



PYTHON'DA MODELLEME VE SİMÜLASYON

Bilim İnsanları ve Mühendisler için Bir Giriş

ALLEN B. DOWNEY

Yazar Hakkında

Allen Downey, DrivenData'da kadrolu bir bilim insanı ve Olin Üniversitesin'de emeritus profesördür. Olin Üniversitesin'de Modelleme ve Simülasyon dersleri vermiştir ayrıca yazılım ve veri bilimiyle ilgili başka dersler de vermiştir. Think Python, Think Bayes ve Elements of Data Science ders kitapları da dâhil olmak üzere çeşitli ders kitaplarının yazarıdır. Daha önceleri Wellesley Üniversitesi'nde ve Colby Üniversitesi'nde ders vermiştir. Doktora derecesini 1997'de Berkeley'deki Kaliforniya Üniversitesi'nde bilgisayar bilimleri bölümünden almıştır. Lisans ve yüksek lisans derecelerini MIT'teki (Massachusetts Institute of Technology) inşaat mühendisliği bölümünden almıştır. Veri bilimi ve Bayesci istatistikle ilgili olan Probably Overthinking It isimli blogun yazarıdır.

Teknik Eleştirmen Hakkında

Valerie Barr her alandaki öğrencileri bilgisayarlı hesaplama tabi kılmak için disiplinler arası uygulamalar ve müfredatla ilgili stratejilere odaklanmaya on yıldan fazla bir zaman harcamıştır. Buna modelleme ve simülasyon, veri görselleştirme ve şimdilerde veri bilimiyle kesişen öteki alanlarla ilgili dersler oluşturmak ve sunmak da dâhil olmuştur. Rutgers Üniversitesin'de bilgisayar bilimleri bölümünden doktora derecesine sahiptir. Mount Holyoke College'da Jean Sammet Kürsüsü'nde görev almıştır ve şu anda Bard College'da Margaret Hamilton Kürsüsü'nde görev yapmaktadır ve burada Bard Network Computing Initiative'i başlatmaktadır.

Yayın Numarası: 2407
1. Baskı: Nisan 2024
ISBN 978-625-98552-4-0

Genel Yayın Yönetmeni
Fatih ÖZDEMİR

Yazar
Allen B. DOWNEY

Çevirmen
Özgür KAYA

Editör
Doç. Dr. Vedat ÇELİK
Dr. Mustafa Murat ARAT

Son Okuma
Muhammed TOPRAK

Dizgi
Veysel TOPRAK

Sertifika Numarası: 41825
Buzdağı Yayınevi
Yeni Bağlıca Mah.
1068 Sok. No:4/1
Etimesgut/ANKARA
t: +90 312 219 55 43
info@buzdagiyayinevi.com

Baskı
Başak Matbaacılık
51529

© 2024 Türkçe yayın hakları Buzdağı Yayınevi'ne aittir.

Copyright © 2023 by Allen B. Downey. Title of English-language original:
*Modeling and Simulation in Python: An Introduction for Scientists and
Engineers*, ISBN 9781718502161, published by No Starch Press Inc. 245 8th
Street, San Francisco, California United States 94103. The Turkish-language
1st edition Copyright © 2024 by Buzdağı Yayınevi under license by No Starch
Press Inc. All rights reserved.

Bu kitabın hiçbir bölümü, yazarın ve yayınevinin izni alınmadan basılı ve
dijital olarak çoğaltılamaz, yayınlanamaz.

İçindekiler

Teşekkür	ix
Giriş	xi
Bu Kitap Kimler için?	xii
Ne Kadar Matematik ve Bilime İhtiyacım Var?	xii
Ne Kadar Programlamaya İhtiyacım Var?	xiii
Kitaba Genel Bir Bakış	xiv
Modellemeyi Öğretmek	xviii
İşe Başlamak	xix
Python'ı Kurmak	xx
Jupyter'i Çalıştırmak	xx
Öneriler ve Düzeltmeler	xxi
I AYRIK ZAMANLI SİSTEMLER	1
1 MODELLEMeye GİRİŞ	3
1.1 Modelleme Çerçevesi	3
1.2 Düşen Peni Efsanesini Test Etmek	5
1.3 Python'da Hesaplama	7
1.3.1 Yanlış Hassasiyet	8
1.3.2 Birimlerle Hesaplama	8
1.4 Özet	11
1.5 Alıştırmalar	12
2 BİR BİSİKLET PAYLAŞIM SİSTEMİNİ MODELLEMek	15
2.1 Bisiklet Paylaşım Modelimiz	15
2.2 Fonksiyonları Tanımlamak	17
2.3 Print Deyimleri	18
2.4 if Deyimleri	20
2.5 Parametreler	21

2.6	for Döngüleri	22
2.7	TimeSeries	23
2.8	Çizdirme	25
2.9	Özet	25
2.10	Alıştırmalar	26
2.11	Kaputun Altındakiler	27
3	YİNELEMELİ MODELLEME	29
3.1	Bisiklet Paylaşım Modelimiz Üzerinde Yineleme Yapmak	29
3.2	Birden Çok State Nesnesi Kullanmak	30
3.3	Dokümantasyon	31
3.4	Negatif Bisikletleri Ele Almak	32
3.5	Karşılaştırma Operatörleri	33
3.6	Metrikleri Tanıtmak	34
3.7	Özet	36
3.8	Alıştırmalar	36
4	PARAMETRELER VE METRİKLER	37
4.1	Değer Döndüren Fonksiyonlar	37
4.2	Döngüler ve Diziler	39
4.3	Parametreleri Taramak	40
4.4	Artımlı Geliştirme	42
4.5	Özet	44
4.6	Alıştırmalar	44
4.7	Zorlayıcı Alıştırmalar	45
4.8	Kaputun Altındakiler	46
5	BİR NÜFUS MODELİ GELİŞTİRMEK	47
5.1	Verileri İncelemek	47
5.2	Mutlak ve Göreceli Hatalar	51
5.3	Nüfus Artışımı Modellemek	53
5.4	Nüfus Artışımı Simüle Etmek	55
5.5	Özet	57
5.6	Alıştırmalar	57
6	NÜFUS MODELİNİ YİNELEMEK	59
6.1	Sistem Nesneleri	59
6.2	Oransal Bir Artış Modeli	62
6.3	Güncelleme Fonksiyonunu Hesaba Katmamak	64
6.4	Doğum ve Ölümü Birleştirmek	65
6.5	Özet	65

6.6	Alıştırma	66
6.7	Kaputun Altındakiler	66
7	ARTIŞIN SINIRLARI	67
7.1	Kuadratik Artış	67
7.2	Net Artış	69
7.3	Dengeyi Bulmak	71
7.4	Fonksiyon Bozuklukları	72
7.5	Özet	74
7.6	Alıştırmalar	75
8	GELECEĞE YÖNELİK TAHMİNLERDE BULUNMAK	77
8.1	Tahminler Üretmek	77
8.2	Tahminleri Karşılaştırmak	80
8.3	Özet	82
8.4	Alıştırmalar	83
9	ANALİZ VE SEMBOLİK HESAPLAMA	87
9.1	Fark Denklemleri	87
9.2	Diferansiyel Denklemler	89
9.3	Analiz ve Simülasyon	90
9.4	WolframAlpha ile Analiz	92
9.5	SymPy ile Analiz	92
9.6	SymPy’da Diferansiyel Denklemler	93
9.7	Kuadratik Artış Modelini Çözmek	95
9.8	Özet	97
9.9	Alıştırma	98
10	ÖRNEK DURUM İNCELEMELERİ KISIM I	99
10.1	Tarihi Dünya Nüfusu	99
10.2	Bir Kuyruk mu, İki Kuyruk mu?	100
10.3	Somon Nüfuslarını Tahmin Etmek	101
10.4	Ağaç Büyümesi	102
II	BİRİNCİ DERECEDEKİ SİSTEMLER	103
11	EPİDEMİYOLOJİ VE SIR MODELLERİ	105
11.1	Freshman Salgını	105
11.2	Kermack-McKendrick Modeli	106
11.3	KM Denklemleri	107
11.4	KM Modelini Gerçeklemek	108

11.5	Güncelleme Fonksiyonu	110
11.6	Simülasyonu Çalıştırmak	111
11.7	Sonuçları Bir Araya Getirmek	112
11.8	Şimdi Bir TimeFrame ile	113
11.9	Özet	116
11.10	Alıştırma	116
12	MÜDAHALELERİ NİCELEMEK	117
12.1	Bağışıklamanın Etkileri	117
12.2	Metrikleri Seçmek	119
12.3	Bağışıklamayı Taramak	120
12.4	Özet	122
12.5	Alıştırma	122
13	PARAMETRELERİ TARAMAK	123
13.1	Beta'yı Taramak	123
13.2	Gamma'yı Taramak	125
13.3	SweepFrame Kullanmak	126
13.4	Özet	130
13.5	Alıştırma	130
14	BOYUTSUZLAŞTIRMA	131
14.1	Beta ve Gamma	131
14.2	Sonuçları İncelemek	132
14.3	Temas Sayısı	134
14.4	Analiz ile Simülasyonu Karşılaştırmak	136
14.5	Temas Sayısını Kestirmek	137
14.6	Özet	138
14.7	Alıştırma	138
14.8	Kaputun Altındakiler	139
15	TERMAL SİSTEMLER	141
15.1	Kahvenin Soğuması Problemi	141
15.2	Sıcaklık ve Isı	142
15.3	Isı Transferi	143
15.4	Newton'un Soğuma Yasası	144
15.5	Newton Soğumasını Gerçeklemek	144
15.6	Kökleri Bulmak	148
15.7	r'yi Kestirmek	150
15.8	Özet	151
15.9	Alıştırmalar	152

16 KAHVE PROBLEMİNİ ÇÖZMEK	153
16.1 Sıvıları Karıştırmak	153
16.2 Başta mı Karıştırmalı, Sonda mı Karıştırmalı?	155
16.3 Optimal Zamanlama	156
16.4 Analitik Çözüm	157
16.5 Özet	159
16.6 Alıştırmalar	159
17 KAN ŞEKERİNİ MODELLEMEK	161
17.1 Minimal Model	161
17.2 Glikozun Minimal Modeli	162
17.3 Verileri Elde Etmek	165
17.4 İnterpolasyon	165
17.5 Özet	167
17.6 Alıştırmalar	167
18 MİNİMAL MODELİ GERÇEKLEMEK	169
18.1 Modeli Gerçeklemek	170
18.2 Güncelleme Fonksiyonu	171
18.3 Simülasyonu Çalıştırmak	172
18.4 Diferansiyel Denklemleri Çözmek	175
18.5 Özet	179
18.6 Alıştırma	179
19 ÖRNEK DURUM İNCELEMELERİ KISIM II	181
19.1 Minimal Modele Yeniden Bakış	181
19.2 İnsülinin Minimal Modeli	182
19.3 Alçak Geçiren Filtre	183
19.4 Bir Duvarın Termal Davranışı	184
19.5 HIV	186
III İKİNCİ DERECEDEN SİSTEMLER	187
20 DÜŞEN PENİYE YENİDEN BAKIŞ	189
20.1 Newton'un İkinci Hareket Yasası	189
20.2 Penileri Düşürmek	191
20.3 Olay Fonksiyonları	194
20.4 Özet	195
20.5 Alıştırmalar	195
21 SÜRÜKLEME	197

21.1	Sürüklenme Kuvvetini Hesaplamak	197
21.2	Params Nesnesi	199
21.3	Peninin Düşüşünü Simüle Etmek	200
21.4	Özet	204
21.5	Alıştırmalar	204
22	İKİ BOYUTLU HAREKET	205
22.1	Varsayımlar ve Kararlar	205
22.2	Vektörler	206
22.3	Beyzbol Topunun Uçuşunu Simüle Etmek	209
22.4	Sürüklenme Kuvveti	211
22.5	Bir Olay Fonksiyonu Ekleme	213
22.6	Takip Edilen Yolları Görselleştirmek	215
22.7	Beyzbol Topunun Animasyonunu Yapmak	217
22.8	Özet	218
22.9	Alıştırmalar	218
23	OPTİMİZASYON	221
23.1	Manny Ramirez Problemi	221
23.2	Menzili Bulmak	223
23.3	Özet	226
23.4	Alıştırmalar	226
23.5	Kaputun Altındakiler	227
24	DÖNME HAREKETİ	229
24.1	Tuvalet Kâğıdının Fizîği	230
24.2	Parametreleri Belirlemek	231
24.3	Sistemi Simüle Etmek	232
24.4	Sonuçları Çizdirmek	234
24.5	Analitik Çözüm	236
24.6	Özet	238
24.7	Alıştırmalar	238
25	TORK	241
25.1	Açısal İvmelenme	241
25.2	Eylemsizlik Momenti	242
25.3	Demlikler ve Döner Tablalar	242
25.4	İki Aşamalı Simülasyon	245
25.4.1	Aşama 1	246
25.4.2	Aşama 2	247
25.4.3	Sonuçları Birleştirmek	247

25.5	Sürtünmeyi Kestirmek	249
25.6	Döner Tablanın Animasyonunu Yapmak	252
25.7	Özet	253
25.8	Alıştırmalar	253
26	ÖRNEK DURUM İNCELEMELERİ KISIM III	255
26.1	Bungee Atlaması	255
26.2	Bungee Dalışına Yeniden Bakış	256
26.3	Güneşin Etrafında Yörüngede İlerlemek	257
26.4	Örümcek Adam	257
26.5	Kedi Yavruları	259
26.6	Bir Yo-Yo'yu Simüle Etmek	260
26.7	Tebrikler	262
A	KAPUTUN ALTINDAKİLER	263
A.1	<code>run_solve_ivp</code> Fonksiyonu Nasıl Çalışır?	263
A.2	<code>root_scalar</code> Fonksiyonu Nasıl Çalışır?	268
A.3	<code>maximize_scalar</code> Fonksiyonu Nasıl Çalışır?	269
Dizin		271

Kitaba Genel Bir Bakış

Bu kitap üç kısım hâlinde düzenlenmiştir. Kısım I ayırık bileşenlerden oluşan sistemlerle ilgilidir. Örnekler arasında bisiklet paylaşım sistemi ve nüfus artışı bulunmaktadır. Bu kısımdaki bölümler aşağıdaki gibidir:

Bölüm 1: Modellemeye Giriş Bu bölüm, modelleme çerçevesini tanıtmakta ve düşen peni efsanesini test etmek suretiyle bu yazılım çerçevesini açıklamaktadır. Ayrıca, bu bölüm, metre ve kilogram gibi birimlerle hesaplama yapmaya yarayan bir kütüphane olan Pint'i sunmaktadır.

Bölüm 2: Bir Bisiklet Paylaşım Sistemini Modellemek Bu bölüm, modelleme çerçevesini bir bisiklet paylaşım sistemine uygulamakta ve fonksiyonlar, döngüler ve `TimeSeries` veri yapısı da dâhil olmak üzere bu kitap boyunca ihtiyacımız olacak programlama araçlarından bazılarını tanıtmaktadır.

Bölüm 3: Yinelemeli Modelleme Bu bölüm, bisiklet paylaşım modelimizi kademeli bir şekilde geliştirmek suretiyle yinelemeli modelleme sürecini açıklamaktadır. Bu süreç, bisikletleri takip etmek için State nesnelerini kullanmaktadır ve negatif bisikletler gibi imkânsız koşulları denetlemektedir.

Bölüm 4: Parametreler ve Metrikler Bisiklet paylaşım modelini çalıştırıyoruz ve modeli, müşteri talebindeki değişiklikler gibi bir dizi koşullar altında sistemin davranışını tahmin etmek için kullanıyoruz. Ayrıca, kodu test etme ve koddan hata ayıklama yolları da öneriyoruz.

Bölüm 5: Bir Nüfus Modeli Geliştirmek Bu bölüm, dünyanın nüfus artışıyla ilgili beş bölümden ilkidir. Bölüm, son elli yılın verileriyle başlamaktadır ve verilere uyup uymadıklarını görmek için çeşitli modelleri test etmektedir.

Bölüm 6: Nüfus Modelini Yinelemek Burada nüfus modelimizi kademeli olarak geliştiriyoruz modeli daha gerçekçi hâle getirmek ve kodu okunması ve değiştirilmesi daha kolay hâle getiriyoruz. Ayrıca, modellediğimiz sistemle ilgili bilgileri temsil eden System nesnesini tanıtıyoruz.

Bölüm 7: Artışın Sınırları Bu bölümde, nüfus artışının sınırlarını temsil etmek için modele yeni özellikler ekliyoruz ve fonksiyonları test etmek ve hata ayıklamak için yollar öneriyoruz.

Bölüm 8: Geleceğe Yönelik Tahminlerde Bulunmak Burada gelecek 80 yıllla ilgili projeksiyonlar üretmek ve bu projeksiyonları profesyonel demografların tahminleriyle karşılaştırarak dünya nüfusu üzerine olan çalışmamıza devam ediyoruz.

Bölüm 9: Analiz ve Sembolik Hesaplama Bu bölüm, dünya nüfusu modellerimizi analiz etmek için matematiksel yöntemler sunmakta ve sembolik hesaplama yapmaya yarayan bir kütüphane olan SymPy'ı tanıtmaktadır. Bu bölüm, matematiksel analizi sevenleri ilgilendirmelidir fakat isteğe bağlıdır: Bu bölümü atlamayı tercih ederseniz, kitapta daha sonraları ihtiyacımız olacak hiçbir şeyi kaçırmış olmazsınız.

Bölüm 10: Örnek Durum İncelemeleri Kısım I Bu bölüm, şu ana dek öğrendiğimiz yöntemleri çeşitli konulara uygulayan dört örnek durum incelemesini sunmaktadır: Tarih öncesi dünya nüfusu, vahşi yaşam nüfusu, ağaç büyümesi ve kuyruklama sistemleri.

Kısım II, ısınıp soğuyan nesnelere, kimyasal konsantrasyonlar ve tepkimeler ve bulaşıcı hastalığın yayılması da dâhil olmak üzere birinci dereceden diferansiyel denklemlerle açıklanabilen sistemlerle ilgilidir. Aşağıda bölümlerin dökümü bulunmaktadır:

Bölüm 11: Epidemiyoloji ve SIR Modelleri Bulaşıcı hastalığın modellenmesi üzerine olan dört bölümün ilki olmak üzere, bu bölüm, SIR modelini ve TimeFrame veri yapısını tanıtmaktadır.

Bölüm 12: Müdahaleleri Nicelemek Bu bölüm, karantina ve bağışıklama gibi bulaşıcı hastalığa yönelik müdahalelerin etkisini incelemek için SIR modelini kullanmakta ve sürü bağışıklığı hadisesini incelemektedir.

Bölüm 13: Parametreleri Taramak Bu bölüm, enfeksiyonun ve iyileşme oranlarının bulaşıcı bir hastalığın ilerleyişi üzerindeki etkisini incelemekte ve `SweepFrame` veri yapısını tanıtmaktadır.

Bölüm 14: Boyutsuzlaştırma Burada SIR modeliyle olan işimizi sonlandırıyoruz. Enfeksiyon oranı, iyileşme oranı ve bir hastalığın bulaşıcılığını ölçen temel çoğalma (üreme) sayısı arasındaki ilişkiyi anlamak için matematiksel analiz kullanıyoruz. Bu bölüm isteğe bağlıdır.

Bölüm 15 Termal Sistemler Termal sistemlerle ilgili iki bölümün ilki olmak üzere, bu bölüm, kahvenin soğuması problemini tanıtmaktadır. Bu problem, sıcaklık ve ısı akışı arasındaki ilişkiyi incelemektedir.

Bölüm 16: Kahve Problemini Çözmek Optimal karıştırma zamanını bulmak için farklı sıcaklıklardaki kahve ve süt karışımını modellemek suretiyle kahvenin soğuması problemini tamamlıyoruz.

Bölüm 17: Kan Şekerini Modellemek İnsan vücudundaki glikoz ve insülinin etkileşimini modellemek suretiyle kan şekerinin regülasyonunu inceleyen iki bölümden ilkidir. Bu bölüm interpolasyon (aradeğerleme) kavramını tanıtmaktadır.

Bölüm 18 Minimal Modeli Gerçeklemek Glikoz-insülin modelini bir ODE çözücüsü ve diferansiyel denklemlerle çalışan sayısal bir yöntem kullanarak uyguluyoruz. ODE (ÇN: Ordinary Differential Equations kısaltması, adi diferansiyel denklemler) çözücü, diferansiyel denklemler çözmek için kullanılan bir sayısal yöntemdir.

Bölüm 19: Örnek Durum İncelemeleri Kısım II Bu bölüm, bir elektrik devresi, ısıl olarak yalıtıcı bir duvar ve HIV ile bağışıklık sisteminin etkileşimi de dâhil olmak üzere şu ana dek öğrendiğimiz araçları çeşitli problemlere uygulayan örnek durum incelemelerini sunmaktadır.

Kısım III, uzayda hareket eden ve dönen nesnelere de dâhil olmak üzere, ikinci dereceden diferansiyel denklemlerle açıklanan sistemlerle ilgilidir. Örnekler arasında beyzbol topu gibi fırlatılan cisimler ve yo-yo gibi dönen nesnelere yer almaktadır. Bölümler şu şekildedir:

Bölüm 20: Düşen Peniye Yeniden Bakış Bölüm 1'deki düşen peni örneğine geri dönüyoruz. Bu sefer bir boyutta ilerleyen fırlatılmış bir cisimi modellemek için bir ODE çözücü kullanıyoruz.

Bölüm 21: Sürüklenme Hava direncinden kaynaklanan sürüklenmeyi de dâhil etmek için düşen peni modelini genişletiyoruz. Hava direnci olmadan sonuçlar doğru olmaya yakın bile değildir.

Bölüm 22: İki Boyutlu Hareket Hava direnci etkisi de dâhil olmak üzere bir beyzbol topunun uçuşunu modellemek için bir boyuttan iki boyuta geçiyoruz. Basit animasyonlar kullanarak sonuçları görselleştiriyoruz.

Bölüm 23: Optimizasyon Bir optimizasyon sorusunu çözerek beyzbol örneğini sonlandırıyoruz. Fenway Park'ta sayı vuruşu yapmak için gereken minimum çaba nedir?

Bölüm 24: Dönme Hareketi Dönme hareketi üzerine olan iki bölümün ilki olmak üzere, bu bölüm, açısal hız kavramını sunmakta ve bu kavramı bir tuvalet kâğıdı rulosunu üretme örneğine uygulamaktadır.

Bölüm 25: Tork Bu bölüm tork ve açısal ivmelenme kavramlarını tanıtarak kuvvet ve dönme hareketi içeren sistemlerin modellenmesini mümkün kılmaktadır.

Bölüm 26: Örnek Durum İncelemeleri Kısım III Bu bölüm, bir bungee atlayıcısı, bir yo-yo, Örümcek Adam ve bir tuvalet kâğıdını açan bir kedi yavrusu da dâhil olmak üzere, öğrendiklerimizi çeşitli sistemlere uygulayan altı örnek durum incelemesini sunmaktadır.

Her bir bölümün sonunda, öğrendiklerinizi uygulayabileceğiniz en az bir alıştırmayı sunuyoruz. Her bir kısmın sonunda, daha geniş çeşitlilikteki problemlerle çalışabileceğiniz örnek durum incelemeleri sunuyoruz.

Son olarak, kitabın ek kısmı “kapağı kaldırmakta” ve ODE çözücü, kök bulucu ve optimize edici de dâhil olmak üzere, kullandığımız bazı yöntemlerin ayrıntılarını açıklamaktadır.

Modellemeyi Öğretmek

Modellemenin temel becerileri –soyutlama, analiz, simülasyon ve doğrulama– mühendislik, doğa bilimleri, sosyal bilimler, tıp ve diğer birçok alanda merkezdedir. Bazı öğrenciler bu becerileri dolaylı olarak öğrenir fakat çoğu okulda bu beceriler açık bir şekilde öğretilmez ve öğrenciler az pratik yapmış olurlar. Bu kitabın ele almayı amaçladığı problem budur.

Olin Üniversitesi’nde bu becerileri bütün öğrencilerin ilk altı aylık dönemde aldıkları Modelleme ve Simülasyon adlı bir derste öğretiyoruz. Bu dersi meslektaşlarımız John Geddes ve Mark Somerville ile birlikte oluşturduk ve ilk kez 2009’da verdik. Bu ders, modellemenin açık bir şekilde, erkenden ve müfredat boyunca öğretilmesi gerektiği yönündeki inancımıza dayanmaktadır. Bu ders aynı zamanda hesaplamaların bu sürecin gerekli bir parçası olduğu yönündeki kanaatimize de dayanmaktadır. Eğer öğrenciler elle yapabildikleri matematiksel analizle sınırlı iseler, vakum içerisinde hareket eden atılmış bir cisim veya sürtünmesiz bir düzlem üzerindeki bir blok gibi az sayıdaki basit fiziksel sistemle sınırlı olurlar. Ve bu öğrenciler birçok kötü model görürler. Yani amaçladıklarıyla karşılaştırıldığında çok basit olan modeller.

Giriş niteliğindeki çoğu fizik dersinde öğrenciler modelleme kararları vermez. Bazen onlar için verilmiş kararların farkında bile değildirler. Amacımız modelleme sürecinin tamamını öğretmek ve öğrencilere bu sürecin pratiğini yapma şansı vermektir.

Ayrıca, programlamayı bir ön koşul olarak istemek yerine programlama ve modellemeyi aynı anda öğretmenin değerli olduğunu düşünüyoruz. Çoğu programlama dersi temel dil özelliklerinden başlayarak ve kademeli olarak daha güçlü araçlar ekleyerek aşağıdan yukarıya doğru ilerler. Sonuç olarak öğrencilerin Fahrenheit’ı Celsius’a çevirmekten daha ilginç bir şeyler yapabilmesi uzun zaman alır. Ve sıklıkla öğrencilerin bağlamı olmaz. Öğrenciler zihinlerinde belirli

5 sayıyı içerir. Sonuç, bir NumPy dizisidir ve bu dizi daha önce görmediğimiz yeni bir nesne türüdür. Dizi, bir sayı sekansının konteyneridir.

Bir diziyi bir for döngüsünde şu şekilde kullanabiliriz:

```
for p1 in p1_array:
    print(p1)
```

```
0.0
0.25
0.5
0.75
1.0
```

Bu döngü çalıştığında şunları yerine getirir:

1. Diziden ilk değeri alır ve p1'e atar.
2. Döngünün gövdesini çalıştırır ve bu da p1'i ekrana yazdırır.
3. Diziden bir sonraki değeri alır ve p1'e atar.
4. Döngünün gövdesini çalıştırır ve bu da p1'i ekrana yazdırır.

Döngü bunları dizinin sonuna ulaşıncaya dek yapar. Bu, bir sonraki kısımda kullanışlı olacaktır.

4.3 Parametreleri Taramak

p1 ve p2 gibi parametrelerin gerçek değerlerini bilirsek, bu değerleri, bir saat sonra Olin'de kaç tane bisikletin olacağı gibi spesifik tahminler yapmak için kullanabiliriz.

Ancak tahmin yapmak tek amaç değildir. Bunun gibi modeller, sistemlerin neden öyle davrandıklarını açıklamak ve alternatif tasarımları değerlendirmek için de kullanılır. Örneğin sistemi gözlemler ve bisikletlerin sıklıkla belirli bir zamanda tükendiğini fark edersek, neden böyle olduğunu anlamak için modeli kullanabiliriz. Ve eğer daha fazla bisiklet veya başka bir istasyon eklemeyi düşünüyorsak, çeşitli “eğer böyle olursa ne olur” senaryolarının etkisini değerlendirebiliriz.

Bir örnek olarak, p2'nin yaklaşık 0.2 olduğunu kestirmek için yeterince veriye sahip olduğumuzu fakat p1 hakkında hiçbir bilginin olmadığını farz ediniz. p1'in bir aralıktaki değerleriyle birlikte simülasyonlar gerçekleştirilebilir ve sonuçların nasıl değiştiğini görebiliriz. Bu süreç, parametrenin değerinin

olası bir değerler aralığı boyunca “tarandığı” anlamına gelecek şekilde, bir parametreyi *taramak* olarak adlandırılır.

Döngüler ve diziler hakkında bilgi sahibi olduğumuza göre onları aşağıdaki gibi kullanabiliriz:

```
p1_array = linspace(0, 0.6, 6)
p2 = 0.2
num_steps = 60

for p1 in p1_array:
    final_state = run_simulation(p1, p2, num_steps)
    print(p1, final_state.olin_empty)
```

```
0.0 0
0.12 0
0.24 0
0.36 3
0.48 14
0.6 11
```

Döngü boyunca her seferinde farklı bir *p1* değeri ve aynı *p2* değeri olan 0.2 ile bir simülasyon yaparız. Ardından, *p1*'i ve *Olin*'deki memnun olmayan müşteri sayısını ekrana yazdırırız.

Sonuçları kaydetmek ve çizdirmek için *TimeSeries* nesnesine benzer olan *SweepSeries* nesnesini kullanabiliriz. İkisi arasındaki fark, *SweepSeries* nesnesindeki etiketlerin zaman değerlerinden ziyade parametre değerleri olmasıdır.

Boş bir *SweepSeries* nesnesini şu şekilde oluşturabiliriz:

```
sweep = SweepSeries()
```

ve şu şekilde değerler ekleyebiliriz:

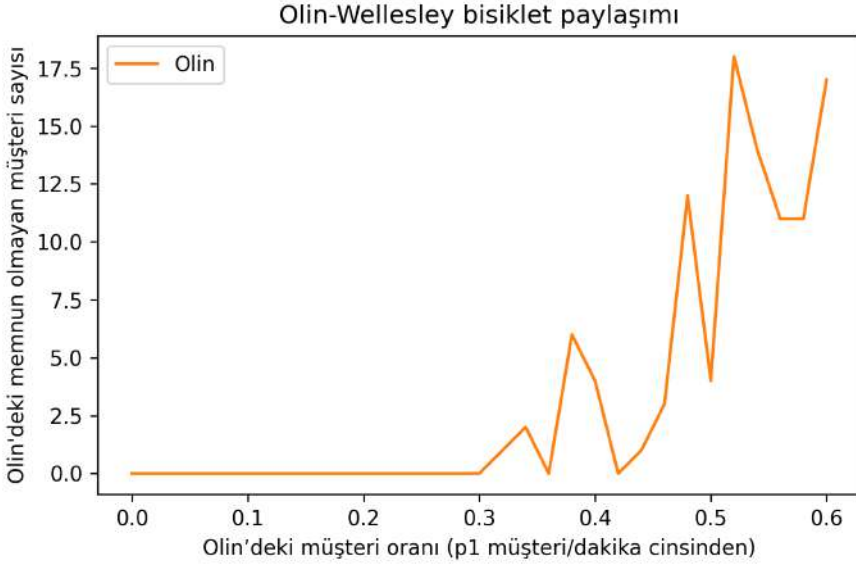
```
p1_array = linspace(0, 0.6, 31)

for p1 in p1_array:
    final_state = run_simulation(p1, p2, num_steps)
    sweep[p1] = final_state.olin_empty
```

Sonuç, *p1*'in her bir değerini memnun olmayan müşteri sayısına eşleyen bir *SweepSeries* nesnesidir.

SweepSeries'in elemanlarını şu şekilde çizdirebiliriz:

```
sweep.plot(label='Olin', color='C1')
decorate(title='Olin-Wellesley bisiklet paylaşımı',
  xlabel="Olin'deki müşteri oranı (p1 müşteri/dakika cinsinden)",
  ylabel="Olin'deki memnun olmayan müşteri sayısı")
```

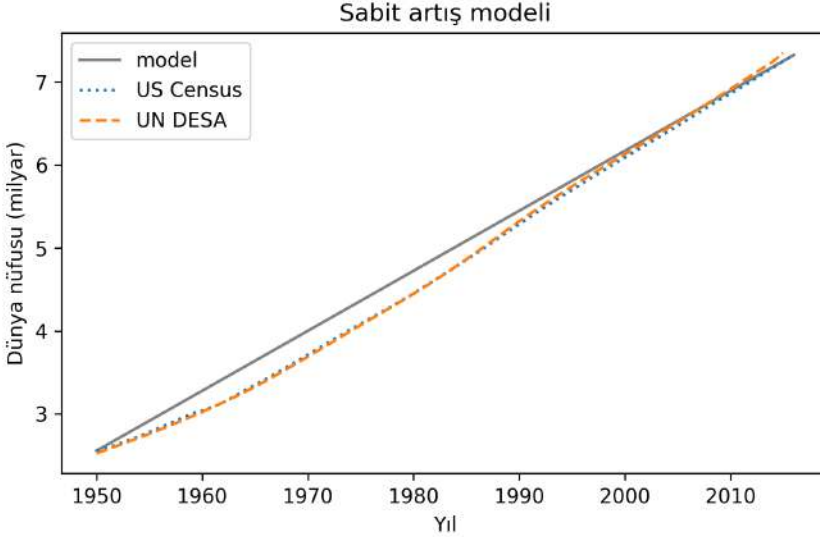


`color = 'C'` argümanı çizginin rengini belirtir (bu kitabın basılı versiyonunda bunu göremeyeceksiniz). Şimdiye dek çizdirdiğimiz `TimeSeries`, mavi, yani `C0` varsayılan rengini kullanmaktadır. `TimeSeries` olmadığını bize hatırlatması için `SweepSeries` için farklı bir renk kullanıyorum.

Olin'deki varış hızı düşükken bol miktarda bisiklet mevcuttur ve memnun olmayan müşteri yoktur. Varış hızı arttıkça bisikletlerin tükenmesi daha muhtemel olur ve memnun olmayan müşteri sayısı artar. Çizgi çentiklidir çünkü simülasyon rastgele sayılara dayanmaktadır. Bazen şanslıyızdır ve memnun olmayan müşteri sayısı nispeten azdır. Öteki zamanlar ise şanssızızdır ve memnun olmayan müşteri sayısı fazladır.

4.4 Artımlı Geliştirme

Birkaç satırdan daha uzun olan programlar yazmaya başladıkça kendinizi hata ayıklamaya daha fazla vakit harcarken bulabilirsiniz. Hata ayıklamaya başlamadan önce ne kadar çok kod yazarsanız, problemi bulmak o kadar zor olur.



Fonksiyonları ve System nesnelerini kullanmanın büyük bir ilerleme olduğu aşikâr olmayabilir ve sadece bir kere çalıştırdığımız basit bir model için belki de bu, büyük bir ilerleme olmayabilir. Ancak daha karmaşık modellerle çalıştıkça ve farklı parametrelerle birçok simülasyon gerçekleştirdiğimiz zaman, kodu bu şekilde düzenlemenin büyük bir fark oluşturduğunu göreceğiz.

Şimdi modeli iyileştirip iyileştiremeyeceğimizi görelim.

6.2 Oransal Bir Artış Modeli

Sabit artış modeliyle ilgili en büyük problem anlamlı olmamasıdır. Dünya genelinde insanların nüfus artışını yıldan yıla sabit tutmak için nasıl anlaşabileceklerini hayal etmek zordur. Her yıl nüfusun bir kısmı ölürken ve bir kısmı doğururken nüfustaki net değişimi şu şekilde hesaplayabiliriz:

```
def run_simulation2(system):
    results = TimeSeries()
    results[system.t_0] = system.p_0

    for t in range(system.t_0, system.t_end):
        births = system.birth_rate * results[t]
        deaths = system.death_rate * results[t]
        results[t+1] = results[t] + births - deaths

    return results
```

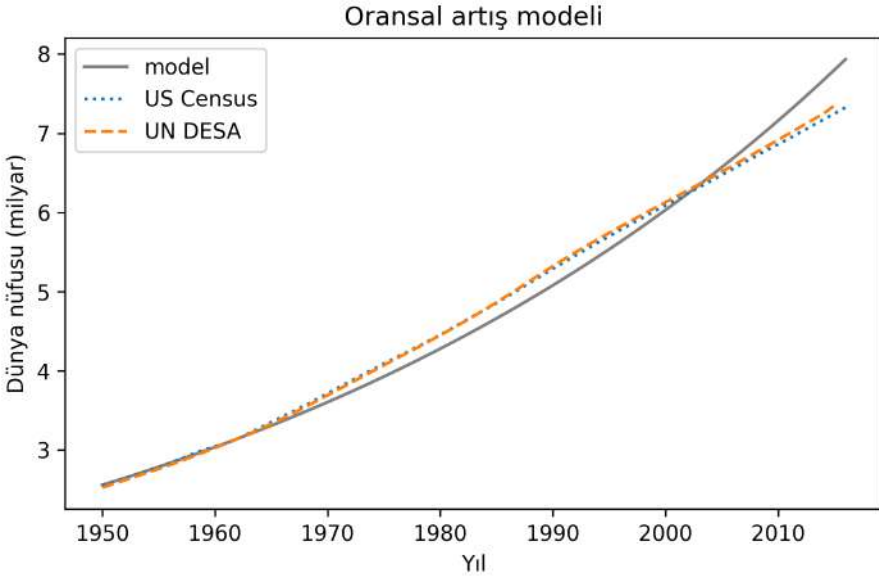
Döngü boyunca her seferinde doğum sayısını hesaplamak için `birth_rate` parametresini, ölüm sayısını hesaplamak için `death_rate` parametresini kullanıyoruz. Fonksiyonun geri kalanı `run_simulation1` ile aynıdır.

Artık `birth_rate` ve `death_rate` parametrelerinin verilere en iyi uyan değerlerini seçebiliriz. Ölüm oranı için kabaca 2020'deki küresel ölüm oranı olan 1000 kişide 7.7 ölümü kullanacağız (https://www.indexmundi.com/world/death_rate.html). Nüfus verilerine uymak için doğum oranını elle seçtim.

```
system.death_rate = 7.7 / 1000
system.birth_rate = 25 / 1000
```

Simülasyonu çalıştırıp, sonuçları çizdirdiğimizde şunları elde ederiz:

```
results2 = run_simulation2(system)
results2.plot(label='model', color='gray')
plot_estimates()
decorate(title='Oransal artış modeli')
```



Oransal model 1950'den 1965'e kadar verilere iyi uymaktadır fakat 1965'ten sonra o kadar da iyi uymamaktadır. Genel olarak, uymanın kalitesi sabit artış modeli kadar iyi değildir ve bu, şaşırtıcıdır çünkü oransal model daha gerçekçi gibi görünmektedir.

Bir sonraki bölümde anlamlı olan ve verilere uyan bir model bulmak için bir deneme daha yapacağız. Ancak öncelikle koda birkaç iyileştirme daha yapalım.

6.3 Güncelleme Fonksiyonunu Hesaba Katmamak

Bir sonraki yıl için nüfusu hesapladığımız `for` döngüsünün gövdesi hariç `run_simulation1` ve `run_simulation2` neredeyse özdeştir. Özdeş kodu tekrarlamaktan ziyade, değişen şeyleri değişmeyenlerden ayırabiliriz. Öncelikle doğumları ve ölümleri `run_simulation2` fonksiyonundan çıkaracağız ve bir fonksiyon oluşturacağız:

```
def growth_func1(t, pop, system):
    births = system.birth_rate * pop
    deaths = system.death_rate * pop
    return births - deaths
```

`growth_func1` argüman olarak içerisinde bulunan yılı, mevcut nüfusu ve bir `System` nesnesini alır ve içerisinde bulunan yıl boyunca oluşan net nüfus artışını döndürür.

Bu fonksiyon `t`'yi kullanmaz, dolayısıyla `t`'yi dışarıda bırakabiliriz. Ancak, `t`'nin gerekli olduğu başka artış fonksiyonları da göreceğiz ve kullanılsın veya kullanılmasın bu fonksiyonların hepsinin aynı parametreleri alması uygundur. Artık herhangi bir modeli çalıştıran bir fonksiyon yazabiliriz:

```
def run_simulation(system, growth_func):
    results = TimeSeries()
    results[system.t_0] = system.p_0

    for t in range(system.t_0, system.t_end):
        growth = growth_func(t, results[t], system)
        results[t+1] = results[t] + growth

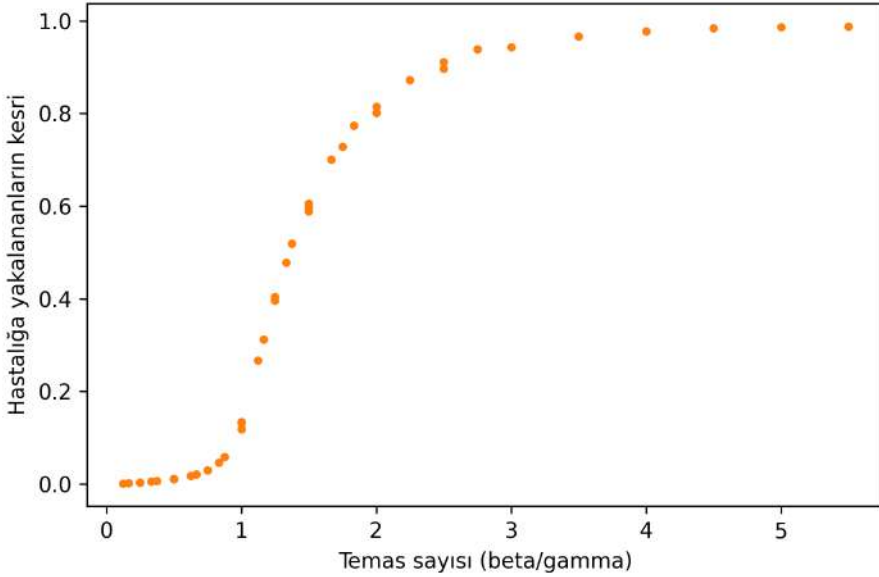
    return results
```

Bu fonksiyon daha önceleri görmediğimiz bir özelliği tanıtmaktadır: Bu fonksiyon bir fonksiyonu parametre olarak almaktadır! `run_simulation` fonksiyonunu çağırdığımızda, ikinci parametre bir sonraki yıl için nüfusu hesaplayan `growth_func1` gibi bir fonksiyondur.

Aşağıda bu fonksiyonu nasıl çağıracağımız gösterilmiştir:

```
results = run_simulation(system, growth_func1)
```

Bir fonksiyonu argüman olarak göndermek başka bir parametreyi göndermekle aynı şeydir. Bu örnekte `growth_func1` olan argüman `growth_func` olarak adlandırılan parametreye atanır. `run_simulation` fonksiyonunun içerisinde `growth_func` fonksiyonunu öteki fonksiyonlar gibi çağırabiliriz.



Sonuçlar en azında yaklaşık olarak tek bir eğri üzerine düşmektedir. Bu ise popülasyonun hastalığa yakalanacak kesrini tek bir parametreye, yani β/γ oranına bağlı olarak tahmin edebileceğimiz anlamına gelir. β ve γ 'nın değerlerini ayı ayrı bilmemize gerek yoktur.

14.3 Temas Sayısı

Bölüm 11'den, belirli bir gündeki yeni enfeksiyon sayısının βsiN ve iyileşme sayısının γiN olduğunu hatırlayalım. Bu nicelikleri bölersek sonuç $\beta s/\gamma$ olur. Bu ise iyileşme başına düşen yeni enfeksiyon sayısıdır (popülasyonun bir kesri olarak).

Savunmasız bir popülasyona yeni bir hastalık girdiği zaman s yaklaşık olarak 1'dir. Dolayısıyla her bir hasta kişi yoluyla hastalığa yakalanan insan sayısı β/γ olur. Bu oran temas sayısı veya temel çoğalma sayısı olarak adlandırılır. Geleneksel olarak bu sayı R_0 ile gösterilir fakat SIR modeli bağlamında bu gösterim kafa karıştırıcıdır. Bu yüzden R_0 yerine c kullanacağız.

Bir önceki kısımdaki sonuçlar c ile toplam enfeksiyon sayısı arasında bir ilişki olduğunu öne sürmektedir. Bu ilişkiyi Bölüm 11'deki diferansiyel denklemleri analiz ederek türetebiliriz:

$$\begin{aligned}\frac{ds}{dt} &= -\beta si \\ \frac{di}{dt} &= \beta si - \gamma i \\ \frac{dr}{dt} &= \gamma i\end{aligned}$$

Boyutsuz bir nicelik olan c 'yi elde etmek için temas oranını enfeksiyon oranına bölmüştük. Aynı şekilde oranların oranını elde etmek için di/dt 'yi ds/dt 'ye böleceğiz:

$$\frac{di}{ds} = \frac{\beta si - \gamma i}{-\beta si}$$

Bunu aşağıdaki gibi basitleştirebiliriz:

$$\frac{di}{ds} = -1 + \frac{\gamma}{\beta s}$$

β/γ yerine c koyarak aşağıdakini yazabiliriz:

$$\frac{di}{ds} = -1 + \frac{1}{cs}$$

Bir diferansiyel denklemi bir başka diferansiyel denkleme bölmek alışılmış bir durum değildir fakat burada kullanışlıdır. Çünkü i , s ve c arasında zamana bağlı olmayan bir ilişki verir bize. Bu ilişkiden yola çıkarak, c 'yi s 'nin son değeriyle ilişkilendiren bir denklem türetebiliriz. Teoride, bu denklem bir salgının gidişatını gözlemlemek suretiyle c 'yi çıkarsamayı mümkün kılar.

Aşağıda bu türetme işinin nasıl yürüdüğü görülmektedir. Bir önceki denklemin her iki tarafını ds ile çarpalım:

$$di = \left(-1 + \frac{1}{cs}\right) ds$$

ve ardından, her iki tarafın integralini alırız:

$$i = -s + \frac{1}{c} \log s + q$$

Burada q bir integral sabitidir. Terimleri yeniden düzenlersek:

$$q = i + s - \frac{1}{c} \log s$$

Şimdi q 'nın ne olduğunu bulup bulamayacağımıza bakalım. Bir salgının başlangıcında hastalığa yakalananların kesri küçükse ve neredeyse herkes savunmasızsa, q 'yu hesaplamak için $i(0) = 0$ ve $s(0) = 1$ yaklaşımlarını kullanabiliriz:

$$q = 0 + 1 + \frac{1}{c} \log 1$$

$\log 1 = 0$ olduğundan, $q = 1$ 'i elde ederiz.

Şimdi, salgının sonunda $i(\infty) = 0$ ve $s(\infty)$ 'un bilinmeyen bir nicelik olan s_∞ olduğunu varsayalım. Şimdi aşağıdakine sahibiz:

$$q = 1 = 0 + s_\infty - \frac{1}{c} \log s_\infty$$

c için çözerek aşağıdakini elde ederiz:

$$c = \frac{\log s_\infty}{s_\infty - 1}$$

c ile s_∞ 'u ilişkilendirmek suretiyle bu denklem, verilere dayalı olarak c 'yi kestirmeyi ve muhtemelen gelecekteki salgınların davranışını tahmin etmeyi mümkün kılar.

14.4 Analiz ile Simülasyonu Karşılaştırmak

Bu analitik sonucu simülasyon sonuçlarıyla karşılaştıralım. s_∞ için bir değerler dizisi oluşturacağız:

```
s_inf_array = linspace(0.003, 0.99, 50)
```

ve c 'nin ilgili değerlerini hesaplayacağız:

```
from numpy import log
```

```
c_array = log(s_inf_array) / (s_inf_array - 1)
```

Hastalığa yakalananların toplamını elde etmek için $s(0)$ ve $s(\infty)$ arasındaki farkı hesaplıyoruz ve ardından, sonuçları bir `Series` içerisine depoluyoruz:

```
frac_infected = 1 - s_inf_array
```

ModSim kütüphanesi, `c_array` ve `frac_infected`'i bir pandas `Series` içerisine koymak için kullanabileceğimiz `make_series` isimli bir fonksiyon sağlar:

```
frac_infected_series = make_series(c_array, frac_infected)
```

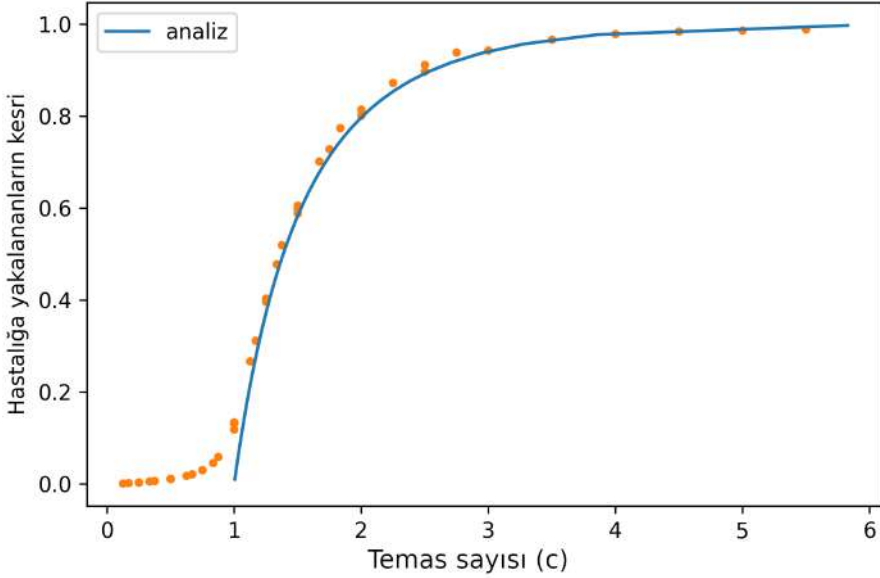
Artık sonuçları çizdirebiliriz:

```

plot_sweep_frame(frame)
frac_infected_series.plot(label='analiz')

decorate(xlabel='Temas sayısı (c)',
          ylabel='Hastalığa yakalananların kesri')

```



Temas sayısı 1'i aştığında analiz ve simülasyon uyumludur. Temas sayısı 1'den küçük olduğunda ise uyumlu değildirler: Analiz sonuçları hiçbir enfeksiyonun olmaması gerektiğini söylerken, simülasyonlarda az sayıda enfeksiyon vardır.

Farklılığın nedeni, simülasyonun zamanı ayrık gün serilerine bölmesi, buna karşın analizin zamana sürekli bir nicelik olarak muamele etmesidir. Temas sayısı büyük olduğunda iki model uyumlu olur. Temas sayısı küçük olduğunda ise iki model birbirinden ayrışır.

14.5 Temas Sayısını Kestirmek

Bir önceki şekil, temas sayısını bildiğimizde, popülasyon içerisinde hastalığa yakalanacakların kesrini sadece birkaç aritmetik işlemle kestirebileceğimizi göstermektedir. Simülasyon gerçekleştirmemiz gerekmemektedir.

Şekli öteki türlü de okuyabiliriz. Geçmiş bir salgında popülasyonun hangi kesrinin etkilendiğini bilirse, temas sayısını kestirebiliriz. Daha sonra, örneğin

15.4 Newton'un Soğuma Yasası

Newton'un soğuma yasası, bir nesnenin sıcaklığındaki değişim oranının, nesne ile nesneyi çevreleyen ortam arasındaki sıcaklık farkıyla doğru orantılı olduğunu ileri sürer:

$$\frac{dT}{dt} = -r(T - T_{env})$$

Burada t zaman, T nesnenin sıcaklığı, T_{env} ortamın sıcaklığı ve r ise ısının nesne ile ortam arasında ne kadar hızlı aktarıldığını belirten bir sabittir.

Newton'un "yasası" gerçekten bir modeldir: Bazı koşullarda iyi bir yaklaşım iken, öteki koşullarda o kadar iyi değildir. Örneğin ısı transferinin birincil mekanizması iletim ise Newton'un yasası "doğrudur". Yani r geniş bir sıcaklık aralığında sabittir. Ve bazı zamanlar malzemenin özellikleri ve nesnenin şekline dayalı olarak r 'yi kestirebiliriz.

Konveksiyon, ısı transferinin ihmal edilemez bir kısmına katkıda bulunduğu, r sıcaklığa bağlı olur fakat Newton'un yasası en azından dar bir sıcaklık aralığında çoğu zaman yeterince doğrudur. Bu durumda r ; yüzey şekli, hava akışı, buharlaşma ve diğer unsurlara bağlı olduğundan, deneysel olarak kestirilmelidir.

Radyasyon, ısı transferinin önemli bir kısmını oluşturduğunda, Newton'un yasası hiç iyi bir model değildir. Uzaydaki veya vakumdaki nesnelere veya yüksek sıcaklıktaki (örneğin birkaç yüz santigrat dereceden daha sıcak) nesnelere için bu durum geçerlidir. Ancak kahvenin soğuması gibi bir durumda Newton'un modelinin oldukça iyi olmasını bekleriz.

Bununla birlikte vereceğimiz sadece bir modelleme kararı bulunur: Kahve ve bardağa ayrı nesnelere olarak mı, yoksa tek bir nesne olarak mı muamele edeceğiz. Bardak kâğıttan yapılmışsa kütlesi kahveden azdır ve özgül ısı kapasitesi de düşüktür. Bu durumda bardağa ve kahveye tek bir nesne olarak muamele etmek mantıklıdır. Seramik kupa gibi önemli bir termal kütleyle sahip bir bardak söz konusu olduğunda, kahve ve bardağın sıcaklığını ayrı ayrı hesaplayan bir model düşünebiliriz.

15.5 Newton Soğumasını Gerçeklemek

Başlamak için kahveye odaklanacağız. Ardından, bir alıştırma olarak sütü simüle edebilirsiniz. Bir sonraki bölümde kahveyi ve sütü kelimenin tam anlamıyla bir araya getireceğiz.

Aşağıda bir sistemin parametrelerini alan ve bir `System` nesnesi oluşturan fonksiyon görülmektedir:

```
def make_system(T_init, volume, r, t_end):
    return System(T_init=T_init,
                  T_final=T_init,
                  volume=volume,
                  r=r,
                  t_end=t_end,
                  T_env=22,
                  t_0=0,
                  dt=1)
```

Parametrelere ilave olarak `make_system`, `T_env` ortam sıcaklığını, `t_0` başlangıç zaman damgasını ve soğuma sürecini simüle etmek için kullanacağımız `dt` zaman adımını belirler.

Aşağıda kahveyi temsil eden bir `System` nesnesi görülmektedir:

```
coffee = make_system(T_init=90, volume=300, r=0.01, t_end=30)
```

`T_init`, `volume` ve `t_end` değerleri problemin ifade edilmişinden gelmektedir. `r`'nin değerini şimdilik keyfi olarak seçiyorum. `r`'nin nasıl kestirileceğini yakında göreceğiz.

Aslına bakılırsa Newton'un yasası bir diferansiyel denklemdir fakat kısa bir zaman aralığında bu denkleme bir fark denklemiyle yaklaşabiliriz:

$$\Delta T = -r(T - T_{env})dt$$

Burada dt zaman adımı ve ΔT ise bu zaman adımı boyunca sıcaklıkta meydana gelen değişiktir.

Not

ΔT 'yi sıcaklıkta zamanla meydana gelen değişimi belirtmek için kullanıyorum. Fakat ısı transferi bağlamında ΔT 'nin bir nesne ve bu nesnenin içerisinde bulunduğu ortam arasındaki ΔT sıcaklık farkını belirtmek için kullanıldığını da görebilirsiniz. Kafa karışıklığımı en aza indirmek için ikinci kullanımdan kaçınıyorum.

Aşağıdaki fonksiyon mevcut zaman olan t 'yi, mevcut sıcaklık olan T 'yi ve bir `System` nesnesini alır ve bir zaman adımı boyunca sıcaklıkta meydana gelen değişimi hesaplar:

```
def change_func(t, T, system):
    r, T_env, dt = system.r, system.T_env, system.dt
    return -r * (T - T_env) * dt
```

Bu fonksiyonu kahvenin başlangıç sıcaklığıyla şu şekilde test edebiliriz:

```
change_func(0, coffee.T_init, coffee)
```

```
-0.68
```

`dt = 1` dakika olduğunda, sıcaklık, `r`'nin en azından bu değeri için yaklaşık 0.7°C düşer.

Şimdi ise aşağıda `run_simulation` fonksiyonunun `t_0`'dan `t_end`'e kadar olan bir zaman serisini simüle eden versiyonu bulunmaktadır:

```
def run_simulation(system, change_func):
    t_array = linrange(system.t_0, system.t_end, system.dt)
    n = len(t_array)

    series = TimeSeries(index=t_array)
    series.iloc[0] = system.T_init

    for i in range(n-1):
        t = t_array[i]
        T = series.iloc[i]
        series.iloc[i+1] = T + change_func(t, T, system)

    system.T_final = series.iloc[-1]
    return series
```

Burada `run_simulation` fonksiyonunun önceki versiyonlarından farklı olan iki şey vardır.

İlk olarak, `dt` zaman adımıyla birlikte `t_0`'dan `t_end`'e kadar olan bir değerler dizisi oluşturmak için `linrange` fonksiyonunu kullanıyoruz. `linrange` fonksiyonu `linspace` fonksiyonuna benzerdir. Her ikisi de bir başlangıç ve bir bitiş değeri alır ve eşit aralıklı bir değerler dizisi döndürür. Farkları ise üçüncü argümandadır: `linspace` aralıktaki nokta sayısını belirten bir tam sayı alırken, `linrange` değerler arasındaki aralığı belirten bir adım büyüklüğü alır. `TimeSeries` oluştururken, `TimeSeries`'in indeksinin zaman damgaları dizisi olan `t_array` olduğunu belirtmek için anahtar kelime argümanı `index`'i kullanırız.

İkinci olarak, `run_simulation` fonksiyonunun bu versiyonu `TimeSeries`'teki satırları belirtmek için `loc`'dan ziyade `iloc` kullanır. Fark şudur:

- `loc` ile birlikte köşeli parantezler içerisindeki etiket herhangi türde bir değer olabilir. Yani herhangi bir başlangıç, bitiş ve zaman adımı. Örneğin

dünya nüfusu modelinde etiketler 1950 ile başlayan ve 2016 ile biten yıllardır.

- `iloc` ile birlikte köşeli parantezler içerisindeki etiket her zaman 0 ile başlayan bir tam sayıdır. Dolayısıyla etiketlerin ne olduğuna bakmaksızın, her zaman ilk elemanı `iloc[0]` ve son elemanı `iloc[-1]` ile elde edebiliriz.

`run_simulation` fonksiyonunun bu versiyonunda döngü değişkeni tam sayı olan `i`'dir ve 0 dâhil, `n-1` dâhil olmamak üzere 0'dan `n-1`'e kadar gitmektedir. Dolayısıyla döngüde ilk seferde `i`, 0'dır ve `TimeSeries`'e eklediğimiz değerin indeksi 1'dir. Döngüde son seferde `i`, `n-2`'dir ve eklediğimiz değerin indeksi ise `n-1`'dir.

Simülasyonu şu şekilde çalıştırabiliriz:

```
results = run_simulation(coffee, change_func)
```

Sonuç, zaman adımı başına bir satırla birlikte bir `TimeSeries`'tir. İşte ilk birkaç satır:

```
show(results.head())
```

Time	Quantity
0.0	90.000000
1.0	89.320000
2.0	88.646800
3.0	87.980332
4.0	87.320529

Ve son birkaç satır:

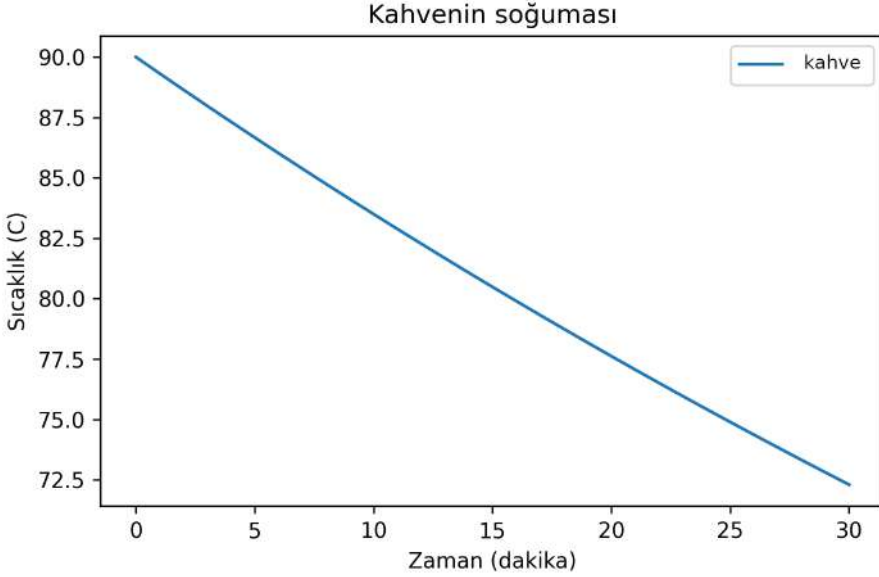
```
show(results.tail())
```

Time	Quantity
26.0	74.362934
27.0	73.839305
28.0	73.320912
29.0	72.807702
30.0	72.299625

`t_0=0`, `t_end=30` ve `dt=1` olmak üzere, zaman damgası 0.0'dan 30.0'a kadar gider.

TimeSeries aşağıdaki gibidir:

```
results.plot(label='kahve')
decorate(xlabel='Zaman (dakika)',
         ylabel='Sıcaklık (C)',
         title='Kahvenin soğuması')
```



30 dakika sonra sıcaklık 72.3°C 'dir. Bu ise eşleştirmeye çalıştığımız ölçüm olan 70°C 'den biraz yüksektir:

```
coffee.T_final
```

72.2996253904031

Son sıcaklığın tam olarak 70°C olduğu r değerini bulmak için deneme yanılma yöntemiyle hareket edebiliriz fakat kök bulma algoritmasını kullanmak daha verimlidir.

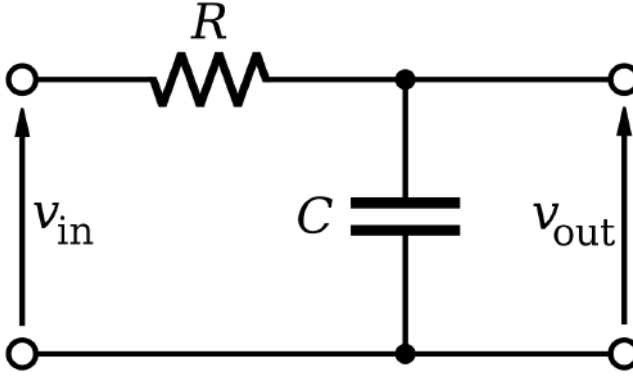
15.6 Kökleri Bulmak

ModSim kütüphanesi doğrusal olmayan denklemlerin köklerini bulan `root_scalars` isimli bir fonksiyon sağlar. Örneğin aşağıdaki polinomun köklerini bulmak istediğinizi farz ediniz:

defterini insülin modelini gerçeklemek, verilere en iyi uyan parametreleri bulmak ve ϕ_1 ve ϕ_2 'yi kestirmek için kullanınız. Bu not defterini <https://github.com/AllenDowney/ModSimPy/raw/master/examples/insulin.ipynb> adresinden indirebilir veya <https://colab.research.google.com/github/AllenDowney/ModSimPy/blob/master/examples/insulin.ipynb> adresinde Colab üzerinde çalıştırabilirsiniz.

19.3 Alçak Geçiren Filtre

Şekil 19.1'deki devre şeması bir direnç ve bir kondansatörle oluşturulan alçak geçiren filtreyi göstermektedir.



Şekil 19.1: Bir RC filtresinin devre şeması

Filtre denilen şey, V_{in} gibi bir sinyali giriş olarak alan ve V_{out} gibi bir sinyali çıkış olarak veren bir devredir. Bu bağlamda sinyal denilen şey, zamanla değişen bir gerilimdir (voltajdır).

Bir filtre, düşük frekanslı bir sinyalin V_{in} 'den V_{out} 'a değişmeden geçmesine izin veriyorsa fakat yüksek frekanslı sinyallerin genliğini azaltıyorsa alçak geçiren filtredir.

Devre analizi yasalarını uygulayarak, sistemin davranışını açıklayan bir diferansiyel denklem türetebiliriz. Diferansiyel denklemi çözmek suretiyle bu devrenin herhangi bir giriş sinyali üzerindeki etkisini tahmin edebiliriz.

Belirli bir andaki V_{in} ve V_{out} 'un verilmiş olduğunu farz ediniz. Dirençlerin davranışının basit bir modeli olan Ohm yasasına göre dirençten geçen anlık akım:

$$I_R = (V_{in} - V_{out})/R$$

Burada R ohm (Ω) cinsinden dirençtir.

Devrenin çıkışından hiçbir akımın akmadığı varsayılarak, Kirchhoff'un akım yasasına göre kondansatörden geçen akım şu şekilde ifade edilir:

$$I_C = I_R$$

Kondansatörlerin davranışının basit modeline göre kondansatörden geçen akım, kondansatörün uçları arasındaki gerilimde bir değişikliğe neden olur:

$$I_C = C \frac{dV_{out}}{dt}$$

Burada C , farad (F) cinsinden kapasitanstır. Bu denklemleri birleştirmek, V_{out} için aşağıdaki diferansiyel denklemi verir:

$$\frac{dV_{out}}{dt} = \frac{V_{in} - V_{out}}{RC}$$

Bu kitabın GitHub deposunda bu örnek durum incelemesi için başlangıç kodunu içeren `filter.ipynb` isimli bir not defteri bulacaksınız. Bu not defterini <https://github.com/AllenDowney/ModSimPy/raw/master/examples/filter.ipynb> adresinden indirebilir veya <https://colab.research.google.com/github/AllenDowney/ModSimPy/blob/master/examples/filter.ipynb> adresinde Colab üzerinde çalıştırabilirsiniz. Aşağıdakine benzer giriş sinyalleri için alçak geçiren filtreyi simüle etmek için talimatları takip ediniz:

$$V_{in}(t) = A \cos(2\pi ft)$$

Burada A , girdi sinyalinin genliğidir. Örneğin 5 V ve f , sinyalin hertz (Hz) cinsinden frekansdır.

19.4 Bir Duvarın Termal Davranışı

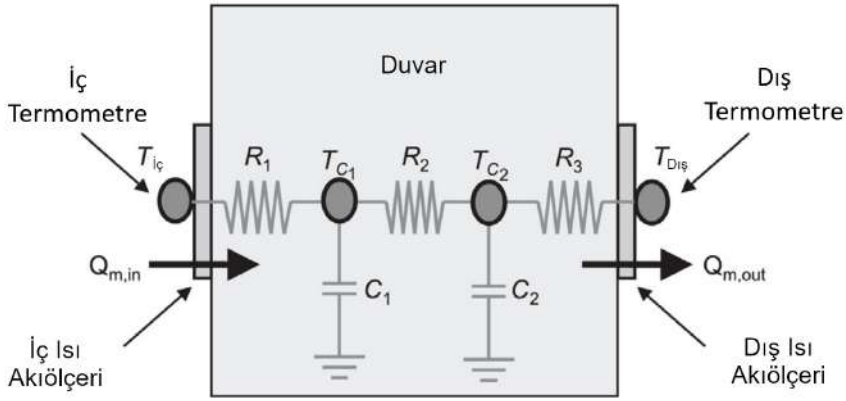
Bu örnek durum incelemesi, Gori vd.'nin, “binaların beklenen enerji kullanımları ile ölçülen enerji kullanımları arasındaki performans farkını” anlama amacıyla bir tuğla duvarın termal davranışını modelleyen makalesine dayanmaktadır.

Şekil 19.2 senaryoyu ve Gori vd.'nin duvar modelini göstermektedir.

Duvarın iç ve dış yüzeylerindeki sıcaklığı ve ısı akısını (ısı akışı oranı) üç gün boyunca ölçerler. Yüzeyle ve birbirine termal dirençlerle bağlanan iki termal kütleyle kullanarak duvarı modellerler.

Makalenin birincil metodolojisi sistemin parametrelerini çıkarsamaya yönelik istatistiksel bir yöntemdir (iki termal kütle ve üç ısıl direnç).

Birincil sonuç iki modelin karşılaştırılmasıdır: Burada gösterilen iki termal kütleyle sahip model ve sadece bir termal kütleyle sahip daha basit olan model. İki kütleli modelin ölçülmüş akıları önemli ölçüde daha iyi üretebildiğini buldular.



Şekil 19.2: Bir duvarın termal davranışının modeli (Gori vd., 2017)

Bu örnek durum incelemesi için onların modelini gerçekleyecek ve makaledeki kestirilen parametrelerle modeli çalıştıracamız ve ardından, daha az hata veren parametreler bulup bulamayacağımızı görmek için `leastsq` fonksiyonunu kullanacağız.

Bu kitabın GitHub deposunda bu örnek durum incelemesine yönelik kod ve sonuçların olduğu `wall.ipynb` isimli bir not defteri bulacaksınız. Bu not defterini <https://github.com/AllenDowney/ModSimPy/raw/master/examples/wall.ipynb> adresinden indirebilir veya <https://colab.research.google.com/github/AllenDowney/ModSimPy/blob/master/examples/wall.ipynb> adresinde Colab üzerinde çalıştırabilirsiniz.

Bu örnek durum incelemesi Gori, Marincioni, Biddulph ve Elwell tarafından 2017 yılında *Energy and Buildings* dergisinde yayımlanan “Inferring the Thermal Resistance and Effective Thermal Mass Distribution of a Wall from In Situ Measurements to Characterise Heat Transfer at Both the Interior and Exterior Surfaces” isimli makaleye dayanmaktadır (makale, <https://www.sciencedirect.com/science/article/pii/S0378778816313056> adresinde mevcuttur). Yazarlar makalelerini Creative Commons lisansı ile çıkarmış ve verilerini <https://discovery.ucl.ac.uk/id/eprint/1526521> adresinde kullanıma sunmuştur. Onlara bu örnek durum incelemesini mümkün kılan açık, tekrarlanabilir bilime olan bağlılıkları için teşekkür ediyorum.

Dizin

- açı, 208
- açıklama, 32
- açısal hız, 231
- açısal ivme, 241, 244, 260
- adaptif zaman adımı, 266
- ağaç büyümesi, 102
- akışlar, 108
- alçak geçiren filtre, 183
- altın bölümlleme yöntemi, 269
- Amerika Birleşik Devletleri Nüfus Sayım İdaresi, 49
- Amerikan Atlantik Somon Değerlendirme Komitesi, 101
- Amerikan Doğa Tarihi Müzesi, 47
- analiz, matematiksel, 3, 11, 90, 134, 158
- Andrew Phillips, 186
- animasyon, 217, 252
- append fonksiyonu, 247
- argüman, 22
- artırmalı geliştirme, 43, 101
- artış oranı, 72
- aşı, 117
- atama operatörü, 33
- atama, çoklu, 110
- attribute, 26
- ayrık, 137, 158
- ayrık model, 89

- bağışıklanma, 121
- bağışıklık cevabı, 186
- barometrik basınç, 199
- başlangıç değeri problemi, 175
- beyzbol, 14, 205, 210
- Beyzbol fiziği, 205, 210, 219
- bilimsel gösterim, 49
- Bir Astronoma Sor, 195
- bir fonksiyonun genelleştirilmesi, 22
- birim vektör, 209

- birimler, 8, 9, 142
- birinci dereceden diferansiyel denklem, 189
- Birinci Sınıf Öğrencisi Salgını, 105
- bisiklet paylaşım modeli, 29
- bisiklet paylaşım sistemi, 15
- Boole değeri, 20
- Boston Red Sox, 221
- Boston, Massachusetts, 222
- boyutsuz, 131, 198
- boyutsuzlaştırma, 132
- Brent'in yöntemi, 268
- bungee atlaması, 255
- büyüklik, 10, 206–208, 241

- crossings fonksiyonu, 193

- çap, 199
- çarpma işlemi, 7
- çerçeve, modelleme, 3, 11
- çevre, 230
- çoklu atama, 110
- DataFrame nesnesi, 48
- Dave Rothstein, 195
- David Smith, 105
- decorate fonksiyonu, 25
- değer, 7
 - Boolean, 20
 - NaN, 49
 - None, 74
 - return, 37
- değişken, 7
 - döngü, 23
 - durum, 16, 192, 231, 244
- denge, 71
- denklem
 - diferansiyel, 89, 107, 134, 145, 175, 182, 183, 189, 231
 - fark, 87, 145, 175

- denklemler sistemi, 260
- Denver, Colorado, 219
- derece, 208
- destek öğeleri, 43
- deterministik model, 34
- devre şeması, 183
- deyimler
 - if, 20
 - print, 18
 - return, 33, 37, 74
- diferansiyel denklem, 89, 107, 134, 145, 175, 182, 183, 189, 231
- DimensionalityError, 13
- diyabet, 162
- diyagram, stok ve akış, 108
- dizgi, 19
- dizi, 40, 146, 158
- DNA, 163
- docstring, 32
- doğrulama, 5, 11, 159, 175
 - dahili, 5
 - harici, 5
- doğrusal artış, 97
- doğrusal ilişki, 67
- doğrusal interpolasyon, 166
- dokümantasyon, 30
- Dormand-Prince yöntemi, 263
- döndürülen değer, 37
- döner tabla, 242
- döngü değişkeni, 23
- dönme hareketi, 229
- durum değişkeni, 16, 192, 231, 244
- dünya, 257
- dünya nüfusu, 47, 99
- düşen peni efsanesi, 191

- else koşulu, 20
- Empire State binası, 191, 257
- enerjinin korunumu, 154
- eşittir operatörü, 33
- etiket, 24
- Euler yöntemi, 175
- event fonksiyonu, 194, 213, 233, 246
- Everest Dağı, 143
- evrensel kütle çekim yasası, 195
- eylemsizlik momenti, 242

- f-string, 124
- fark denklemi, 87, 145, 175
- farmakokinetik, 161

- Fenway Park, 221
- fırlatma açısı, 223
- filtre, 183
- fiziksel sistem, 4
- flip fonksiyonu, 20, 27
- fonksiyon
 - append, 247
 - argüman olarak, 64
 - crossings, 193
 - çağırarak, 18
 - decorate, 25
 - dönüş değeri olarak, 166
 - event, 194, 213, 233, 246
 - flip, 20, 27
 - genelleştirilmesi, 22
 - gövdesi, 17
 - hata, 151, 250
 - head, 49
 - interpolate, 166
 - kökü, 148
 - leastsq, 181
 - linspace, 39
 - make_series, 136
 - maximize_scalar, 223, 269
 - parametre olarak, 64, 148
 - plot, 25, 70
 - range, 23, 39
 - read_html, 48
 - root_scalar, 148, 244, 251, 268
 - run_solve_ivp, 175, 263
 - slope, 176, 192, 200, 210, 232
 - sqr, 8
 - State, 15
 - tail, 50
 - tanımlamak, 17
 - unimodal, 270
 - üstel, 90
 - Vector, 206
- fonksiyonun gövdesi, 17
- for loop, 22
- format belirtici, 124
- formatlanmış bir dizgi literalı, 124

- genel çözüm, 94
- George Box, 6, 29
- George Orwell, 29
- gerçeklemek, 108, 170
- gerilme, 260
- gezegen yörüngesi, 5, 257
- glikoz, 161

- gömülü kodlama, 54
göreceli hata, 52, 179
Green Monster, 222
güncelleme operatörü, 16
Güneş, 195, 257
- Hans Rosling, 82
hassasiyet, 8
hata ayıklama, 42
hata fonksiyonu, 151, 250
hatalar
 - DimensionalityError, 13
 - göreceli, 52
 - mutlak, 51
 - NameError, 74
 - sözdizimi, 12, 34
 - ValueError, 149
hava direnci, 6, 194, 197, 218
head fonksiyonu, 49
hız, 189, 198
hızlanma
 - yerçekimi, 6
hiperglisemi, 162
HIV, 186
homeostaz, 162
Hooke'un yasası, 256
- ısı, 142
ısı akışı, 184
ısıl direnç, 184
if deyimi, 20
iki parçaya bölme yöntemi, 268
ikinci dereceden diferansiyel denklem, 189
ikinci dereceden ilişki, 67
iletim, 143
iloc, 146
indeks (Series), 50
insülin, 162
 - minimal model, 182
integral sabiti, 90
interpolasyon, 165
interpolate fonksiyonu, 166
ivme, 190, 201, 212
 - yerçekimi kaynaklı, 191, 256
Jearl Walker, 141
jiroskopik presesyon, 229
joule, 142
- kahvenin soğuması problemi, 141
kan şekeri, 162
kan torbası modeli, 174
karantina, 122
kare kök, 8
karşılaştırma operatörleri, 34
kedi yavruları, 259
Kermack-McKendrick modeli, 106
Kirchhoff'un akım yasası, 184
kod bloğu, 17
kompartıman modelleri, 108
kontur grafiği, 129
konum, 189
konveksiyon, 143
kök (denklemin), 71, 148
köşeli parantez (kök olarak), 149, 251, 268
köşeli parantez operatörü, 54, 207
kuadratik model, 70, 88
kuvvet, 190, 211, 241
 - sürüklenme, 197, 201, 210, 218
kuvvet kolu, 241
kuyruklama teorisi, 100
kütle, 190, 242
kütüphaneler
 - Matplotlib, 27, 46, 70, 129, 252
 - ModSim, 15, 66, 123, 136, 139, 148, 175, 193, 206, 217, 223
 - NumPy, 8, 39, 208
 - pandas, 27, 46, 48, 139, 165
 - Pint, 9
 - SciPy, 167, 175, 182, 263, 268, 269
 - SymPy, 92, 261
 - types, 66
Lang Moore, 105
leastsq fonksiyonu, 181
linspace fonksiyonu, 39
loc, 114, 146
logaritma, 90, 157
lojistik artış, 97
- Magnus kuvveti, 205
make_series fonksiyonu, 136
Manny Ramirez, 221
Mars Climate Orbiter, 9
matematiksel gösterim, 90, 171
matematiksel sabit, pi, 12
Mathematica, 92
Matplotlib kütüphanesi, 27, 46, 70, 129, 252
maximize_scalar fonksiyonu, 223, 225, 269
menzil (yörünge), 223, 259

- metrik, 34, 117
- minimal model, 162, 182
- modelleme çerçevesi, 3, 11
- modelleme kararı, 144, 205
- modelleme, yinelemeli, 5, 29
- modeller, 4
 - ayrık, 89
 - bisiklet paylaşımı, 29
 - deterministik, 34
 - kan torbası, 174
 - Kermack-McKendrick, 106
 - kompartment, 108
 - kuadratik, 70, 88
 - minimal, 162, 182
 - sabit artış, 87
 - SIR, 106
 - stokastik, 34
 - sürekli, 89
 - uzamsal olmayan, 174
- ModSim kütüphanesi, 66, 123, 136, 139, 148, 175, 193, 206, 217, 223
- mutlak hata, 51
- MythBusters, 6, 198
- NameError**, 74
- NaN** değeri, 49
- nem, 199
- nesneler
 - DataFrame**, 48
 - maximize_scalar**, 225
 - OdeResult**, 176
 - Params**, 199
 - Series**, 27, 46
 - SimpleNamespace**, 66
 - State**, 15, 30
 - SweepFrame**, 127
 - SweepSeries**, 41
 - Symbol**, 92
 - System**, 59
 - TimeFrame**, 112
 - TimeSeries**, 23
 - UnitRegistry**, 9
 - Vector**, 206
- net artış (nüfus), 69
- Newton'un evrensel kütle çekim yasası, 5
- Newton'un ikinci hareket yasası, 189, 242
- Newton'un üçüncü hareket yasası, 197
- Newton'un Soğuma Yasası, 144
- nicelik, 10
 - vektör, 206, 242
- nokta operatörü, 16, 49, 206
- None** değeri, 74
- NumPy kütüphanesi, 8, 12, 39, 208
- nüfus, 47
- Nüfus Bombası, 47
- Nüfus Sayım İdaresi, 49
- Occam'ın usturası, 162
- OdeResult** nesnesi, 176
- Ohm kanunu, 183
- Olin Üniversitesi, 15, 105
- operatörler
 - atama, 33
 - eşittir, 33
 - güncelleme, 16
 - karşılaştırma, 34
 - köşeli parantez, 54, 207
 - nokta, 16, 49, 206
- optimizasyon, 225, 259, 268
- oransal model, 88
- Örümcek Adam, 257
- özel çözüm, 94
- özgül ısı kapasitesi, 142, 154
- pandas kütüphanesi, 27, 46, 48, 139, 165
- parabol, 193, 237
- Param nesnesi, 199
- parametre, 21, 30
 - fonksiyon olarak, 148
 - modele karşı fonksiyon, 38
 - serbest, 164
 - system, 59
 - taramak, 40, 120, 125, 157
- parametreleri taramak, 40, 120, 125, 157
- parametreleştirme, 72, 88
- Paul Erlich, 47
- peni efsanesi, 3, 5
- pi, 12
- Pint kütüphanesi, 9
- Pint niceliği, 11
- plot** fonksiyonu, 25, 70
- presesyon, 229
- print deyimi, 18
- Programlama dili, 91
- projeksiyon, 77
 - tahmine karşı, 77
- quiver grafiği, 267
- radyan, 208, 231
- radyasyon, 143

- range** fonksiyonu, 23, 39
 rastgele sayı üretici, 20, 27
read_html fonksiyonu, 48
 referans alanı, 198
 return deyimi, 33, 37, 74
root_scalar fonksiyonu, 148, 244, 251, 268
run_solve_ivp fonksiyonu, 175, 263
 Runge-Kutta yöntemi, 263
- saha indeks (ağaç büyümesi), 102
 Scientific American, 141
 SciPy kütüphanesi, 167, 175, 182, 263, 268, 269
 sekant yöntemi, 268
 serbest parametreler, 164
Series nesnesi, 27, 46, 50
 SI birimleri, 9, 142
 sıcaklık, 142, 184
 sıfır (denklemin), 71
 sıvıları karıştırmak, 153
SimpleNamespace nesnesi, 66
 simülasyon, 3, 11, 90, 136
 sinyal, 183
 SIR modeli, 106
 sistem durumu, 16
 sistem, fiziksel, 4
 sistemin durumu, 16
 slope fonksiyonu, 176, 192, 200, 210, 232
 somon nüfusu, 101
 soyutlama, 3, 11
 sözdizimi hatası, 12, 34
sqrt, 8
State Fonksiyonu, 15
State nesnesi, 15, 30
 stok ve akış diyagramı, 108
 stokastik model, 34
 stoklar, 108
 sürekli model, 89
 sürekli zaman, 158
 sürtünme, 242, 251, 259
 sürtünme katsayısı, 243
 sürü bağışıklığı, 121
 sürükleme denklemi, 197, 201
 sürükleme katsayısı, 198, 205
 sürükleme kuvveti, 197, 201, 210, 218
SweepFrame nesnesi, 127
SweepSeries nesnesi, 41
Symbol nesnesi, 92
 SymPy kütüphanesi, 92, 261
System nesnesi, 59
- system parametresi, 59
- tail** fonksiyonu, 50
 takip edilen yol, 258
 takip edilen yolun grafiği, 217
 taşıma kapasitesi, 72, 79
 temel çoğalma sayısı, 134
 tensör, 242
 termal kütle, 142, 154, 184
 termal sistemler, 141, 184
 terminal hızı, 6, 198
TimeFrame nesnesi, 112
TimeSeries nesnesi, 23
 tork, 241, 251, 260
 traceback (geri izleme), 13
 tuğla duvar, 184
 tuvalet kağıdı rulosu fiziği, 230, 259
 türev, 233
types kütüphanesi, 66
- UN DESA, 50
 unimodal fonksiyonu, 270
UnitRegistry nesnesi, 9
 uymanın kalitesi, 63
 uzamsal olmayan model, 174
- üs alma, 7
 üstel artış, 97
 üstel fonksiyon, 90
ValueError, 149
 varsayım, modelleme, 29
Vector fonksiyonu, 206
 vektör çarpımı, 241
 vektör nicelikler, 206, 242
 vektörün bileşenleri, 206
 Vikipedi, 47
- Wellesley Üniversitesi, 15
 WolframAlfa, 92
- yarış koşmak, 14
 yay sabiti, 256
 yer çekimi, 6, 191, 256, 260, 261
 yinelemeli modelleme, 5, 29
 yo-yo örneği, 260
 yoğunluk, 198
 yön (vektörün), 206, 241
 yörünge, 257
- zaman adımı, 21, 101, 145, 175, 179, 268
 zaman damgası, 23