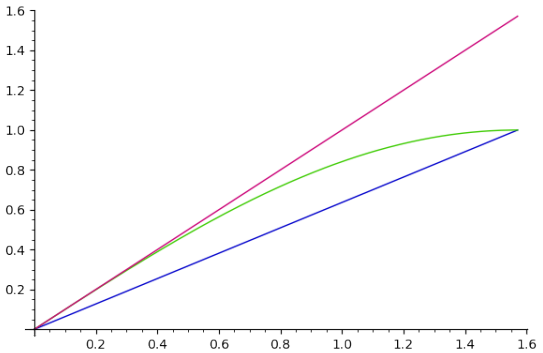
8.8 Jordan Eşitsizliği

$0 \leq x \leq \frac{\pi}{2}$ olmak üzere aşağıdaki eşitsizlik Jordan eşitsizliği olarak bilinir:

$$\frac{2}{\pi}x \leq \sin x \leq x$$

Geometrik bir bakış açısıyla bu eşitsizlik şunu söylüyor: Verilen aralıkta $y = \sin x$ fonksiyonunun grafiği, $y = \frac{2}{\pi}x$ ile $y = x$ doğrularının grafikleri arasında kalır. Bu üç fonksiyonun grafiğini söz konusu aralıkta çizdirelim:

```
plot( [2/pi*x , sin(x) , x ] , (x,0,pi/2) )
```



Gerçekten de $y = \sin x$ fonksiyonun grafiği eşitsizlikte verilen iki doğru arasına sıkışmış durumda. Ayrıca, grafikteki kesişim noktalarından anladığımıza göre x değişkeninin tanımlandığı aralığın uç değerlerinde eşitlik durumları oluşuyor: $x = 0$ olduğunda üç fonksiyon birbirlerine eşit ve sıfır oluyorlar; $x = \frac{\pi}{2}$ durumunda da $y = \sin x$ ve $y = \frac{2}{\pi}x$ fonksiyonları aynı anda 1 değerine, böylece birbirlerine eşit oluyorlar.

Bu resim ikna edici olsa da matematiksel bir ispat olmaktan çok uzak diye düşünebilirsiniz. Bunun felsefesini burada yapmayalım ama bu konuda biraz kafa yormak isteyenlere James Robert Brown'ın *Proofs and Pictures* (İspatlar ve Resimler) [3] adlı makalesini önerebilirim.

Alıştırma 8.8.1 (Kober Eşitsizliği) $0 \leq x \leq \frac{\pi}{2}$ *olmak üzere*

$$\cos x \geq 1 - \frac{2}{\pi}x$$

eşitsizliğini görselleştirin.

Not 8.8.1 Kober eşitsizliği Jordan eşitsizliğinin özel bir hali. Bunu $x \longmapsto \frac{\pi}{2} - x$ dönüşümüyle görebilirsiniz.

Alıştırma 8.8.2 (Bernoulli Eşitsizliği) $r > 1$ *ve* $x > -1$ *olmak üzere, aşağıda Bernoulli eşitsizliğinin bir varyantı var:*

$$(1 + x)^r \geq 1 + rx$$

Bu eşitsizlik için etkileşimli bir araç yaratın ve böylece pek çok r reel sayısı için ikna edici grafikler çizdirin.

Alıştırma 8.8.3 *Her* $x > 0$ *reel sayısı için* $x^x \geq x$. *Bu eşitsizliği görselleştirin.*

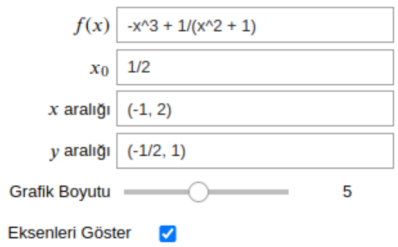
8.9 Türevin Geometrik Anlamı: Teğetin Eğimi

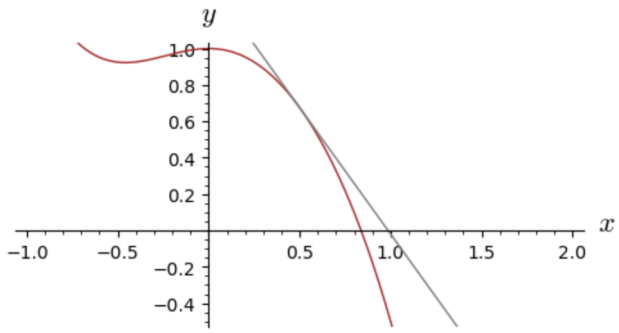
Bu alt bölümde, verilen bir f fonksiyonu ve x_0 değeri için teğet doğrusunun denklemini ve grafiğini veren etkileşimli bir araç yaratacağız. Bunun için türevin geometrik anlamını hatırlayalım: x_0 noktasında türevlenebilir bir $y = f(x)$ fonksiyonu için $f'(x_0) = m$ demek, $y = f(x)$ fonksiyonunun grafiğine $x = x_0$ değerinde çizilen teğet doğrusunun eğimi m demektir.

Aşağıda bir program ve onun çıktısını göreceksiniz. Sonrasında da bazı komutlara dair yorumlar var. Etkileşimli araç için detayları 6.6 numaralı alt bölümde bulabilirsiniz.

```
x, y = var("x y")
@interact
def _(f=input_box(label="$f(x)$",default= 1/(1+x^2) - x^3 ,width=50 ),
      x0=input_box(label="$x_0$",default= 1/2,width=30 ),
      x_aralik=input_box(label="$x$ aralığı",default=(-1,2) , width=30),
      y_aralik=input_box(label="$y$ aralığı",default=(-1/2,1) ,width=30),
      fs=slider( [1..10] ,label="Grafik Boyutu",default= 5 ),
```

```
  axes_check=("Eksenleri Göster", True)
 ):
a=x_aralik[0]
b=x_aralik[1]
c=y_aralik[0]
d=y_aralik[1]
D=diff(f,x)
egim=D.subs(x=x0)
y0=f(x=x0)
teget=plot(egim*(x-x0)+y0,(x,a,b),ymin=c,ymax=d,color="grey")
plo=plot(f,(x,a,b), ymin=c,ymax=d,color="brown" ,
    aspect_ratio=1,figsize=fs )
show(plo+teget, frame=not axes_check,
    axes=axes_check,axes_labels=["$x$","$y$"] )
show( LatexExpr(r"\text{Fonksiyon: } y="),f)
show( LatexExpr(r"(x_0,y_0)= "), (x0,y0))
show( LatexExpr(r"\text{Teğet Doğru: } y="),(egim*x-egim*x0+y0) )
```





Fonksiyon: $y = -x^3 + \dfrac{1}{x^2+1}$

$(x_0, y_0) = \left(\dfrac{1}{2}, \dfrac{27}{40}\right)$

Teğet Doğru: $y = -\dfrac{139}{100}\,x + \dfrac{137}{100}$

Yukarıdaki programa dair bazı açıklamalar:

1. f fonksiyonu, x_0 noktası ve çerçeveyi oluşturacak x ve y aralıkları `input_box` komutu ile kullanıcı tarafından girilecek şekilde ayarlanıyor.

2. Kullanıcının, grafik boyutunu kaydırgaç (*slider*) ile ayarlamasını istiyoruz.

3. Onay kutusu (*checkbox*), eksenler ve çerçeveyi gösterip gizlemek için kullanılacak.

4. Girilen x aralığı (a, b), y aralığı da (c, d) olacak şekilde değer atamalarını yapıyoruz.

5. f fonksiyonunun x değişkenine göre türevine D diyoruz ve hemen sonra da D ifadesinde $x = x_0$ uyguluyoruz. Bu şekilde $f'(x_0)$ değerini, yani eğimi buluyoruz ve bunu `egim` adlı değişkene atıyoruz.

6. Sonra da `egim`, `x0` ve `y0` değerlerini kullanarak adı `teget` olan teğet doğru denklemini oluşturuyoruz.

7. Grafikleri, girilen `a,b,c,d` sayılarıyla sınırlanmış şekilde çizdiriyoruz.

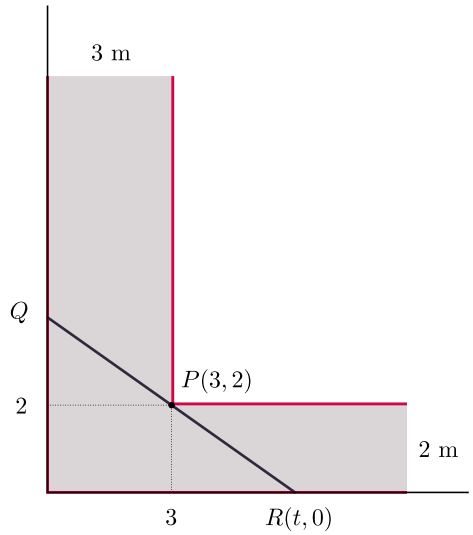
8. LaTeX ifadesi kullanma imkanı veren `LatexExpr` komutuyla gerekli bilgileri ekranda gösteriyoruz.

Diğer komut ve seçeneklerin yorumu okuyucuya bırakılmıştır.

Alıştırma 8.9.1 *Kullanıcının $y = f(x)$ fonksiyonunu girince kaydırgaç kullanarak en fazla 2 birim sağa ve 2 birim sola kaydıracağı bir grafik elde etmesini istiyoruz. Böyle bir interaktif araç oluşturun.*

Alıştırma 8.9.2 *Girilen bir f fonksiyonu için f^{-1} fonksiyonunu (f fonksiyonunun tersi) aynı koordinat sisteminde çizdiren bir interaktif araç oluşturun. Ters fonksiyonu gösterip gizlemek için onay kutusu (checkbox) kullanın. Eksen aralıkları ve grafik boyutu da kullanıcı tarafından değiştirilebilir olsun.*

8.10 Dik Koridor Problemi



Şekilde yukarıdan görüntüsü verilen dik köşeli koridordan, çelik düz bir boru (borunun kalınlığını ihmal ediyoruz) yere paralel tutulacak şekilde geçirilecektir. Koridor genişlikleri 3 metre ve 2 metre olduğuna göre, borunun boyu en fazla hangi uzunlukta olabilir?

Bu bölümde yukarıdaki probleme SageMath yardımıyla cevap vereceğiz. Sezgisel olarak, aranan en uzun borunun duvarlara P, Q ve R noktalarında değeceği açıktır. Aslında $P(3,2)$ noktasından geçip x eksenini $t > 3$ noktasında kesen doğrulardan QR doğru parçasını en kısa yapan durumla ilgileniyoruz. Benzer üçgenlerden yararlanarak veya doğru denklemini kullanarak $Q = \left(0, \frac{2t}{t-3}\right)$ olduğunu bulabilirsiniz. Böylece $|QR|$ uzunluğu t değişkenine bağlı aşağıdaki $L(t)$ fonksiyonu olacaktır:

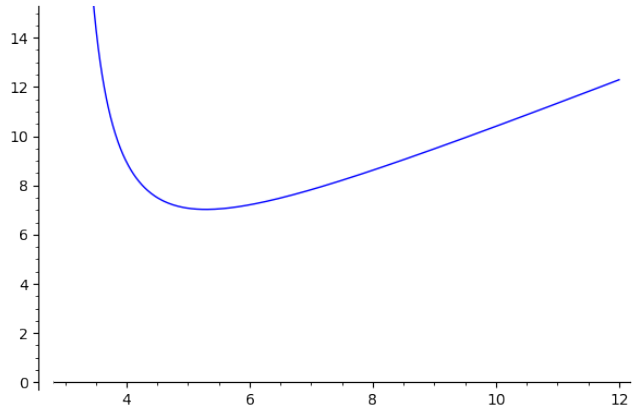
$$|QR| = L(t) = \sqrt{t^2 + \left(\frac{2t}{t-3}\right)^2}$$

Yukarıdaki resimden ya da denklemden görüleceği gibi, büyük t değerleri ve 3'e yakın t değerleri $L(t)$ uzunluğunu büyük yapacaktır. Biz $L(t)$ uzunluğunu en kısa yapan t değeri ile ilgileniyoruz.

Not 8.10.1 Karekök fonksiyonu artan bir fonksiyon olduğundan karekökün içini minimum yapan t değeri, kendisini de yani $|QR|$ değerini de minimum yapar. Böylece verilen L fonksiyonunda karekökün içini en küçük yapan t değerini bulup daha sonra $L(t)$ değerini bulabilirsiniz. Ama biz kareköklü şekliyle işlemlerimizi yapacağız.

Şimdi $L(t)$ fonksiyonunun grafiğini çizip uzunluğun t değişkenine göre nasıl değiştiğini görelim:

```
var("t")
rr=t
qq=2*t/(t-3)
L(t)=sqrt(rr^2+qq^2)
plot( L , (t,3,12) ,ymax=15, ymin=0 )
```



Grafikten gözlemlediğimize göre $L(t)$ fonksiyonunun ilgili aralıkta bir tane minimum değeri var. Şimdi de bu değeri bulmak için fonksiyonun türevinin sıfır olduğu t değerini bulmaya çalışalım:

```
cozum=solve(diff(L(t),t)==0,t)
show(cozum)
```

$$\left[t = -\frac{1}{2} \cdot 12^{\frac{1}{3}}\left(i\sqrt{3}+1\right)+3, t = -\frac{1}{2} \cdot 12^{\frac{1}{3}}\left(-i\sqrt{3}+1\right)+3, t = 12^{\frac{1}{3}}+3, t = 0\right]$$

Çözüm kümesinden anladığımıza göre iki tane reel kök var ve bunlardan $t = 0$ durumu $t > 3$ koşulunu sağlamadığından uygun değil. Bu durumda istediğimiz çözüm sondan ikinci çözüm, yani:

```
show(cozum[-2])
```

$$t = 12^{\frac{1}{3}}+3$$

Bu değeri L fonksiyonuna uygulayalım:

```
L(t).subs(cozum[-2]).show()
```

$$\sqrt{\frac{1}{3} \cdot 12^{\frac{1}{3}}\left(12^{\frac{1}{3}}+3\right)^2+\left(12^{\frac{1}{3}}+3\right)^2}$$

Son olarak da nümerik yaklaşık sonucu `n()` komutu ile bulalım:

```
L(t).subs(cozum[-2]).n()
```

```
7.02348237921997
```

Not 8.10.2 Eğer P noktasını (a, b) gibi genel bir nokta olarak alacak olursak problemi sembolik hesaplamalarla çözebiliriz. Bu durumda en küçük değerini istediğimiz fonksiyon

$$L(t) = \sqrt{t^2 + \left(\frac{tb}{t-a}\right)^2}$$

olacaktır.

Alıştırma 8.10.1 (Genel Durum) *Verilen problemi genel $P(a, b)$ durumu için çözün ve sonucun*

$$\left(a^{\frac{2}{3}} + b^{\frac{2}{3}}\right)^{\frac{3}{2}}$$

ifadesine eşit olduğunu sembolik hesaplamalarla gösterin.

Alıştırma 8.10.2 (İki Gemi) *Koordinat düzleminde biri $A(200, 0)$ noktasından kuzeye 50 km/saat, diğeri $B(0, 100)$ noktasından kuzeydoğuya 70 km/saat hızla aynı anda harekete başlayan iki gemi düşünün. Gemilerin birbirlerine en yakın olduğu an ne zaman olur? O anda aralarındaki mesafe kaç km'dir? (Bu soruya cevap vermek için mesafenin zamana bağlı fonksiyonunu ve bu fonksiyonun alacağı en küçük değer bulunmalıdır. Grafik ile sonuçları destekleyin.)*

8.11 Sophie Germain Asalları

p bir asal sayı olmak üzere, eğer $2p+1$ sayısı da asal ise p asalına *Sophie Germain asalı* denir (kısaca SGA diyelim). Örneğin 5 sayısı SGA'dır, çünkü 5'in kendisi asal olduğu gibi $2 \times 5 + 1 = 11$ sayısı da asaldır.

Bu alt bölümde SGA ile ilgili aşağıdaki üç uygulamayı yapacağız:

1. Bir sayının SGA olup olmadığını veren bir program.
2. İlk 100 asal sayıdan SGA olanları listeleyen bir program.
3. İlk 100 tane SGA'yı listeleyen program.

Alıştırma yapmak isterseniz bu üç uygulamayı önce kendiniz yapmayı deneyebilirsiniz. Bu uygulamalarda kullanacağımız temel komutlar sırasıyla `if`, `for` ve `while` komutlarıdır.

SGA mı, Değil mi?

Bu uygulama için `is_sga` diye bir fonksiyon tanımlayalım:

```
def is_sga(p):
       if is_prime(p)==True and is_prime(2*p+1)==True:
           print(True)
       else:
           print(False)
is_sga(5)
```

```
True
```

Yukarıdaki basit algoritma şunu yapıyor: Eğer p ve $2p+1$ sayıları asalsa ekrana `True` yaz, değilse `False` yaz.

İlk n Tane Asaldan SGA Olanlar

Aşağıdaki program da ilk n tane p asalı için $2p+1$ de asalsa ekrana p asalını yazdırıyor. Yani p asalının SGA olduğu durumda onu ekrana yazdırıyor.

```
def sga(a):
    for i in range(a):
        p=Primes()[i]
        if is_prime(2*p+1)==True:
            print(p, end=" ")
sga(100)
```

```
2 3 5 11 23 29 41 53 83 89 113 131 173 179 191 233 239 251 281 293 359
    419 431 443 491 509
```

İlk n Tane SGA

Şimdi de ilk n tane SGA'yı veren bir program yazalım. Tanımladığımız foksiyonun adını öncekilerden farklı olsun diye `SGA` diye adlandıralım.

```
def SGA(m):
    sga_liste=[]
    k=0
    while len(sga_liste)<m:
        asal=Primes()[k]
        if is_prime(2*asal+1)==True:
            sga_liste.append(asal)
        k=k+1
    print(sga_liste)
```

Yukarıdaki programı şöyle yorumlayabiliriz: `sga_liste` adında boş bir listeyle işe başlıyoruz. Bu listeye SGA testinden geçen sayıları ekleyeceğiz. Aslında $2p+1$ testinden

geçen asalları demek daha doğru. Bu `sga_liste` listesinin eleman sayısı m'den küçük olduğu sürece bir sonraki asal sayıyı testten geçireceğiz. Bir sonrakine geçiş için k sayacını 1 arttırıyoruz. `while` komutu ile ilgili detayları 6.5 numaralı alt bölümde bulabilirsiniz. Programı daha detaylı incelemek okuyucuya kalsın.

```
SGA(100)
```

```
[2, 3, 5, 11, 23, 29, 41, 53, 83, 89, 113, 131, 173, 179, 191, 233, 239,
    251, 281, 293, 359, 419, 431, 443, 491, 509, 593, 641, 653, 659,
    683, 719, 743, 761, 809, 911, 953, 1013, 1019, 1031, 1049, 1103,
    1223, 1229, 1289, 1409, 1439, 1451, 1481, 1499, 1511, 1559, 1583,
    1601, 1733, 1811, 1889, 1901, 1931, 1973, 2003, 2039, 2063, 2069,
    2129, 2141, 2273, 2339, 2351, 2393, 2399, 2459, 2543, 2549, 2693,
    2699, 2741, 2753, 2819, 2903, 2939, 2963, 2969, 3023, 3299, 3329,
    3359, 3389, 3413, 3449, 3491, 3539, 3593, 3623, 3761, 3779, 3803,
    3821, 3851, 3863]
```

Alıştırma 8.11.1 (Goldbach Sanısı) *Sayılar teorisindeki en popüler çözülmemiş problemlerden biri olan Goldbach sanısı diyor ki: 2'den büyük her çift sayı iki tane asal sayının toplamı şeklinde yazılabilir. Örneğin 28 çift sayısı 5 ile 23 asallarının toplamı olduğundan bu iddiayı sağlar.*

Girilen bir çift sayı için toplamları bu çift sayı olan iki asal sayı bulmaya çalışan bir program yazın.

Alıştırma 8.11.2 (Bertrand Postulatı) *Bertrand postulatının ifadesi şöyle: 1'den büyük her n doğal sayısı için n ile 2n arasında bir asal sayı vardır. 1845'te Joseph Bertrand tarafından ortaya atılan bu postulat, Chebyshev tarafından 1852 yılında ispatlandı. Böylece bu önermeye bir teorem diyebiliriz.*

Kullanıcının girdiği bir n değeri için n ile 2n arasında bir asal sayı veren bir fonksiyon ve bu foksiyonu kullanarak bir interaktif araç oluşturun.

8.12 Cassini Eğrileri

Elipsin geometrik yer ile ifade edilen tanımını hatırlayalım: Düzlemde verilen iki noktaya uzaklıkları toplamı sabit olan noktaların geometrik yerine elips denir. Bu tanımda "toplamı" yerine "çarpımı" yazarsak ne olur? Bu problem 1680 yılında Giovanni Domenico Cassini adında İtalyan bir astronom matematikçi tarafından incelenmiş ve oluşan eğriler de ismini bu matematikçiden almış.

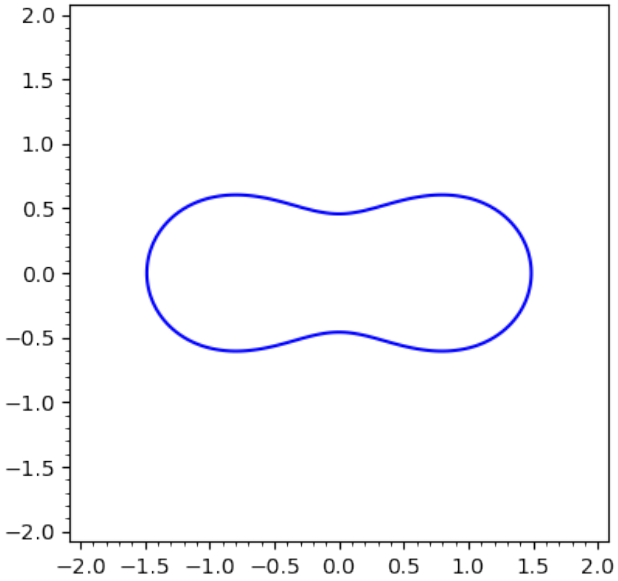
Söz konusu noktalar P_1 ve P_2, $b > 0$ bir sabit sayı olmak üzere $|PP_1| \times |PP_2| = b^2$ koşulunu sağlayan P noktalarının kümesi bir *Cassini eğrisi* olacaktır. P_1 ve P_2 noktalarına eğrinin *odak*ları denir.

Odak noktalarını koordinat düzleminde $P_1(-a, 0)$ ve $P_2(a, 0)$ olarak alıp uzaklık formülünü kullanacak olursak, eğrinin denklemi şöyle olur:

$$\left((x-a)^2 + y^2\right)\left((x+a)^2 + y^2\right) = b^4$$

Burada $e = \frac{b}{a}$ oranı sabit kalacak şekilde a ve b sayılarını değiştirirsek, benzer (yani, biri diğerinin düzgün büyütülmüş ya da küçültülmüş hali) Cassini eğrileri buluruz. Bu e oranı eğrinin karakterini belirler. $e < 1$, $e = 1$ ve $e > 1$ durumları farklı karakterde grafikler verir. Aşağıdaki kodla $e = 1.1$ durumunu çizdiriyoruz:

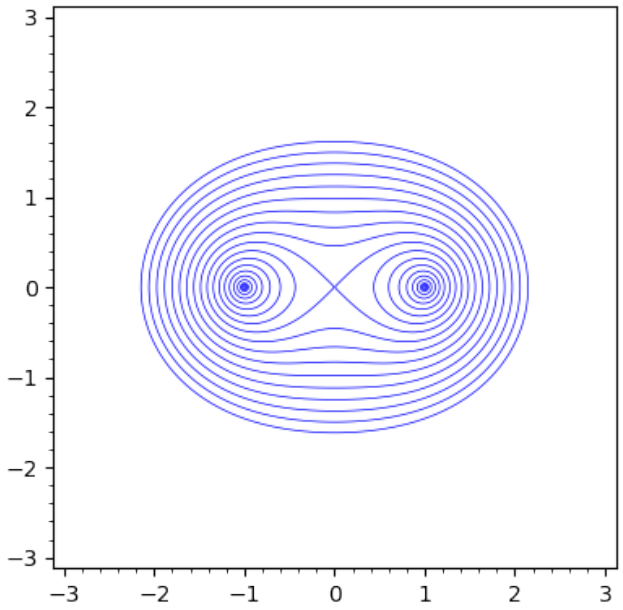
```
var("y b")
a=1
b=1.1
implicit_plot( ((x-a)^2+y^2)*((x+a)^2+y^2)==b^4 , (x,-2,2), (y,-2,2) )
```



Şimdi de sadece b değerini değiştirip $e = 0.1$'den $e = 2$'ye kadar 0.1'lik adımlarla pek çok Cassini eğrisi çizdirelim. Aşağıda bunun için yazılmış, aynı işlevi gören farklı iki kod var. İlk programda farklı b değerlerine karşılık gelen grafikler `+=` komutuyla biriktiriliyorken, ikinci programda bu işlem `sum` komutuyla yapılıyor:

```
resim=plot([])
for b in srange(.1,2,.1):
    resim += implicit_plot( ((x-1)^2+y^2)*((x+1)^2+y^2)==b^4, (x,-3,3),
        (y,-3,3) , linewidth=.5 , plot_points=400 )
show(resim)
```

```
var("y b")
sum( implicit_plot( ((x-1)^2+y^2)*((x+1)^2+y^2)==b^4 , (x,-3,3),
    (y,-3,3), linewidth=.5 , plot_points=400 ) for b in
    srange(0.1,2,.1) )
```



Not 8.12.1 $e < 1$ durumu iki tane kapalı simetrik eğri veren durumdur. $e > 1$ olduğu durum ise bir tane kapalı eğri verir ve bu kapalı eğri $e \geq \sqrt{2}$ durumunda dışbükeydir.

Not 8.12.2 (Bernoulli'nin Lemniskatı) $e = 1$ olduğu durum yatay 8 şeklindeki durumdur ve bu eğriye *Bernoulli'nin Lemniskatı* denir.

Alıştırma 8.12.1 (İnteraktif Araç) *Cassini eğrilerini* $0.1 \leq e \leq 2$ *aralığında 0.05'lik adımlarla görmemizi sağlayacak kaydırgaçlı interaktif bir araç oluşturun.*

Alıştırma 8.12.2 (Animasyon) *Cassini eğrilerini* $0.1 \leq e \leq 2$ *aralığında 0.05'lik adımlarla görmemizi sağlayacak bir animasyon oluşturun.*

8.13 Bir Aşk Hikayesi

Bu bölümde Steven Strogatz'ın 1988'de Harvard Üniversitesi tarafından yayımlanan *Mathematics Magazine* adlı dergideki makalesini [9] biraz yerelleştirip Leyla ile Mecnun'un aşk hikayesini modelleyen basit bir diferensiyel denklem sistemi oluşturacağız. Şöyle sağlıksız bir ilişkileri olduğunu varsayalım: Mecnun Leyla'nın sevgi ya da nefretine benzer tepkiyi veriyorken, Leyla Mecnun'un sevgi ya da nefretine karşıt bir tepki veriyor. Örneğin Leyla Mecnun'u sevdikçe Mecnun da onu seviyor, Leyla ilgisizken o da ilgisizleşiyor. Diğer taraftan, Mecnun Leyla'yı sevdikçe Leyla'nın ilgisi azalmaya başlıyor, Mecnun ilgisini kaybedince de Leyla'nın ilgisi artıyor.

L fonksiyonu Leyla'nın Mecnun'a olan sevgi (pozitif değer) ya da nefret (negatif değer) duygusunu, M fonksiyonu da Mecnun'un Leyla'ya olan sevgi ya da nefret duygusunu ifade etsin. Aşağıdaki diferensiyel denklem sistemi Leyla ile Mecnun ilişkisini modelleyebilir:

$$\begin{aligned} \frac{dL}{dt} &= -M, \\ \frac{dM}{dt} &= L \end{aligned} \tag{8.7}$$

Örneğin Mecnun Leyla'yı seviyorken, yani M pozitif iken, ilk denklemde L'nin türevinin negatif olacağını, buradan da Leyla'nın sevgisinin azalacağını görebiliriz. Benzer gözlemleri size bırakalım ve birbirlerine olan sevgileri 1'er birim iken nasıl bir hayat onları bekliyor görelim. Yani sistemi $L(0) = 1$, $M(0) = 1$ başlangıç koşuluyla çözelim. Önce diferensiyel denklem sistemini SageMath'e tanıtalım. İlgili bazı detaylar için 7.15 ve 7.18 numaralı alt bölümleri inceleyebilirsiniz.

```
var("t")
L = function("L")(t)
M = function("M")(t)
difdenk1= diff(L,t)==-M
difdenk2= diff(M,t)==L
```

Denklem sistemini tanıttık. Şimdi de `desolve_system` komutuyla sistemi çözelim. $L(0) = 1$, $M(0) = 1$ başlangıç koşulunu $[0, 1, 1]$ olarak yazdığımıza dikkat edin.

```
difdenkcozum = desolve_system( [difdenk1,difdenk2], [L,M], [0,1,1] )
show(difdenkcozum)
```

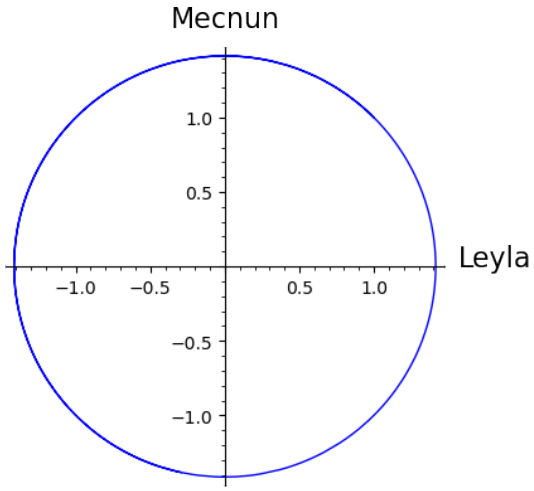
$$[L(t) = \cos(t) - \sin(t), M(t) = \cos(t) + \sin(t)]$$

Böylece (8.7) diferensiyel denklem sisteminin çözümünü parametrik olarak elde etmiş olduk.

Çözümün Grafiği ve Yorumu

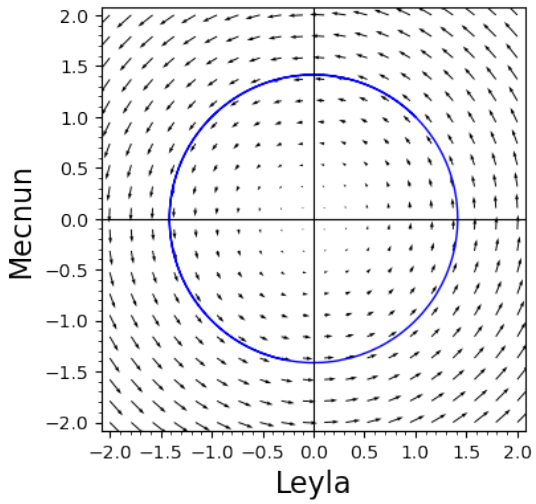
Yukarıda elde ettiğimiz çözümün grafiği aşağıdaki parametrik eğri (bu problem için çember) olacaktır:

```
var("t")
parametric_plot( [difdenkcozum[0].rhs() , difdenkcozum[1].rhs() ] ,
    (t,0,10), axes_labels=["Leyla","Mecnun"], aspect_ratio=1, ).show()
```



Bu başlangıç değer problemini $L(0) = 1$, $M(0) = 1$ için çözdük. Yani Leyla-Mecnun koordinat sisteminde, $(1, 1)$ noktası ile başlarsak neler olacağının grafiğini görüyoruz. Daha açıklayıcı olması açısından vektör alanını da grafiğe ekleyelim ve durumu yorumlayalım:

```
var("t L M")
p1=parametric_plot( [difdenkcozum[0].rhs() , difdenkcozum[1].rhs() ] ,
    (t,0,10), axes_labels=["Leyla","Mecnun"], aspect_ratio=1, )
p2=plot_vector_field( (-M,L), (L,-2,2), (M,-2,2), aspect_ratio=1 )
show(p1+p2)
```



Birbirlerine olan sevgileri 1'er birim olarak başlarsa, bu pozitif değere karşılık Mecnun'un sevgisi artacakken, Leyla'nınki azalacaktır. Yani grafikte (1, 1) noktasından başlayıp saat yönünün tersine bir hareket olacaktır. L negatif olduğunda Mecnun'un sevgisi de azalmaya başlayacak ve bir süre sonra (M negatif olduğunda) Leyla'nın sevgisi artacaktır. Buna karşılık Mecnun'un sevgisi tekrar artacak, Leyla'nın tekrar azalacak ve bu döngü bir ömür sürüp gidecektir.

Alıştırma 8.13.1 *(8.7) diferensiyel denklem sisteminin çözümü olarak elde edilen, zamana bağlı $L(t)$ ve $M(t)$ fonksiyonlarını aynı koordinat sisteminde çizdirip yorumlayın. Bu konuda 3.11 numaralı alt bölüm yardımcı olabilir.*

Alıştırma 8.13.2 *Aşağıdaki diferensiyel denklem sistemi veriliyor:*

$$\frac{dx}{dt} = -\frac{x}{2} + \frac{y}{3},$$
$$\frac{dy}{dt} = \frac{x}{3} - \frac{y}{3}$$

$x(0) = 1$, $y(0) = \frac{1}{2}$ başlangıç koşuluyla başlangıç değer problemini çözün ve çözümün grafiğini sistemin vektör alanıyla birlikte çizdirip çözümü yorumlayın.

8.14 Lojistik Fonksiyon

Aşağıdaki özyineli (recursive/rekürsif) fonksiyonu düşünelim:

$$x_{n+1} = f(x_n)$$

Bir x_0 değeri verilmiş olsun. Bu durumda:

$$\begin{aligned} x_1 &= f(x_0), \\ x_2 &= f(x_1) = f(f(x_0)) = f^2(x_0), \\ x_3 &= f(x_2) = f(f^2(x_0)) = f^3(x_0) \\ &\vdots \\ x_k &= f^k(x_0) \end{aligned}$$

olur. Bu bileşke işlemleri sonucunda *yörünge* dediğimiz

$$\{x_0, x_1, x_2, x_3, \dots, x_k, \dots\}$$

dizisini elde ederiz.

Bu alt bölümün amacı Lojistik fonksiyon denilen meşhur

$$x_{n+1} = r x_n (1 - x_n)$$

özyineli fonksiyonunun farklı r parametre değerleri için davranışını, yani ne tip yörüngeler verdiğini hem nümerik olarak incelemek hem de daha anlaşılır olması açısından zaman serisi ile görselleştirmektir.

$$f(x) = rx(1 - x)$$

alıp örneğin $r = 3.1$ ve $x_0 = 0.5$ değerleri için x_1, x_2, x_3,...,x_{10} değerlerini bulalım. Tabii ki bu sıkıcı işlemi bir `for` döngüsü ile yapacağız:

```
r=3.1
x0=.5
f=r*x*(1-x)
L=[x0]
for i in range(10):
    L.append( f(x=L[-1]) )
show(L)
```

```
[0.500000000000000, 0.775000000000000, 0.540562500000000,
    0.769899519140625, 0.549178173659744, 0.767502672430025,
    0.553171192752663, 0.766235755209903, 0.555267420208218,
    0.765531088016937, 0.556429048019278]
```

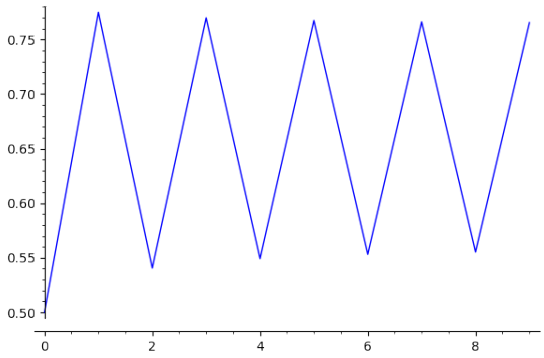
Elde ettiğimiz x_k değerlerini k değerleriyle birlikte aynı tabloda gösterelim:

```
T=[ (i,L[i] ) for i in range(10) ]
table(T ,header_row = ["$k$", "$x_k$"] )
```

k	x_k
0	0.500000000000000
1	0.775000000000000
2	0.540562500000000
3	0.769899519140625
4	0.549178173659744
5	0.767502672430025
6	0.553171192752663
7	0.766235755209903
8	0.555267420208218
9	0.765531088016937

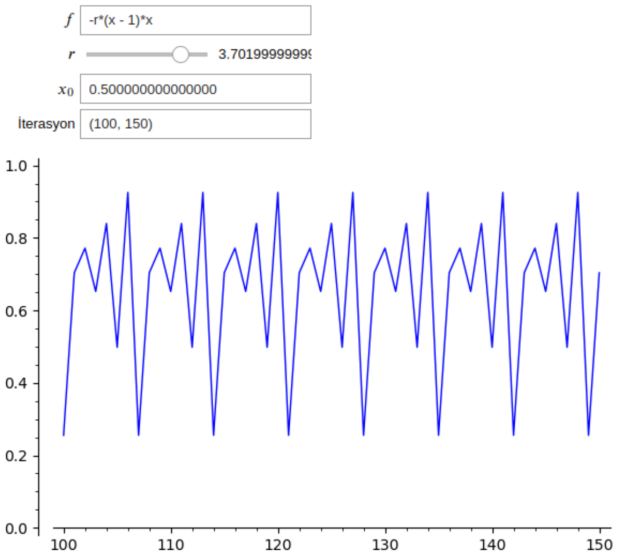
Tabloda verilen bilgileri biraz daha somutlaştıralım ve liste grafiği olarak gösterelim:

```
list_plot(L, plotjoined=True)
```



Elde ettiğimiz grafik $r = 3.1$ değeri için. Şimdi de $0 \leq r \leq 4$ olmak üzere, pek çok r değeri için denklemin davranışını gözlemleyebileceğimiz bir interaktif araç yaratalım:

```
var("r f x0 L")
@interact
def _( f=input_box( label="$f$", default=r*x*(1-x) ),
     param=slider( srange(2.7,4,.001) ,label="$r$",default= .5 ),
     x0=input_box( label="$x_0$",default=.5 ),
     iterasyon=input_box( label="iterasyon", default=(0,10) )
):
    f=f(r=param)
    L=[x0]
    for i in [1..iterasyon[1] ]:
        L.append( f(x=L[-1]) )
    gosterL=[ (k,L[k]) for k in [iterasyon[0]..iterasyon[1] ] ]
    graf=list_plot(gosterL, plotjoined=True ,ymax=1,ymin=0 )
    show(graf)
```



Yukarıdaki programda pek yeni bir şey yok aslında. Kulanıcının r parametresi olarak girdiği değeri `param` değişkenine atıyoruz ve bu değeri de `f=f(r=param)` satırıyla f fonksiyonuna uyguluyoruz.

Bu interaktif aracı kullanarak farklı r değerleri için Lojistik denklemin ne kadar farklı davranışlar sergilediğini gözlemleyebilirsiniz. Buna kaotik davranış da dahildir.

Alıştırma 8.14.1 (Collatz Sanısı) *1'den büyük pozitif bir tam sayı seçin. Sayı çift ise 2'ye bölün, tek ise 3 ile çarpıp 1 ekleyin. Elde ettiğiniz yeni sayıya aynı şeyi uygulayın. Çift ise 2'ye bölün, tek ise 3 ile çarpıp 1 ekleyin. Mesela 34 ile başlayalım ve bu kuralı uygulayalım:*

$$34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Collatz sanısı diyor ki, bu işlemi yeteri kadar sayıda uygularsanız bütün başlangıç değerleri için 1 sayısına ulaşırsınız. Collatz sanısının ifadesi oldukça basit ve anlaşılır olsa da günümüze kadar ispatlanamamıştır ve bu sanı matematikteki çözülememiş en ünlü problemlerden biridir.

a. *Kullanıcının girdiği sayıya tekrar tekrar yukarıdaki kuralı uygulayıp elde ettiği sayıları listeleyen bir program yazın. Ayrıca, zaman serisi kullanarak bu listeyi görselleştirin.*

b. *İlk 1000 tane doğal sayının bu sanıyı sağladığını SageMath yardımıyla gösterin.*

Alıştırma 8.14.2 (Göç Problemi) *İzmir ve İstanbul'un nüfusları sırasıyla 4.4 ve 15.5 milyondur. Varsayalım ki her sene İzmir'in nüfusunun yüzde 2'si İstanbul'a, İstanbul'un nüfusunun da yüzde 1'i İzmir'e göç ediyor. Yüz yıl sonra bu şehirlerin nüfusları nasıl olur? İki şehir için senelik değişimleri tablo ve zaman serisi olarak gösterin.*

8.15 Doğum Günü Problemi

Bir odada 23 kişi var. Bu odadaki birilerinin aynı doğum gününe sahip olma olasılığı % 50 gibi bir şey. Kişi sayısı 75 olsaydı bu olasılık % 100'e yakın olacaktı. Şaşırtıcı ama gerçek.

Bu bölümde n kişilik bir grupta en az iki kişinin aynı doğum gününe sahip olma olasılığını SageMath yardımıyla hesaplayacağız. Birkaç varsayımla başlayalım: Grupta ikiz, üçüz vb. olmadığını, yılın 365 gün ve doğum günlerinin eş olası olduğunu varsayıyoruz.

Daha somut olması açısından, ilk olarak grubun 23 kişi olduğu durumu inceleyelim; sonra genelleriz. Bu grupta en az iki kişinin ortak doğum gününe sahip olması olayına A diyelim. Bu durumda hiç kimsenin ortak doğum gününe sahip olmaması olayı A'nın tümleyeni yani A^c olacaktır. Bizim amacımız A'nın olasılığını, yani $P(A)$ değerini bulmak. Bunun için önce hesaplaması daha kolay olan $P(A^c)$ olasılığını bulacağız sonra da

$$P(A) = 1 - P(A^c) \tag{8.8}$$

eşitliğinden istediğimiz olasılığı bulabiliriz. A^c olayını, yani hiç kimsenin aynı doğum gününe sahip olmaması durumunu şöyle ifade edelim: 2'nci kişinin 1'inci kişiden farklı, 3'üncü kişinin 2 ve 1'inci kişilerden farklı, 4'üncü kişinin 3, 2 ve 1'inci kişilerden farklı ve benzer şekilde 23'üncü kişinin de 22'nci, 21'inci, ..., 2'nci ve 1'inci kişilerden farklı doğum gününe sahip olması durumu. Matematiksel olarak 2'inci kişinin 1'inci kişiden farklı olma olasılığı $1 - \frac{1}{365} = \frac{364}{365}$ olacaktır. 1'incinin 2'nciden farklı olması koşuluyla 3'üncünün diğer ikisinden farklı olma olasılığı $1 - \frac{2}{365} = \frac{363}{365}$ olacaktır. Benzer şekilde bir hesaplamayla 23 kişinin farklı doğum günlerine sahip olma olasılığını aşağıdaki gibi elde ederiz:

$$P(A^c) = \frac{364}{365} \times \frac{363}{365} \times \frac{362}{365} \times \cdots \times \frac{343}{365}$$

Buradan, (8.8) eşitliğini kullanarak $P(A)$ olasılığını elde ederiz.

Genel olarak, k kişi için bu çarpımı $1 - \frac{k-1}{365}$ terimine kadar sürdürürüz. SageMath ile bu hesaplamayı yaparken bir `for` döngüsü işleri kolaylaştıracaktır. Aşağıda gruptaki kişi sayısına bağlı olarak istediğimiz olasılığı veren, `dg` adında bir doğum günü fonksiyonu tanımlıyoruz:

```
def dg(k):
    olas=1
    for j in [1..(k-1)]:
        olas=olas*(1-j/365)
    return( (1-olas).n() )
```

```
dg(23)
```

```
0.507297234323986
```

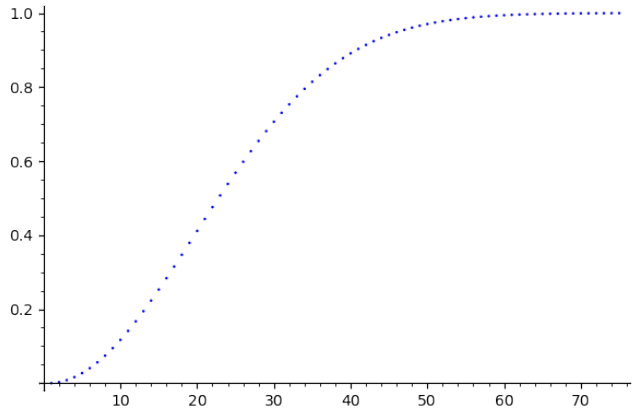
Yukarıda, 23 kişilik durum için olasılık sonucunu elde ettik. 75 kişi için:

```
dg(75)
```

```
0.999719878173811
```

Bu durumu daha somut ve genel bir şekilde görmek için grafiği inceleyelim. Kişi sayısı ve ona karşılık gelen olasılık bilgisini veren grafiği ilk 75 değer için `list_plot` ile çizelim:

```
list_plot( [ (j, dg(j)) for j in [1..75] ] , size=3 )
```



Görüldüğü üzere oldukça hızlı büyüyen bir fonksiyon var karşımızda.

Not 8.15.1 Bu hızlı büyüme kişi sayısı biraz arttığında hesaplamada bazı sorunlara neden olur. Eğer `dg(150)` değerini hesaplarsanız cevabın saçma bir şekilde 1 olduğunu göreceksiniz. Bunun sebebi SageMath'in kullandığı hassassiyetin (*precision*) varsayılan değerinin 53 bit olması. Bunu arttırmak için `prec` komutunu kullanırız. Aşağıda bu hassasiyeti 600 bit alıyoruz ve artık 365'e kadar hassas hesaplamalar yapacak durumdayız.

```
def dg(k):
    olas=1
    for j in [1..(k-1)]:
        olas=olas*(1-j/365)
    return( (1-olas).n(prec=600) )
```

```
dg(365)
```

```
0.99999999999999999999999999999999999999999999999999999999999999999\
99999999999999999999999999999999999999999999999999999999999999999\
99999999999999999999854504478438129659662860
```

Not 8.15.2 Görüldüğü üzere kişi sayısının 365 olduğu durumda bile olasılık hala 1 değil, 1'den biraz küçük. Çünkü olasılık `1-dg(365)` gibi çok küçük bir sayı olsa da 365 kişinin doğum gününün 365 farklı günde olma olasılığı var tabii ki. Ama 366 (ya da daha fazla) kişi olduğu durumda doğum günü aynı gün olan birilerinin kesin olarak bulunacağı açıktır. Bu durum matematikte *Güvercin Yuvası İlkesi* olarak bilinir.

Alıştırma 8.15.1 $x_0 = 2$ *olmak üzere*

$$x_{n+1} = \frac{1}{2}\left(\frac{n}{x_n} - x_n\right)$$

özyineli fonksiyonu veriliyor. İlk 50 iterasyonu zaman serisi olarak gösterin.

8.16 Sudoku

SageMath sizin için sudoku da çözüyor. Çözmek istediğiniz sudoku bulmacasını matris olarak yazıyorsunuz ve `sudoku` komutu size problemin bir çözümünü veriyor. Bilmemiz gereken tek şey bulmacadaki boşluklar yerine 0 yazıyor olmamız. Aşağıda basit bir 4×4 sudoku bulmacası çözdürüyoruz.

```
S = matrix( [ [3,0,4,0],[0,1,0,2],[0,4,0,3],[2,0,1,0] ] )
S
```

```
[3 0 4 0]
[0 1 0 2]
[0 4 0 3]
[2 0 1 0]
```

```
sudoku(S)
```

```
[3 2 4 1]
[4 1 3 2]
[1 4 2 3]
[2 3 1 4]
```

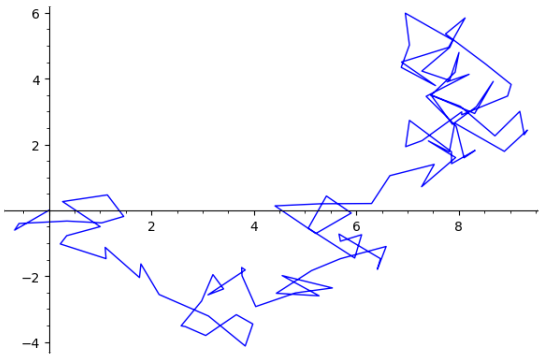
8.17 Algoritmik Sanat

Programlama konusunda gelişmek için en uygun alanlardan biri hiç şüphesiz Algoritmik Sanat. Çünkü yaratıcılığınız sınırsız takılırken kodlama beceriniz de onu takip etmek zorunda kalacak ve böylece eğlenerek gelişme şansınız olacaktır. Bu bölümde SageMath kullanarak birkaç adımda basit bir algoritmik sanat örneği oluşturacağız.

Rastgele İlerleyen Bir Doğru Parçası Zinciri

Diyelim ki koordinat düzleminde $(0,0)$ noktasını alıp bu noktanın x ve y bileşenlerine -1 ile 1 arasından rastgele seçilen birer sayı ekledik. Böylece yeni bir nokta elde ettik. Yeni noktaya aynı işlemi uyguladık. Yani rastgele sayılar ekleyip bir nokta daha elde ettik. Bu işlemi 100 defa uygulayıp noktaları aynı sırayla, doğru parçaları kullanarak birleştirdik. Aşağıda bu işlemi yapan bir program yazıyoruz. Detaylar ise şöyle: Sadece ilk noktayı, yani $(0,0)$ noktasını içeren bir L listesi oluşturarak işe başlıyoruz. Her adımda bu listenin son elemanının bileşenlerine rastgele sayıları, yani `rx` ve `ry` değerlerini (rastgele seçim için 4.9 numaralı bölüme bakınız) ekliyoruz. Daha sonra `append` komutu ile listeyi genişletip `line` komutu ile noktaları birleştirip son olarak da grafiği çizdiriyoruz.

```
x0=0; y0=0
L=[(x0,y0)]
for i in range(100):
    rx=-1+2*random()
    ry=-1+2*random()
    L.append( (L[-1][0]+rx , L[-1][1]+ry ) )
line(L)
```



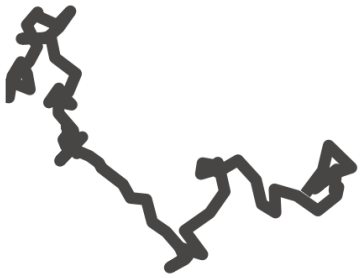
Şimdi de bu işlemin biraz daha detaylı ve bol seçenekli halini fonksiyon kullanarak yapmaya çalışalım. Aşağıda `rastgele1` diye bir fonksiyon tanımlıyoruz. Girdi olarak sırasıyla; başlangıç noktası P_0, renk, çizgi kalınlığı, saydamlık, boy/en oranı, x aralığı, y aralığı ve figür boyutu var. Bu arada `axes=False` komutu ile eksenleri gizleyelim.

```
def rastgele1(P0,iterasyon,renk,kalinlik,saydamlik,oran, xaralik,
    yaralik, figboyut ):
    x0=P0[0]
    y0=P0[1]
    L=[(x0,y0)]
    for i in [1..iterasyon]:
        rx=-1+2*random()
```

```
        ry=-1+2*random()
        L.append( (L[-1][0]+rx , L[-1][1]+ry ) )
    return line(L, axes=False, hue=renk, thickness=kalinlik,
         alpha=saydamlik, aspect_ratio=oran, xmin=xaralik[0],
         xmax=xaralik[1], ymin=yaralik[0], ymax=yaralik[1],
         figsize=figboyut )
```

Fonksiyonu tanımladık. Şimdi de bunu çalıştırıp bir doğru parçası zinciri elde ediyoruz.

```
rastgele1( (1.4,0) , 100, (.1, .1, .1 ), 10 , .8 , 1 , (-10,10),
     (-10,10) , 10 )
```

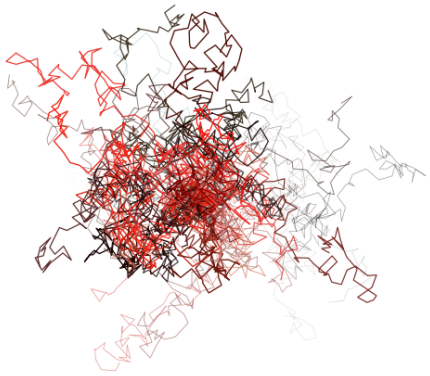


Burada renk kodunu RGB üçlüsü olarak almadık. Bunun yerine bu tip uygulamlarda yönetimi daha kolay olan HSV üçlüsü, yani Hue-Saturation-Value (Ton-Doygunluk-Değer) olarak aldık. Bunun SageMath ifadesi 3.8.1 numaralı notta verildiği üzere `hue` komutu.

Rastgele İlerleyen Pek Çok Doğru Parçası Zinciri

Şimdi `rastgele1` fonksiyonunu $(0,0)$ noktasına defalarca uygulayıp elde edilen grafik çıktılarını birleştirelim. Bunu yaparken grafiğin seçeneklerini `i` parametresine bağlı olarak değiştiridiğimize dikkat edin:

```
sum( rastgele1( (0,0), 100, ( 1/(i^2+1) , abs(cos((i^2+1) )) , random()
    ) , abs(cos(i^2+1)) , abs(cos(i^2+1)) , 1 , (-30,30), (-30,30) ,15
    ) for i in [-20..20] )
```



Kullanmış olduğumuz görsel seçenekler, komutlar ve fonksiyonlar için biraz açıklama, detay ve öneri faydalı olabilir:

1. Normalde grafik birleştirmeyi `p1+p2` şeklinde toplama işlemi ile yapıyoruz. Ancak burada pek çok grafik birleştireceğimiz için bu işlemi `sum` komutunu kullanarak yaptık.

2. Bu örnekte başlangıç noktası olarak orijini aldık ve `rastgele1` fonksiyonunu defalarca orijine uyguladık.

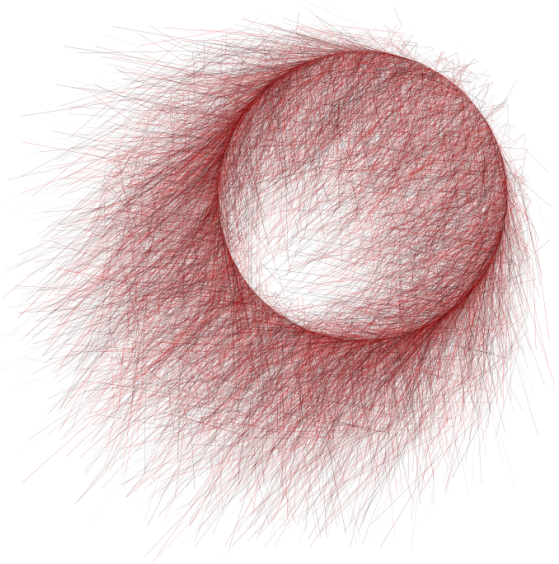
3. Renk için kullanılan `hue` komutu (HSV üçlüsü) önemli bir rol üstleniyor burada. Kırmızı yoğunluklu bir sonuç elde etmek istedik ve bunu `hue` komutunun ilk bileşenini birkaç adım sonra 0'a yakın değerler alacak olan $\frac{1}{i^2+1}$ fonksiyonu ile yaptık.

4. Kosinüs fonksiyonu 1'den büyük olmayan değerler alacağından, çizgiler hiçbir zaman çok kalın olmayacaktır. Ayrıca saydamlık da kalın çizgilerde daha az, yani çizgiler daha belirgin olacaktır.

Çembersel Başlangıç Noktaları

Şimdi de `rastgele1` fonksiyonunda küçük bir değişiklik yapalım: Rastgele sıçramaları `rx=-8+10*random()` ve `ry=-8+10*random()` olarak alalım. Yeni fonksiyonu uygulamak için de başlangıç noktası olarak $(0,0)$ yerine hareketli noktalar seçelim. Örneğin $(10\cos i, 10\sin i)$ şeklinde `i` parametresine bağlı bir fonksiyon alalım. Bu fonksiyon, başlangıç noktalarını merkezi orijin olan 10 birim yarıçaplı çember üzerinde gezdirecektir. Bununla birlikte çizgi kalınlığını 0.1 gibi çok daha ince seçip iterasyon sayısını da 2 gibi çok daha küçük bir sayı alalım. Çember üzerinde hareketli noktalara 10000 defa bu fonksiyonu uygulayalım. Sonuç aşağıda:

```
def rastgele1(P0,iterasyon,renk,kalinlik,saydamlik,oran, xaralik,
    yaralik, figboyut ):
    x0=P0[0]
    y0=P0[1]
    L=[(x0,y0)]
    for i in [1..iterasyon]:
        rx=-8+10*random()
        ry=-8+10*random()
        L.append( (L[-1][0]+rx , L[-1][1]+ry ) )
    return line(L, axes=False, hue=renk, thickness=kalinlik,
        alpha=saydamlik, aspect_ratio=oran, xmin=xaralik[0],
        xmax=xaralik[1], ymin=yaralik[0], ymax=yaralik[1],
        figsize=figboyut )
sum( rastgele1( (10*cos(i),10*sin(i)), 2, ( 1/(i^2+1) , abs(cos((i^2+1)
    )) , random() ) , .1 , abs(cos(i^2+1)) , 1 , (-30,30), (-30,30) ,15
    ) for i in [0..10000] )
```



Not 8.17.1 SageMath kullanılarak elde edilen çeşitli algoritmik sanat çalışmaları için:

`www.k-interact.net/ddsart`

Alıştırma 8.17.1 *Yukarıdaki kodları kullanarak, kullanıcının iterasyon sayısını, renk fonksiyonunu ve başlangıç noktasını (iterasyona bağlı bir fonksiyon) gireceği bir interaktif araç oluşturun.*

Alıştırma 8.17.2 *Yaratıcılığınızı kullanın ve kendinize ait bir algoritmik sanat eseri yaratın.*