درسنامه محاسبات عـددی

علی فهیم
دانشگاه تهران

بهار ۱۴۰۱

فهرست مطالب

فصل ۱

تحلیل خطاها

در این فصل با مفهوم خطا، انواع آن در محاسبات عددی و ریشه‌ی بوجود آمدن آنها آشنا می‌شویم. آشنا شدن با خطاها به ما کمک می‌کند تا بهتر بتوانیم آنها را در محاسبات خود کنترل و تحلیل نماییم.

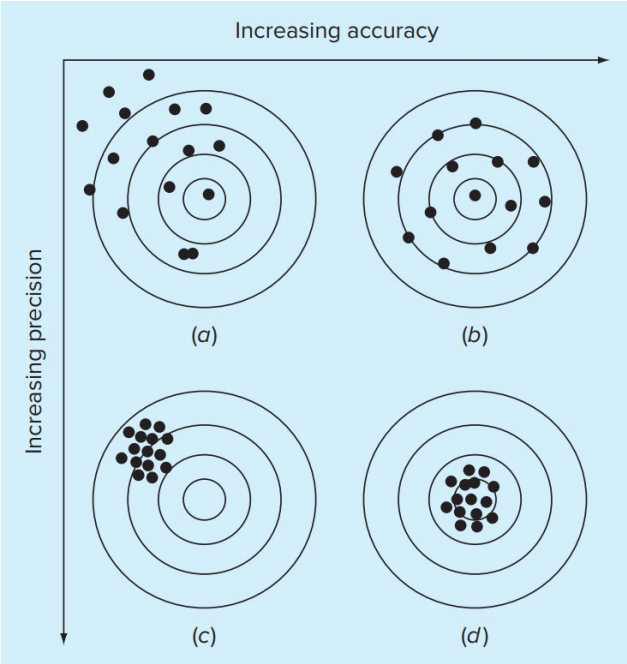
۱.۱ مفاهیم اولیه‌ی خطاها

ابتدا با مفهوم دقت (Precision) و درستی (Accuracy) آشنا می‌شویم. همانطور که در شکل ۱.۱ مشاهده می‌کنیم، میزان دقت در یک عملکرد به میزان تمرکز و یکسان بودن نتایج تکرارهای آن عملکرد باز می‌گردد در صورتی‌که میزان درستی آن عملکرد به نزدیک بودن میانگین نتیجه‌ی تکرارها به هدف نهایی آن عملکرد بستگی دارد. پس بطور مثال یک ابزار اندازه‌گیری دقیق و اما نادرست همواره با یک فاصله‌ای مشخص از اندازه‌ی واقعی، هرچند دور اندازه‌گیری می‌کند. این ابزار با کالیبراسیون قابل اصلاح می‌باشد. اما یک ابزار نادقیق ولی درست حتی با کالیبراسیون هم قابل استفاده نیست. عدم دقت (Imprecision) را گاهی عدم قطعیت (Uncertainty) و عدم درستی (Inaccuracy) را پیش‌قدر (Bias) نیز می‌نامیم.

در محاسبات عددی هدف اصلی تقریب زدن یک مقدار واقعی است. که این مقدار در عمل یا دست‌نیافتنی است و یا بسیار پرهزینه بدست می‌آید و ما در پی یافتن یک تقریب خوبی از این مقدار واقعی هستیم. این تقریب می‌بایست دارای دقت و درستی کافی باشد.

فرض کنید مقدار $\tilde{x}$ تقریبی برای مقدار دقیق x باشد ($x \approx \tilde{x}$). به عنوان تعریف، **خطای مطلق واقعی** (True Absolute Error) که با e_t نمایش می‌دهیم، از اختلاف مقدار تقریبی $\tilde{x}$ از مقدار واقعی x بدست می‌آید. از آنجایی که مقدار واقعی x برای ما مشخص نیست، خطای واقعی نیز مشخص نمی‌باشد. فقط می‌دانیم که تابع مقدار تقریبی $\tilde{x}$ و دارای حد بالای $e_{\tilde{x}}$ می‌باشد.

$$e_t(\tilde{x}) = |x - \tilde{x}| \leq e_{\tilde{x}}, \quad \rightarrow \quad \tilde{x} - e_{\tilde{x}} \leq x \leq \tilde{x} + e_{\tilde{x}} \tag{۱.۱}$$



شکل ۱.۱: مفهوم دقت و درستی (منبع: کتاب چاپرا)

پس مقدار واقعی که ناشناخته است، بین دو مقدار حدی قرار دارد و بنابر قرارداد، مقدار واقعی را به شکل زیر نمایش می‌دهیم.

$$x = \tilde{x} \pm e_{\tilde{x}} \tag{۲.۱}$$

حد بالای خطا ($e_{\tilde{x}}$) مقدار یکتایی ندارد و در روش‌های محاسباتی سعی می‌کنیم کوچکترین تخمین ممکن را بدست آوریم. مقدار خطای مطلق بسیار حساس و وابسته به مقدار واقعی است و در بسیاری از موارد محاسباتی شاید معیار صحیحی برای مقایسه تقریب‌های مختلف و تشخیص بهترین تقریب نباشد. لذا در مقایسه تقریب‌ها با یکدیگر از **خطای نسبی واقعی** (True Relative Error) استفاده می‌کنیم که نسبت خطای مطلق واقعی به مقدار واقعی است.

$$\epsilon_t = \frac{|x - \tilde{x}|}{|x|} \tag{۳.۱}$$

همانطور که مطرح شد، مقدار واقعی قاعدتا در دسترس نمی‌باشد و به همین دلیل است که خود را محدود به مقدار تقریبی کرده‌ایم. در نتیجه در مقابل خطای مطلق و نسبی دقیق، **خطای مطلق و نسبی تقریبی** هم وجود دارد که در آن‌ها خطا اختلاف تقریب از مقدار مورد انتظار (Expected value) در هر محاسبه بدست می‌آید. بطور مثال فرض کنید برای تقریب مقدار واقعی x از الگوریتمی استفاده می‌کنیم که گام به گام به مقدار واقعی نزدیک می‌شود. یعنی در هر گام یک تقریب پیدا می‌کنیم که تقریب بهتری نسبت به گام قبلی است. الگوریتم در گام (i)ام مقدار x را با $\tilde{x}_i$ و در گام $(i+1)$ام

با $\tilde{x}_{i+1}$ تقریب می‌زند. از آنجاییکه الگوریتم در گام $(i+1)$ام تقریب بهتری نسبت به گام قبل دارد، لذا در این لحظه مقدار مورد انتظار $\tilde{x}_{i+1}$ است و خطای مطلق تقریبی e_a و نسبی تقریبی ϵ_a را با روابط زیر تعریف می‌کنیم.

$$e_a = |\tilde{x}_{i+1} - \tilde{x}_i|, \qquad \epsilon_a = \frac{|\tilde{x}_{i+1} - \tilde{x}_i|}{|\tilde{x}_{i+1}|} \tag{۴.۱}$$

در روش‌هایی که تقریب‌زدن مرحله به مرحله انجام می‌شود و روش بر مبنای تکرار می‌باشد، قطعا الگوریتم نمی‌تواند تا بی‌نهایت ادامه پیدا کند و در نهایت در مرحله‌ای باید متوقف شود. انتخاب این مرحله معمولا از طریق انتخاب کران بالای خطای تقریب مورد نظر انجام می‌گیرد. این خطای از پیش تعیین شده را خطای توقف می‌نامیم و به ترتیب خطای مطلق و نسبی توقف را با e_s و ϵ_s نمایش می‌دهیم. پس در الگوریتم‌های تکراری، مراحل را آنقدر تکرار می‌کنیم تا در مرحله‌ای خطا از خطای توقف کمتر شود. فرض کنید با یک روش عددی تکراری تصمیم داریم عدد $x = \sqrt{2} = 1.414213562...$ را با ۲ رقم اعشار تقریب بزنیم. پس حاصل تقریب باید عددی بزرگتر یا مساوی 1.405 و کوچکتر از 1.415 باشد. هر عددی در این بازه $\tilde{x} \in [1.405, 1.415)$ که به ۲ رقم اعشار گرد شود منجر به عدد $\tilde{x} = 1.41$ خواهد شد. از نگاه دیگر هر تقریبی در این بازه از مقدار واقعی فاصله‌ای کمتر از 0.005 دارد.

$$e_t = |x - \tilde{x}| \leq 0.005 = 0.5 \times 10^{-2}$$

این نتیجه که برای ۲ رقم اعشار بدست آمد را می‌توان به ارقام اعشار بالاتر تعمیم داد و در حالت کلی برای تقریب تا n **رقم اعشار (Decimal Point)** از خطای توقف

$$e_a \leq e_s = 0.5 \times 10^{-n} \tag{۵.۱}$$

استفاده نمود. دقت کنید که برای شرط توقف تکرار مراحل، کوچکتر شدن خطای مطلق تقریبی e_a از خطای توقف e_s بررسی می‌شود که در هر مرحله قابل محاسبه است. بجای شرط روی خطای مطلق، می‌توان شرط توقف را روی خطای نسبی اعمال نمود. در این حالت می‌توان نشان داد که تقریب مورد نظر تا حداقل n **رقم معنی‌دار (Significant Figure)** قابل اطمینان است.

$$\epsilon_a \leq \epsilon_s = 0.5 \times 10^{-n} \tag{۶.۱}$$

مجددا فرض کنید که بخواهیم عدد $x = \sqrt{2} = 1.414213562...$ را تقریب بزنیم و این دفعه مد نظر ما بجای ۲ رقم اعشار، دقت تا ۲ رقم معنی‌دار باشد. برای پیدا کردن ارقام معنی‌دار یک عدد، کافی است آن را در نماد علمی (Scientific Notation) بنویسیم. سپس تعداد ارقام مانتیس، اعم از اعشار و صحیح معرف ارقام معنی‌دار خواهد بود.

چون در این مثال، تقریب با دقت ۲ رقم معنی‌دار می‌خواهیم، کافی است عدد تقریبی در نماد علمی تا ۱ رقم اعشار با عدد واقعی برابر باشد. یعنی برابر $\tilde{x} = 1.4 \times 10^0$ باشد. پس حاصل تقریب باید از 1.35×10^0 بزرگتر یا مساوی و از 1.45×10^0 کوچکتر باشد. هر عدد در این بازه بیشترین خطای نسبی که می‌تواند داشته باشد برابر نصف بازه تقسیم بر کوچکترین عدد ممکن با **همان تعداد ارقام معنی دار** است که با رابطه‌ی زیر داده می‌شود.

$$\epsilon_{max} = \frac{\frac{(1.45-1.35)}{2} \times 10^0}{1.40 \times 10^0} \leq \frac{\frac{(0.1)}{2} \times 10^0}{1.00 \times 10^0} = \frac{0.1}{2 \times 1.00} = \frac{0.05}{1.00} = 0.005 = 0.5 \times 10^{-2}$$

همانطور که دیده می‌شود اگر تقریب ما در بازه‌ای قرار بگیرد که خطای نسبی آن از $\epsilon_s = 0.5 \times 10^{-2}$ کوچکتر باشد حتما در بازه‌ای نیز قرار می‌گیرد که تا ۲ رقم معنی‌دار قابل اطمینان باشد. که این در تایید رابطه‌ی (۶.۱) است. در مثال زیر نحوه‌ی استفاده از خطای توقف را در عمل مشاهده می‌کنیم.

مثال شماره: ۱

تابع نمایی را می‌توان با رابطه زیر تقریب‌زد:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + ... + \frac{x^n}{n!}$$

چه تعداد از جملات این بسط استفاده کنیم تا بتوانیم مقدار $e^{0.5} = 1.648721...$ را با سه رقم **اعشار (معنی‌دار)** تقریب بزنیم؟

پاسخ: ابتدا تابع (approximate) را می‌نویسیم تا مقدار تابع نمایی را به ازای هر تعداد جمله‌ی مد نظر (n) به ازای هر مقدار (x) محاسبه نماییم. برای این منظور از تابع فاکتوریل استفاده می‌کنیم. سپس با زدن حلقه روی تعداد جملات و افزودن تعداد جملات مرحله به مرحله دقت تابع را افزایش می‌دهیم و در هر مرحله با استفاده از خطای توقف چک می‌کنیم که باید ادامه دهیم و یا از حلقه خارج شویم. داخل برنامه متغیر error type مشخص می‌کند که برنامه را برای خطای نسبی و یا مطلق اجرا می‌کنیم که در نتیجه تعداد ارقام معنی‌دار و یا ارقام اعشار را بررسی نمایید. همانطور که در جدول خروجی برای خطای مطلق مشاهده می‌کنیم با استفاده از ۶ جمله مقدار تابع را با عدد $\tilde{x} = 1.6487$ تقریب می‌زنیم و خطای مطلق تقریبی می‌شود: $e_{\tilde{x}} = 0.0003$

برای محاسبه جملات بسط تیلور، ابتدا نیاز به تعریف تابع فاکتوریل داریم.

```
def factorial(n):
    fac_ = 1
    if n >= 1:
        for i in range(1,n+1):
            fac_ = fac_*i
    return fac_
```

حالا می‌توانیم جملات بسط را محاسبه و با هم جمع نماییم. در تابع زیر مقدار بسط در نقطه‌ی x تا درجه‌ی n محاسبه می‌شود.

```
def taylor_exp(x,n):
    sum_ = 0
    for i in range(n+1):
        sum_ = sum_ + (x**i)/factorial(i)
    return sum_
```

با داشتن مقدار بسط در نقطه‌ی x با استفاده از کتابخانه‌ی numpy مقدار تابع نمایی را نیز در نقطه‌ی x محاسبه می‌کنیم. در زیر در ۴ تابع مستقل زیر به ترتیب خطاهای مطلق دقیق، نسبی دقیق، مطلق تقریبی و نسبی تقریبی را محاسبه می‌کنیم.

```
import numpy as np

def error_true(x,n):
    y = np.exp(x)
    yn = taylor_exp(x,n)
    etn_ = abs(y-yn)
    return etn_

def error_relative_true(x,n):
    y = np.exp(x)
    yn = taylor_exp(x,n)
    etn_ = abs(y-yn)/abs(y)
    return etn_

def error_approximate(x,n):
    yn0 = taylor_exp(x,n)
    yn1 = taylor_exp(x,n+1)
    ean_ = abs(yn0-yn1)
    return ean_
```

```
def error_relative_approximate(x,n):
    yn0 = taylor_exp(x,n)
    yn1 = taylor_exp(x,n+1)
    ean_ = abs(yn0-yn1)/abs(yn1)
    return ean_
```

در نهایت نتیجه محاسبات را در جدول زیر خلاصه می‌کنیم. همانطور که دیده می‌شود بازای n=4 به تقریب ۳ رقم معنی‌دار و همچنین ۳ رقم اعشار خواهیم رسید.

$$e^{(0.5)} = 1 + 0.5 + \frac{0.5^2}{2} + \frac{0.5^3}{3!} + \frac{0.5^4}{4!} = 1.64844$$

n	y_app	error_true	error_approx	error_relative_true	error_relative_approx
0.00000	1.00000	0.64872	0.50000	0.39347	0.33333
1.00000	1.50000	0.14872	0.12500	0.09020	0.07692
2.00000	1.62500	0.02372	0.02083	0.01439	0.01266
3.00000	1.64583	0.00289	0.00260	0.00175	0.00158
4.00000	1.64844	0.00028	0.00026	0.00017	0.00016
5.00000	1.64870	0.00002	0.00002	0.00001	0.00001

۲.۱ خطای گرد کردن

خطاها معمولا از یک محدودیت در محاسبات ناشی می‌شوند. و محدودیت در نگهداری اعداد در حافظه‌ی کامپیوتر حین محاسبات منجر به خطای گرد کردن می‌شود. برای آشنا شدن با این نوع خطا ابتدا می‌بایست با نحوه نگهداری و نمایش اعداد در کامپیوتر آشنا شویم. در محاسبات علمی که با اعداد بسیار بزرگ و بسیار کوچک سر و کار داریم، از نمایش **نشانگر شناور (Floating-Point Representation)** که بسیار به نماد علمی شبیه است، استفاده می‌کنیم که در آن یک عدد مخالف صفر بشکل زیر نمایش داده می‌شود.

$$\pm s \times b^e, \qquad 1 \leq s < b \tag{۷.۱}$$

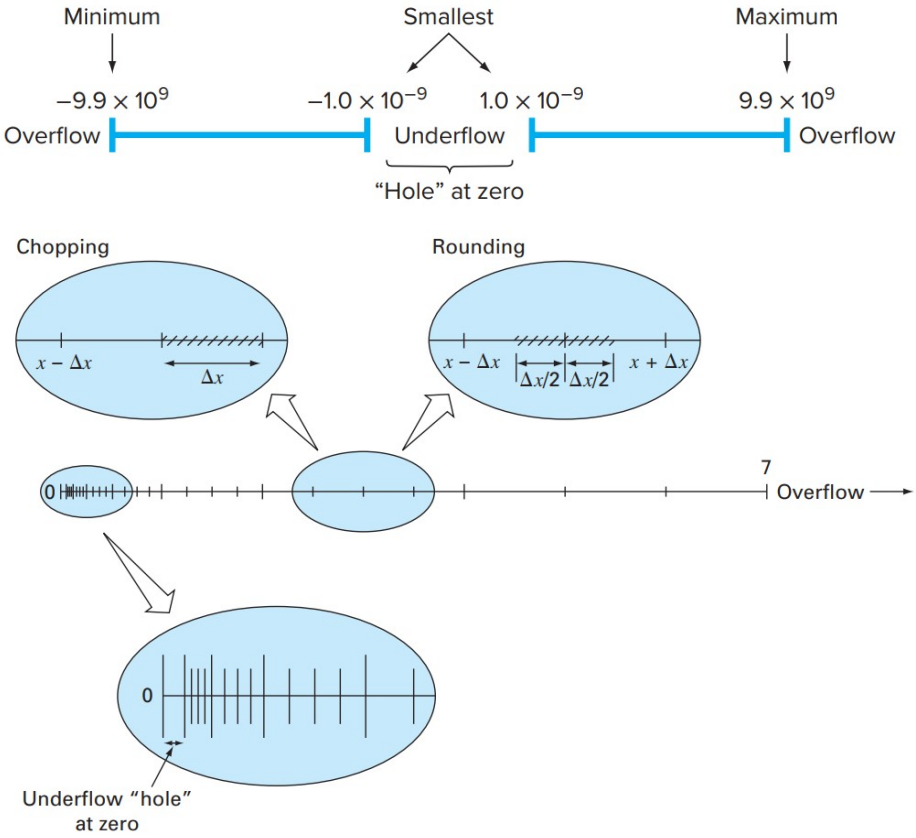
که در آن s رقم بامعنی یا مانتیس (Significand or Mantissa) و b پایه (Base) و e نما (Exponent) می‌باشد. بطور مثال اگر در کامپیوتری هر عدد در پایه ۱۰ در ۵ رقم (Digits) فضا ذخیره شود، آن عدد به شکل زیر نشان داده می‌شود:

$$s_1 d_1 . d_2 \times 10^{s_0 d_0}$$

که در آن s_0 و s_1 معرف علامت‌ها می‌باشند. بزرگترین و کوچکترین عدد مثبتی که این کامپیوتر می‌تواند در حافظه خود نگهداری کند برابر مقادیر زیر است.

$$X_{max} = +9.9 \times 10^{+9}, \qquad X_{min} = +1.0 \times 10^{-9}$$

همین اعداد برای جهت منفی محور با علامت مخالف صادق می‌باشند. لذا در محاسبات با این کامپیوتر، چنانچه با اعدادی خارج از این بازه‌ها مواجه شویم، برنامه پیغام خطای Underflow یا Overflow می‌دهد. در شکل ۲.۱ بطور شماتیک این دو نوع خطا نمایش داده شده است.



شکل ۲.۱ : مفهوم خطاهای Underflow و Overflow و گردکردن. (منبع: کتاب چاپرا)

حال فرض کنید که در این کامپیوتر در حین محاسبات با عدد $x = 2^{-5} = 0.03125$ مواجه شویم که با توجه به ساختار حافظه‌ی این کامپیوتر عدد مذکور بشکل $\tilde{x}_a = +3.1 \times 10^{-2} = 0.031$ دیده می‌شود و کامپیوتر مرتکب خطایی به اندازه‌ی $e_{\tilde{x}} = 0.00025$ می‌شود که اصطلاحاً **خطای گرد کردن (Roundoff Error)** نامیده می‌شود. در شکل ۲.۱ اعدادی که در حافظه‌ی این کامپیوتر می‌توانند دیده شوند توسط خطوط نازک روی محور افقی مشخص شده‌اند. هر عدد بین این اعداد به نزدیک‌ترین خط نازک خود گرد می‌شود و همانطور که گفته شد بخشی از عدد از بین می‌رود که خطای گرد کردن نامیده می‌شود. بیشترین خطایی که در عمل گرد کردن رخ می‌دهد به اندازه‌ی نصف فاصله‌ی دو خط نازک می‌باشد. به وضوح مشخص است که اگر اختلاف بیشتر باشد عدد به

خط نازک دیگر (همسایه) گرد می‌شود. بطور معمول اعداد بین خطوط نازک را که از چشم کامپیوتر پنهان می‌مانند به نقاط قابل رویت کامپیوتر گرد می‌شوند، اما در بعضی موارد بجای گرد کردن اعداد بریده (Chopping) می‌شوند که در این حالت بیشینه‌ی خطای مرتکب شده می‌تواند به اندازه فاصله دو خط نازک از هم باشد. این خطوط نازک که فقط آن‌ها توسط کامپیوتر تشخیص داده می‌شوند، معرف مقدار **تفکیک پذیری (Resolution)** کامپیوتر می‌باشند. بطور مثال در این کامپیوتر مورد مثال ما مقدار تفکیک پذیری اعداد برابر $0.1 \times 10^{s_0 d_0}$ است. و بیشینه‌ی خطای گرد کردن آن نصف این مقدار یعنی $0.05 \times 10^{s_0 d_0}$ می‌شود. بیشینه خطای نسبی گرد کردن **دقت ماشین (Machine Epsilon or Machine Precision)** نامیده می‌شود. پس دقت ماشین برای این کامپیوتر (خطای نسبی بیشینه) برابر $\epsilon_m = 0.05$ می‌شود. در شکل ۲.۱ دقت کنید که فاصله‌ی خطوط روی محور اعداد در نقاط مختلف متفاوت است، پس خطای گرد کردن بسته به مقدار عدد متفاوت است. در اعداد بزرگتر خطای گرد کردن بزرگتر است، اما خطای نسبی (دقت ماشین) آن ثابت است. در مثال زیر با نحوه‌ی محاسبه‌ی دقت ماشین آشنا می‌شوید. در الگوریتم ارایه شده، از این فرض استفاده شده است که اپسیلون ماشین کوچکترین عددی است که توسط ماشین تشخیص داده می‌شود.

مثال شماره: ۲

```
epsilon = 1
while True:
    if epsilon+1 <= 1: break
    epsilon = epsilon/2

epsilon = 2 * epsilon
print(epsilon)
```

```
2.220446049250313e-16
```

خطای گرد کردن هر چند اندک اما در هر عملیات حسابی (Arithmetic Manipulations) بوجود می‌آید. لذا مقدار این خطا در **عملیات‌های حسابی به تعداد زیاد** به خوبی ظاهر می‌شود. در مثال زیر نمونه‌ای از این خطا را مشاهده می‌کنید. به همین دلیل ما سعی می‌کنیم، همواره از الگوریتم‌هایی استفاده کنیم تا تعداد عملیات حسابی را تا حد امکان کاهش دهیم. بطور مثال در محاسبه چند جمله‌ای ها روش لانه‌ی تو در تو یکی از ابزارهایی است که تعداد عملیات حسابی را کم می‌کند.

مثال شماره: ۳

```
s = 0
for i in range(10000):
    s += 0.0001
print(f's={s}')
```

---

s=0.9999999999999062

شکل دوم از خطای گرد کردن وقتی رخ می‌دهد که **یک عدد خیلی بزرگ با یک عدد خیلی کوچک جمع می‌شود**. به مثال زیر توجه کنید.

مثال شماره: ۴

```
x = 0.4*10**(10) + 0.1*10**(-5)
print(f'x={x:.10f}')
```

---

x=4000000000.0000009537

و در نهایت **تفاضل دو عدد که بسیار به هم نزدیک هستند** نیز موجب از دست دادن ارقام معنی دار و خطای گرد کردن می‌شود.

مثال شماره: ۵

مقدار عبارتهای E_1 و E_2 که با هم برابر هستند را برای $x = 1.0$ تا $x = 10^{-8}$ محاسبه می‌کنیم. همانطور که دیده می‌شود در مقادیر بسیار کوچک x مقادیر محاسبه شده برای E_1 و E_2 کاملا متفاوت هستند.

$$E_1 = \frac{1-\cos x}{\sin^2 x}, \qquad E_2 = \frac{1}{1+\cos x}$$

```
import tabulate
from numpy import sin,cos

def E1(x):
    return (1-cos(x))/sin(x)**2

def E2(x):
    return 1/(1+cos(x))

x_,E1_,E2_ = [],[],[]

for i in range(9):
    x = 10**(-i)
    x_.append(x)
    E1_.append(E1(x))
    E2_.append(E2(x))

data = {'x': x_, 'E1': E1_, 'E2' : E2_}
print(tabulate(data, headers="keys", floatfmt="0.8f"))
```

x	E1	E2
1.00000000	0.64922321	0.64922321
0.10000000	0.50125209	0.50125209
0.01000000	0.50001250	0.50001250
0.00100000	0.50000012	0.50000013
0.00010000	0.50000000	0.50000000
0.00001000	0.50000004	0.50000000
0.00000100	0.50004445	0.50000000
0.00000010	0.49960036	0.50000000
0.00000001	0.00000000	0.50000000

۳.۱ خطای برشی

نوع دیگری از خطا که در محاسبات عددی با آن مواجه می‌شویم هنگامی است که از توابع متعالی (Transcendent Functions)، مانند توابع مثلثاتی استفاده می‌کنیم. در محاسبات عددی این توابع را با بسط‌های تیلور آن‌ها تقریب می‌زنیم و این تقریب زمانی دقیق‌تر می‌شود که تعداد بیشتری از جملات بسط استفاده کنیم. در عمل مجبور هستیم در تقریب خود به تعداد محدودی از جملات بسنده کنیم. و این منجر به خطای محاسباتی می‌شود که آن را خطای برشی می‌نامیم. **بسط تیلور درجه‌ی n** برای تابع تک متغیره $f(x)$ حول نقطه‌ی x_0 از رابطه‌ی زیر بدست می‌آید.

$$\begin{aligned} f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2!}f''(x_0)(x-x_0)^2 \\ + ... + \frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n + R_n \end{aligned} \tag{۸.۱}$$

در محاسبات عددی از برش n+1 جمله‌ی اول این بسط که یک چندجمله‌ای درجه‌ی n می‌باشد، برای تقریب تابع $f(x)$ استفاده می‌کنیم. الباقی جملات بسط که اختلاف مقدار واقعی $f(x)$ و تقریب به کار رفته است را **خطای برشی** می‌نامیم. در شکل ۳.۱ تقریب تابع در نقطه‌ی x_{i+1} نشان داده شده است. در این تقریب از مرتبه‌های صفر، یک و دو استفاده شده است. همانطور که در شکل دیده می‌شود با افزایش مرتبه‌ی تقریب، اختلاف مقداری تقریبی از مقدار واقعی تابع در نقطه‌ی x_{i+1} کم می‌شود و به عبارتی با افزایش مرتبه‌ی تقریب خطای برشی کاهش می‌یابد.

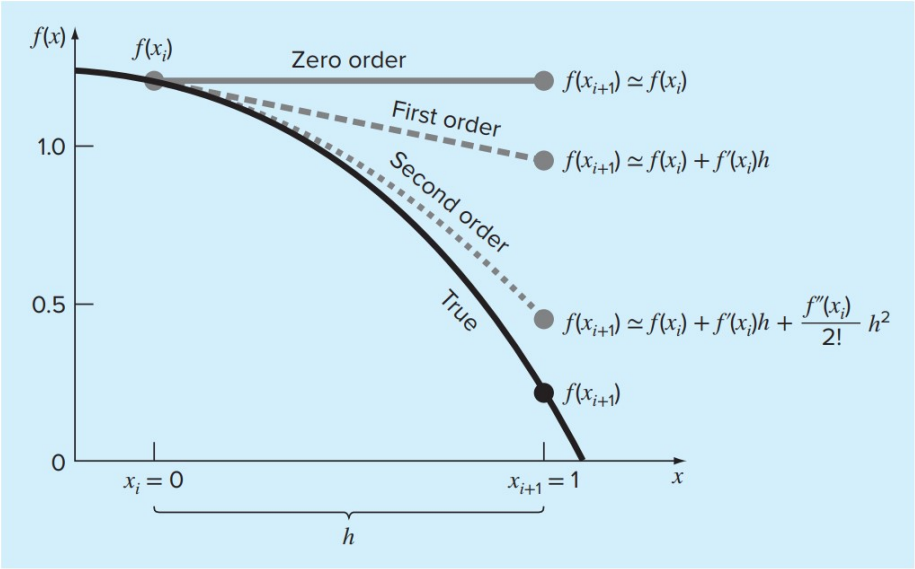
هنگامیکه از مرتبه‌ی تقریب صحبت می‌کنیم معمولا از نماد $\mathcal{O}$ استفاده می‌کنیم. بطور مثال هنگامیکه می‌گوییم تابع $f(x)$ از مرتبه‌ی تابع $g(x)$ است، به شکل $f(x) = \mathcal{O}(g(x))$ نمایش می‌دهیم. و این به معنای آن است که « در نهایت این دو تابع با هم برابر خواهند شد ». منظور از « در نهایت ... » باز می‌گردد به یک پارامتر و یا متغیر مشترکی بین این دو تابع که به یک مقدار مشخصی میل می‌کند. بطور مثال تابع $f(n) = an^2 + bn + c$ برای nهای بزرگ هم مرتبه‌ی تابع $g(n) = n^2$ است و به این شکل $f(n) = \mathcal{O}(n^2)$ نمایش می‌دهیم. و یا تابع $f(n) = an^{-1} + bn^{-2}$ هم مرتبه‌ی $g(n) = n^{-1}$ است و به این شکل $f(n) = \mathcal{O}(n^{-1})$ نشان داده می‌شود. مجدد به شکل ۳.۱ توجه کنید. اولین تقریب تابع $f(x)$ یک چند جمله‌ای از درجه‌ی صفر و یا یک مقدار ثابت است. این تقریب نسبت به پارامتر h که فاصله‌ی نقاط x_{i+1} و x_i است، از مرتبه‌ی صفر است.

$$f(x_{i+1}) \approx f_0(x_{i+1}) = f_0(x_i) = \mathcal{O}(h^0)$$

و تقریب بعدی یک چندجمله‌ای درجه‌ی یک از متغیر h است.

$$f(x_{i+1}) \approx f_1(x_{i+1}) = f(x_i) + f'(x_i)h = \mathcal{O}(h)$$

در عمل **حد بالای خطای برشی** بسط از درجه‌ی n حول نقطه‌ی x_0 را می‌توان از عبارت (۹.۱) بدست آورد. همانطور که در این عبارت دیده می‌شود، خطا از مرتبه‌ی $(n+1)$ام اندازه‌ی گام‌ها



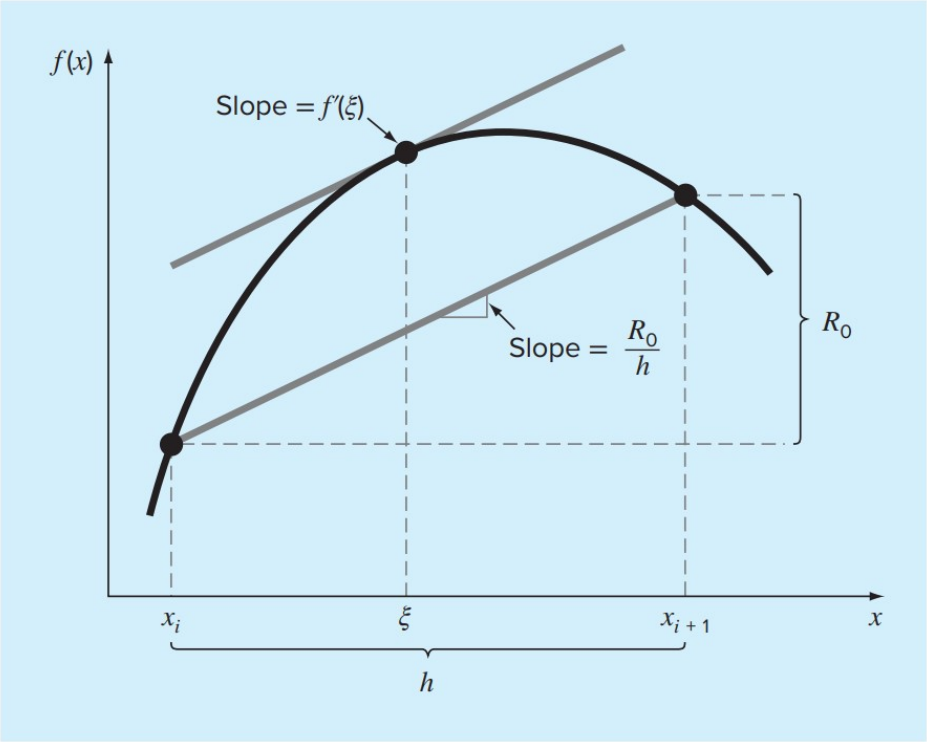
شکل ۱.۳: تقریب تا مرتبه‌ی ۲ تابع $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$ در نقطه‌ی $x = 1$ (منبع: کتاب چاپرا)

است. منظور از یک گام، فاصله‌ی بین نقطه‌ی x_0 که بسط حول آن داده شده است و نقطه‌ی x که تابع را در آن نقطه می‌خواهیم تقریب بزنیم می‌باشد. در رابطه‌ی (۹.۱) حد بالای خطای برشی (R_n) همچنین وابسته به مشتق $(n+1)$ام تابع $f(x)$ در نقطه‌ای درون فاصله‌ی گام است.

$$R_n = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - x_0)^{n+1}, \quad \xi \in [x - x_0]$$

$$R_n = \mathcal{O}(h^{n+1}), \quad h = x - x_0 \tag{۹.۱}$$

برای تصور این رابطه در شکل (۴.۱) خطای برشی مرتبه‌ی صفر نشان داده شده است. همانطور که در شکل مشاهده می‌شود، با استفاده از قضیه مقدار میانگین مشتق، همواره می‌توان نقطه‌ای را در فاصله‌ی گام پیدا نمود که شیب منحنی $f(x)$ در آن نقطه دقیقا برابر شیب مورد نیاز برای یافتن خطای برشی است. در مواردی که تعیین این نقطه برای ما ممکن نباشد، از آنجاییکه ما به دنبال بدست آوردن حد بالای خطای برشی هستیم، از نقطه‌ای استفاده می‌کنیم که بیشترین قدر مطلق مشتق مذکور را داشته باشد.



شکل ۴.۱: قضیه مقدار میانگین مشتق (منبع: کتاب چاپرا)

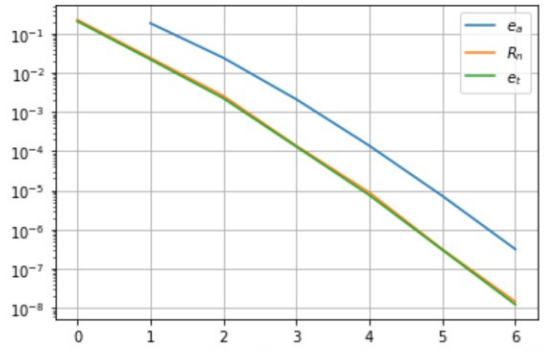
مثال شماره: ۶

با استفاده از بسط سری تیلور حول نقطه‌ی $x_0 = \pi/4$ ، مقدار تقریبی $\cos(\pi/3)$ را تا مرتبه ۶ محاسبه نمایید.

```
from numpy import sin, cos, pi, arange, product
import matplotlib.pyplot as plt

def fun(x,n=0):
    if n%4 == 0: return cos(x)
    if n%4 == 1: return -sin(x)
    if n%4 == 2: return -cos(x)
    if n%4 == 3: return sin(x)

def fac(x):
    if x == 0: return 1
    else:
        return product([x-i for i in range(x)])
```

```
def taylor(x,x0,n):
    h = x-x0
    y = 0
    for i in range(n+1):
        y += fun(x0,i)*(h**i)/fac(i)
    return y

x,x0 = pi/3,pi/4
yt = cos(x)

Rn = []
et = []
ea = []

ya = []
for i in range(7):

    ya_i = taylor(x,x0,i)
    ya.append(ya_i)

    et_i = abs(ya_i-yt)
    et.append(et_i)

    dx = (x-x0)/100
    M = max([abs(fun(xi,i+1)) for xi in arange(x0,x,(x-x0)/100)])
    Rn_i = M*((x-x0)**(i+1))/fac(i+1)
    Rn.append(Rn_i)

    if i != 0:
        ea_i = abs(ya_i-ya_p)
        ea.append(ea_i)
    ya_p = ya_i

plt.plot(list(range(1,7)),ea,label=r'$e_a$')
plt.plot(list(range(7)),Rn,label=r'$R_n$')
plt.plot(list(range(7)),et,label=r'$e_t$')
```

```
plt.yscale('log')
plt.legend()
plt.grid(True)
plt.show()


print(f">> order cos(pi/3) et Rn ea")
for i in range(0,7):
    if i == 0:
        print(f">>\t{i}\t{ya[i]:.10f}\t{et[i]:.10f}\t{Rn[i]:.10f}")
    else:
        print(f">>\t{i}\t{ya[i]:.10f}\t{et[i]:.10f}\t{Rn[i]:.10f}
        \t{ea[i-1]:.10f}")
```




```
>>    order     cos(pi/3)       et              Rn              ea
>>      0       0.7071067812    0.2071067812    0.2263814493
>>      1       0.5219866588    0.0219866588    0.0242321674    0.1851201224
>>      2       0.4977544914    0.0022455086    0.0025859900    0.0242321674
>>      3       0.4998691469    0.0001308531    0.0001384039    0.0021146555
>>      4       0.5000075508    0.0000075508    0.0000088620    0.0001384039
>>      5       0.5000003040    0.0000003040    0.0000003162    0.0000072468
>>      6       0.4999999878    0.0000000122    0.0000000145    0.0000003162
```

از بسط تیلور برای تقریب توابع چند متغیره نیز استفاده می‌شود. همانطور که در رابطه‌ی ۱۰.۱ دیده می‌شود بسط را می‌توان بشکل برداری نوشت و به ابعاد بالاتر با تعداد متغیرهای بیشتر تعمیم داد.

$$
\begin{aligned}
f(x_1, x_2, ..., x_N) &= f(\mathbf{x}) \\
f(\mathbf{x}+\delta\mathbf{x}) &= f(\mathbf{x}) + \sum_{j=1}^{N} \frac{\partial f(\mathbf{x})}{\partial x_j}\delta x_j \\
&\quad + \frac{1}{2!}\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\delta x_i \delta x_j + ...
\end{aligned}
\tag{۱۰.۱}
$$

بطور مثال در رابطه‌ی (۱۱.۱) بسط برای یک تابع دو متغیره ساده شده است و مقدار تقریبی تابع $f(x_1, x_2)$ در نقطه‌ای به فاصله‌ی جزئی $(\delta x_1, \delta x_2)$ تا جملات با مشتقات مرتبه ۲ آورده شده است.

$$
\begin{aligned}
f(x_1+\delta x_1, x_2+\delta x_2) &= f(x_1, x_2) + \frac{\partial f}{\partial x_1}\delta x_1 + \frac{\partial f}{\partial x_2}\delta x_2 \\
&\quad + \frac{1}{2!}\left[\frac{\partial^2 f}{\partial x_1^2}\delta x_1^2 + 2\frac{\partial^2 f}{\partial x_1 \partial x_2}\delta x_1 \delta x_2 + \frac{\partial^2 f}{\partial x_2^2}\delta x_2^2\right] + ...
\end{aligned}
\tag{۱۱.۱}
$$

۴.۱ ترکیب و انتشار خطا

در ترکیب متغیرها همواره دقت می‌کنیم که خطای آن‌ها با هم هم‌افزایی می‌کنند. یعنی بطور مثال خطای تفاضل دو متغیر برابر مجموع خطای هر یک از متغیرها می‌باشد.

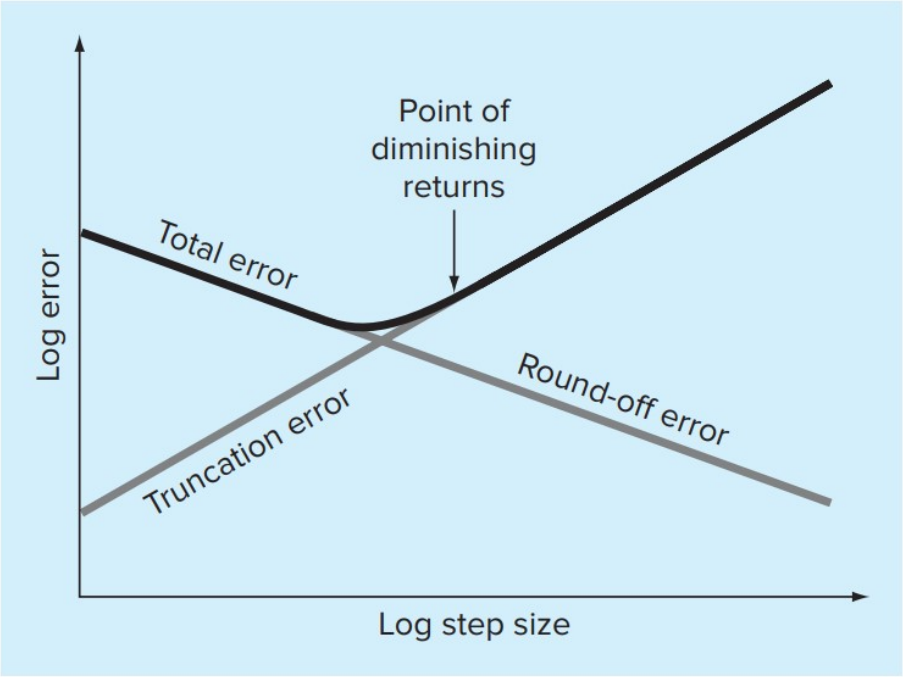
$$
z = x - y, \quad \rightarrow \quad e_z = e_x + e_y \tag{۱۲.۱}
$$

اما خطاها نسبت به پارامترهای مساله می‌توانند رفتار متفاوتی داشته باشند. بطور مثال رابطه‌ی **مشتق تفاضل محدود** مرکزی که در رابطه‌ی (۱۳.۱) آورده شده است را در نظر بگیرید. با اعمال خطای گرد کردن در محاسبات تابع $f(x)$ که با رابطه‌ی (۱۴.۱) داده شده است و دسته‌بندی خطاها، به رابطه‌ی (۱۵.۱) می‌رسیم که در آن همانطور که دیده می‌شود خطاهای گرد کردن و برشی بطور مجزا مشخص می‌باشند. خطای برشی نسبت مستقیم با توان دوم اندازه‌ی گام‌ها در مشتقات تفاضلی (h) تفاضلی دارد. در حالیکه خطای گرد کردن نسبت معکوس با اندازه‌ی گام‌ها دارد. به عبارتی با افزایش گام‌های مشتقات تفاضلی خطای گرد کردن کم می‌شود در حالیکه خطای برشی اضافه می‌شود. در شکل ۵.۱ این رابطه نشان داده شده است. همانطور که در شکل دیده می‌شود، می‌توان طول گام بهینه‌ای پیدا نمود که در آن طول گام خطای کل کمینه باشد.

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(\xi)}{6}h^2 \tag{۱۳.۱}$$

$$f(x_{i\pm1}) = \tilde{f}(x_{i\pm1}) + e_{i\pm1} \tag{۱۴.۱}$$

$$\underset{\text{True Value}}{f'(x_i)} = \underbrace{\frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h}}_{\text{Approximation}} + \underbrace{\frac{e_{i+1} - e_{i-1}}{2h}}_{\text{Roundoff Error}} - \underbrace{\frac{f^{(3)}(\xi)}{6}h^2}_{\text{Truncation Error}} \tag{۱۵.۱}$$



شکل ۵.۱: منحنی خطای گرد کردن، برشی و کل برحسب اندازه‌ی گام‌ها (h) (منبع: کتاب چاپرا)

در توابع خطاهای متغیرهای ورودی (مستقل) در نهایت به متغیرهای خروجی (وابسته) انتقال می‌یابند. انتقال خطا به این نحو را انتشار خطا (Error Propagation) و یا خطای توابع می‌نامیم. تابع تک متغیره در رابطه‌ی (۱۶.۱) را در نظر بگیرید که در آن x متغیر مستقل و y متغیر وابسته می‌باشد. فرض کنید مقدار واقعی x در اختیار نیست و از تقریب آن $\tilde{x}$ استفاده می‌کنیم. همچنین فرض می‌کنیم که جمیع خطاهای مختلف در این تقریب در $\Delta\tilde{x}$ آمده است. حال برای محاسبه‌ی خطا y از بسط تیلور مرتبه اول استفاده می‌کنیم. از آنجاییکه خطاها همواره عددی مثبت می‌باشند،

لذا همانطور که در رابطه‌ی (۱۶.۱) نشان داده است، ضرایب خطاهای متغیر مستقل درون قدر مطلق قرار می‌گیرند تا در نهایت خطای متغیر وابسته مثبت باشد.

$$\begin{aligned} y &= f(x) \\ \Delta\tilde{y} = \Delta f(\tilde{x}) &= |f(x) - f(\tilde{x})| \approx |f'(\tilde{x})|\Delta\tilde{x} \end{aligned} \tag{۱۶.۱}$$

نحوه انتشار خطا در توابع تک متغیره را می‌توان به کمک بسط تیلور توابع چند متغیره که در رابطه‌ی (۱۱.۱) آورده شده بود، به توابع چند متغیره تعمیم داد. که در رابطه‌ی (۱۷.۱) انتشار خطای متغیرهای $\tilde{x_1}$ و $\tilde{x_2}$ به متغیر وابسته‌ی y از طریق تابع دو متغیره $f(x_1, x_2)$ نشان داده شده است. همانطور که دیده می‌شود ضرایب هر یک از خطاهای مستقل درون قدر مطلق آورده شده‌اند.

$$\begin{aligned} y &= f(x_1, x_2) \\ \Delta\tilde{y} &\approx \left|\frac{\partial f}{\partial x_1}\right| \Delta\tilde{x}_1 + \left|\frac{\partial f}{\partial x_2}\right| \Delta\tilde{x}_2 \end{aligned} \tag{۱۷.۱}$$

مثال شماره: ۷

بین متغیرهای x، y و z رابطه‌ی زیر برقرار است.

$$z = \frac{\sin(x)}{xy}$$

(الف) چنانچه از این رابطه استفاده کنیم و متغیر z را بدست آوریم، خطای محاسبه‌ی z را بدست آورید. خطای مطلق متغیرهای x و y را به ترتیب Δx و Δy بگیرید.
(ب) فرض کنید که از رابطه داده شده، می‌خواهید متغیر x را محاسبه نمایید. انتشار خطای متغیرهای y و z را به متغیر x بدست آورید.
(ج) فرض کنید مقادیر x و y را با دقت ۲ رقم اعشار بدست آورده‌ایم:

$$x = 1.57, y = 0.32$$

با رسم منحنی z برحسب x و y خطای متغیر z را مشخص کنید. سپس پاسخ خود را با قسمت (الف) مقایسه نمایید.

پاسخ:

(الف) ابتدا از رابطه برحسب تمام متغیرها دیفرانسیل می‌گیریم. سپس رابطه‌ی حاصل را برحسب دیفرانسیل‌ها مرتب و فاکتورگیری می‌کنیم. ضرایب دیفرانسیل‌ها را درون قدرمطلق قرار داده و در نهایت بجای دیفرانسیل‌ها مقدار خطای هر متغیر را قرار می‌دهیم.

$$\begin{aligned}
\delta z &= \frac{1}{xy}(\cos(x)\delta x) + \frac{\sin(x)}{y}(\frac{-\delta x}{x^2}) + \frac{\sin(x)}{x}(\frac{-\delta y}{y^2}) \\
&= \delta x\left(\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}\right) + \delta y\left(\frac{-\sin(x)}{xy^2}\right) \\
\Delta z &= \Delta x\left|\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}\right| + \Delta y\left|\frac{\sin(x)}{xy^2}\right|
\end{aligned}$$

(ب) در رابطه‌ای که در قسمت (الف) برحسب دیفرانسیل‌ها بدست آوردیم، دیفرانسیل متغیر x را برحسب ۲ متغیر دیگر مرتب می‌کنیم و سپس مانند قسمت (الف) ضرایب را درون قدرمطلق قرار می‌دهیم تا کران بالای خطا را بدست آوریم.

$$\begin{aligned}
\delta x &= \frac{\delta z}{\left(\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}\right)} - \frac{\delta y\left(\frac{-\sin(x)}{xy^2}\right)}{\left(\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}\right)} \\
\Delta x &= \left|\frac{1}{\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}}\right|\Delta z + \left|\frac{\frac{\sin(x)}{xy^2}}{\frac{\cos(x)}{xy} - \frac{\sin(x)}{x^2y}}\right|\Delta y
\end{aligned}$$

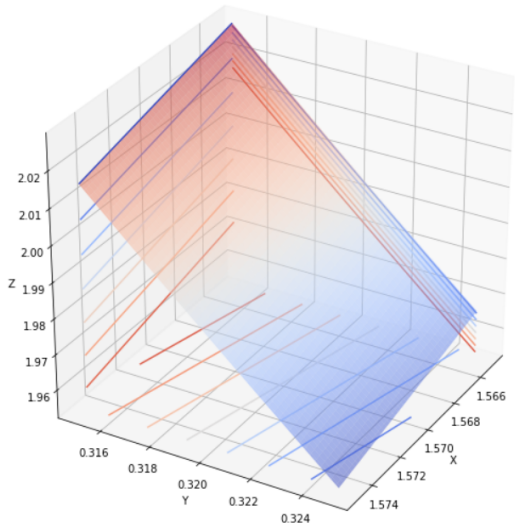
(ج) با توجه به اینکه متغیرهای x و y با دقت ۲ رقم اعشار داده شده‌اند پس برای مقادیر داریم:

$$x = 1.57 \pm 0.005, \quad y = 0.32 \pm 0.005$$

سپس با استفاده از کد زیر سطح z را برحسب x و y رسم می‌کنیم. سایه‌ی سطح z روی صفحات xz و yz نشان می‌دهد که z بطور تقریبی از 1.96 تا 2.02 تغییر می‌کند. پس $z \approx 1.99 \pm 0.03$ با استفاده از رابطه‌ای که در قسمت (الف) برای انتشار خطا به متغیر z نیز مقدار زیر را بدست می‌آوریم که بخوبی نشان می‌دهد مقادیر تقریبی بدست آمده از ترسیم سطح z صحیح می‌باشد.

$$\begin{aligned}
z &= \frac{1}{1.57 * 0.32} = 1.99 \\
\Delta z &= (0.005)\left|0 - \frac{1}{(1.57)^2(0.32)}\right| + (0.005)\left|\frac{1}{(1.57)(0.32)^2}\right| = 0.0374
\end{aligned}$$

```
import matplotlib.pyplot as plt
import numpy as np
from numpy import sin

x = np.linspace(1.57-0.005,1.57+0.005,100)
y = np.linspace(0.32-0.005,0.32+0.005,100)
X,Y = np.meshgrid(x,y)
Z = sin(X)/(X*Y)

x1,y1=1.57-0.005,0.32-0.005
x2,y2=1.57+0.005,0.32+0.005
z1 = sin(x1)/(x1*y1)
z2 = sin(x2)/(x2*y2)

fig = plt.figure(figsize=(10,10))
ax = fig.gca(projection='3d')
ax.plot_surface(X, Y, Z, cmap='coolwarm',alpha=0.5)
cset = ax.contour(X, Y, Z, zdir='z', offset=z2, cmap="coolwarm")
cset = ax.contour(X, Y, Z, zdir='x', offset=1.57-0.005, cmap="coolwarm"
cset = ax.contour(X, Y, Z, zdir='y', offset=0.32-0.005, cmap="coolwarm"

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.view_init(30, 30)
plt.show()
```



در بعضی از توابع انتشار خطای متغیر مستقل به متغیر وابسته بشکلی است که خطا افزایش می‌یابد. این موضوع در روش‌هایی که چند مرحله‌ای می‌باشند بسیار اهمیت می‌یابد. زیرا که یک خطای کم در هر مرحله بشکل فزاینده‌ای انتشار می‌یابد و در نهایت منجر به واگرایی روش و محاسبات می‌شود. لذا برای گریز از جنین مشکلی در ابتدای عملیات، نسبت خطای نسبی متغیر وابسته به متغیر مستقل را محاسبه می‌نماییم. این نسبت را **عدد چگونگی** (Condition Number) می‌نامیم. چنانچه عدد چگونگی از ۱ بیشتر باشد، همانطور که توضیح داده شد، این احتمال وجود دارد که روش واگرا شود. در رابطه‌ی (۱۸.۱) عدد چگونگی برای یک تابع تک متغیره، برحسب تابع و مشتق آن ساده شده است.

$$\mathrm{cn} = \frac{\Delta\tilde{y}/\tilde{y}}{\Delta\tilde{x}/\tilde{x}} = \frac{\tilde{x}f'(\tilde{x})}{f(\tilde{x})} \tag{۱۸.۱}$$

تمرین شماره: ۱

ریشه معادله زیر را با استفاده از فرمول داده شده (روش معمول محاسبه‌ی ریشه معادلات درجه ۲)، بدست آورید. در محاسبه‌ی یکی از ریشه‌ها خطای بسیار زیادی بوجود می‌آید. با راه‌کاری خطای مذکور را رفع نمایید.

$$x^2 + 9^{12}x = 3, \qquad x = \frac{-9^{12} \pm \sqrt{9^{24} + 4(3)}}{2}$$

تمرین شماره: ۲

عبارت‌های داده شده در زیر را در نظر بگیرید.

(۱)
$$y_a(x) = \frac{\tan(x) - x}{x^3}$$

(۲)
$$y_b(x) = \frac{\exp(x) + \cos(x) - \sin(x) - 2}{x^3}$$

(الف) ابتدا به روش تحلیلی حد y_a و y_b را در $x \to 0$ محاسبه نمایید.
(ب) برنامه‌ای بنویسید و در آن $x = 10^{-p}$ را در نظر بگیرید و با افزایش p مقدار x را به صفر نزدیک کنید. متغیرها را در برنامه‌ی خود single تعریف کنید و از $p = 1$ شروع کنید و آنقدر p را افزایش دهید تا مقادیر y همه ارقام معنی‌دار خود را از دست دهند. (در پایتون برای اینکه متغیری single تعریف شود باید از کتابخانه‌ی numpy استفاده کنید و بطور مثال باید بنویسید x = numpy.float32(.....)).
(ج) قسمت (ب) را برای حالتی که متغیرها double تعریف شوند، تکرار نمایید (x = numpy.float64(.....)).
نتایج قسمت‌های (ب) و (ج) را در یک جدول خلاصه نمایید.

تمرین شماره: ۳

در محاسبه‌ی عبارت‌های زیر به ازای $x = 0$ در مرحله‌ای باید دو عدد بسیار نزدیک به هم را از هم کم کنیم و این باعث می‌شود که با خطای محاسباتی گرد کردن مواجه شویم.
(-) بکمک اتحادهای جبری، توابع را به شکل جدیدی تبدیل کنید که مقدار این خطا کاهش یابد.
(-) برنامه‌ای بنویسید و در آن عبارت‌های داده شده و شکل تبدیل یافته‌ی آن‌ها را که خود پیشنهاد داده‌اید، در حد $x \to 0$ محاسبه نمایید. نتایج خود را در یک جدول خلاصه نمایید و نشان دهید که با تبدیل تابع، خطا کاهش پیدا کرده است.

(الف)

$$\frac{1 - \sec x}{\tan^2 x}$$

(ب)

$$\frac{1 - (1 - x)^3}{x}$$

(ج)

$$\frac{1}{1 + x} - \frac{1}{1 - x}$$

تمرین شماره: ۴

(الف) تابع زیر را حول نقطه‌ی $x = \pi/2$ با استفاده از بسط تیلور تقریب می‌زنیم. برای آنکه خطای مطلق تقریب مذکور در بازه‌ی $x \in [0, \pi]$ کمتر از 0.015 باشد، بسط را حداقل تا چه مرتبه‌ای باید استفاده کنیم؟

$$f(x) = x - 1 - \sin x$$

(ب) بیشترین خطا در این بازه در چه نقطه‌ای رخ می‌دهد؟ مقدار خطای مطلق واقعی، تقریبی و برشی را در این نقطه محاسبه نمایید.
(ج) منحنی و تقریب در مرتبه‌ی محاسبه شده و نیز همه‌ی تقریب‌های مرتبه‌ی پایین‌تر را در یک شکل رسم نمایید.

تمرین شماره: ۵

عدد چگونگی را برای هر یک از توابع زیر در نقطه‌ی خواسته شده، محاسبه نمایید و با رسم نمودار تابع مذکور، بر روی عدد محاسبه شده بحث نمایید.

(الف)

$$f(x) = \sqrt{|x-1|} + 1, \quad x = 1.00001$$

(ب)

$$f(x) = e^{-x}, \quad x = 10$$

(ج)

$$\frac{e^{-x} - 1}{x}, \quad x = 0.001$$

تمرین شماره: ۶

سرعت افتادن یک جسم به جرم m از رابطه‌ی زیر بدست می‌آید.

$$v(t) = \frac{mg}{c}(1 - e^{-(c/m)t})$$

که در آن t زمان سقوط، g ثابت گرانش و c ثابت میرایی ناشی از اثرات اصطکاک است. در یک آزمایش که هدف آن اندازه‌گیری این ثابت می‌باشد، جسمی به جرم $m = 5[Kg]$ را از ارتفاعی رها می‌کنیم و بعد از ۲ ثانیه سرعت جسم را $v(2) = 10[m/s]$ اندازه گیری می‌کنیم. در این آزمایش $g = 9.81[m/sec^2]$ در نظر می‌گیریم و همه اندازه‌گیری‌ها را با دقت ۲ رقم معنی‌دار انجام می‌دهیم. مقدار و کران بالای خطای محاسبه c را محاسبه کنید.
راهنمایی: از آنجاییکه معادله‌ی سرعت برحسب ویسکوزیته غیرخطی است و در فصل آینده با حل این معادلات آشنا خواهیم شد، از روش گرافیکی برای حل این معادله استفاده کنید. یعنی خط $v = 10$ را رسم کنید و از روی معادله‌ی داده شده، منحنی v برحسب c از $c = 0.1$ تا $c = 100$ را رسم نمایید. خط و منحنی در یک نقطه همدیگر را قطع می‌کنند. محل تقاطع مقدار c اندازه گیری شده را مشخص می‌کند. بازه‌ی رسم منحنی را بسته تر کرده و منحنی را رسم کنید. این عمل را تکرار کنید تا با دقت مد نظر مقدار c را بدست آورید. برای بدست آوردن خطا نیز کافیست از هر دو طرف معادله‌ی سرعت داده شده نسبت به پارامترهای مساله

دیفرانسیل جزئی گرفته و در نهایت خطای مطلق c را محاسبه نمایید.

فصل ۲

معادلات غیرخطی

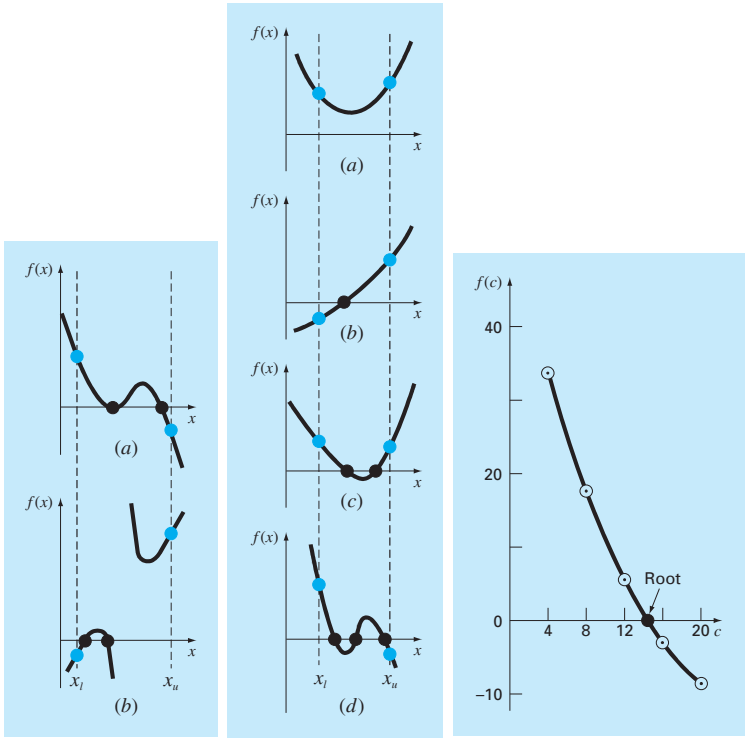
در یک معادله جبری به شکل زیر هدف پیدا کردن نقطه‌ای است که منحنی داده شده با معادله‌ی $f(x)$ محور x را قطع می‌کند. این نقطه را ریشه‌ی معادله می‌نامیم و با x_r نمایش می‌دهیم.

$$f(x_r) = 0$$

همانطور که در شکل (۱.۲) دیده می‌شود، معادلات در بازه‌ی $[x_l, x_u]$ می‌تواند یک یا چند ریشه و یا اصلا ریشه‌ای نداشته باشد. چنانچه علامت مقدار منحنی در دو کران بازه هم علامت باشند، معادله در این بازه یا ریشه‌ای ندارد و یا تعداد زوج ریشه دارد. و همچنین اگر علامت مقدار منحنی در دو کران بازه علامت مخالف داشته باشند، معادله یا یک ریشه و یا تعداد فرد ریشه در این بازه دارد. همانطور که از آخرین منحنی شکل (۱.۲) دیده می‌شود، این قواعد مشروط به پیوسته بودن تابع است و در شمردن تعداد ریشه‌ها در این قاعده، ریشه‌های مضاعف دو برابر شمرده می‌شوند.

۱.۲ روش دوبخشی (Bisection Method)

در این روش ابتدا بازه‌ای که در آن دنبال ریشه می‌گردیم را انتخاب می‌کنیم. یعنی مقدار x_l و x_u به عنوان کران‌های پایین و بالای بازه را مشخص می‌کنیم. چک می‌کنیم که علامت منحنی در این نقاط مخالف باشند و منحنی در این بازه پیوسته باشد، تا مطمئن باشیم که معادله حداقل یک جواب در این بازه دارد. حال بازه را به دو قسمت مساوی تقسیم می‌کنیم. ریشه‌ی واقعی در یکی از این دو قسمت بازه، قسمت پایین یا قسمت بالا قرار دارد. علامت منحنی در وسط بازه (x_m) با علامت منحنی در یکی از دو کران پایین یا بالای بازه مخالف است. ریشه در قسمتی قرار دارد که علامت منحنی در کران آن قسمت مخالف علامت منحنی در وسط بازه است. پس بازه‌ی جدیدی برای جستجوی ریشه انتخاب می‌کنیم که کران‌های آن نقطه‌ی میانی و یکی از دو کران بازه‌ی قدیم (بسته به این که ریشه در قسمت پایین یا بالا قرار دارد کران پایین یا کران بالای بازه‌ی قدیم) می‌باشد.



شکل ۱.۲ : مفهوم ریشه معادله غیر خطی(منبع: کتاب چاپرا)

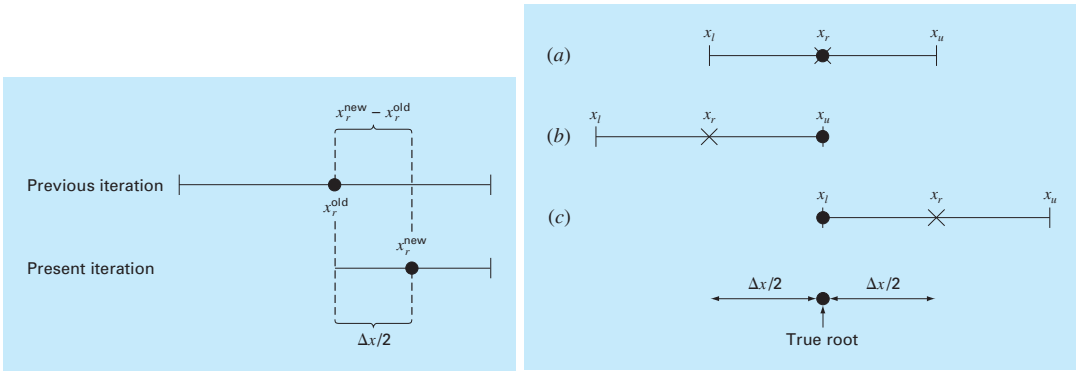
$$x_m = \frac{x_u + x_l}{2}$$

$$\begin{aligned} &\text{if } f(x_m)f(x_u) < 0 : x_l = x_m, \\ &\text{elif } f(x_m)f(x_l) < 0 : x_u = x_m \\ &\text{else : } x_r = x_m \end{aligned} \tag{۱.۲}$$

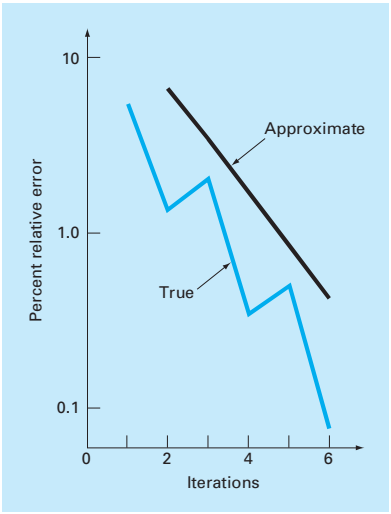
این عمل را به دفعات تکرار می‌کنیم و در هر تکرار طول بازه نصف می‌شود. زمانی این عمل را متوقف می‌کنیم که طول بازه در حد دقت مطلوب کوچک شود. خطای نسبی تقریبی در هر مرحله از رابطه زیر بدست می‌آید. در مرحله‌ای تکرار را متوقف می‌کنیم که خطای نسبی تقریبی از خطای مطلوب توقف (ϵ_s) کوچکتر یا مساوی شود.

$$\epsilon_a = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| = \left| \frac{x_u - x_l}{x_u + x_l} \right| \leq \epsilon_s \tag{۲.۲}$$

همانطور که در شکل ۳.۲ دیده می‌شود همواره و در تمام مراحل، خطای نسبی تقریبی از خطای نسبی واقعی بیشتر است و کران بالای خطای نسبی واقعی محسوب می‌شود. در هر مرحله بازه نصف می‌گردد و بعد از n مرحله تکرار، بازه‌ی جدید و در نتیجه خطا از رابطه زیر محاسبه می‌شود.



شکل ۲.۲: مفهوم روش دو بخشی (منبع: کتاب چاپرا)

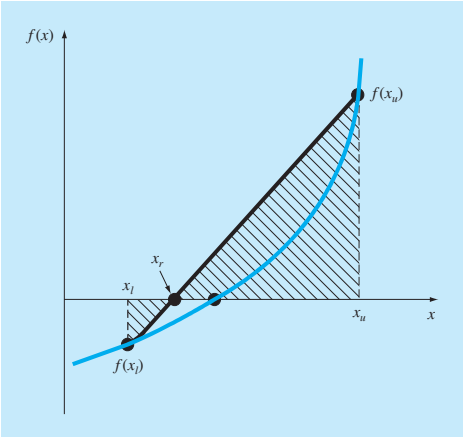


شکل ۳.۲: مقایسه خطای نسبی تقریبی و خطای نسبی واقعی (منبع: کتاب چاپرا)

$$\begin{aligned} \Delta x^0 &= x_u^0 - x_l^0 \\ \Delta x^1 &= \frac{\Delta x^0}{2^1} \\ &\vdots \\ \Delta x^n &= \frac{\Delta x^0}{2^n} \end{aligned} \tag{۳.۲}$$

۲.۲ روش نابجایی (False-Position Method)

در این روش منحنی در بازه‌ی جستجوی ریشه با یک خط تقریب زده می‌شود. و محل تقاطع خط با محور ریشه‌ی تقریبی در این مرحله در نظر گرفته می‌شود. مانند روش دو بخشی، علامت منحنی در این نقطه محاسبه می‌شود. ریشه در سمتی از این نقطه قرار دارد که علامت منحنی در این نقطه مخالف علامت منحنی در کران آن سمت، بالا یا پایین باشد.



شکل ۴.۲: توضیح تصویری از روش نابجایی (منبع: کتاب چاپرا)

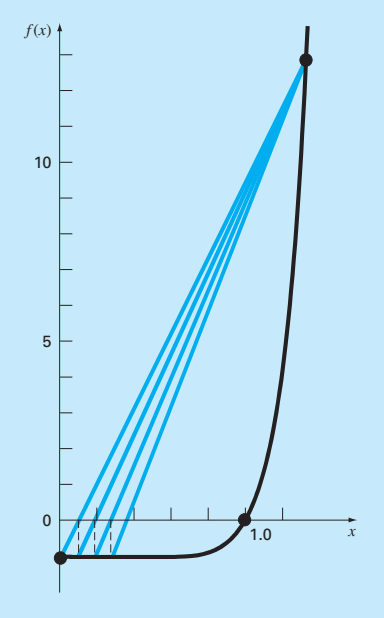
محل برخورد خط تقریب منحنی در بازه‌ی جستجو با استفاده از تشابه مثلث‌ها بدست می‌آید و با ساده کردن روابط به معادله زیر برای محاسبه ریشه‌ی تقریبی در این مرحله می‌رسیم.

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \qquad (۴.۲)$$

با توجه به شکل (۴.۲) ریشه‌ی واقعی باید به کرانی نزدیکتر باشد که مقدار تابع در آن کران به صفر نزدیک‌تر است. اما گاهی اوقات مانند شکل (۵.۲) این اتفاق رخ نمی‌دهد و در چنین مواردی با هر بار تکرار، ریشه‌ی تقریبی x_r، جابجایی کمی خواهد داشت و در نتیجه خطای نسبی تقریبی (ϵ_a) از خطای نسبی واقعی (ϵ_t) کوچکتر خواهد شد.

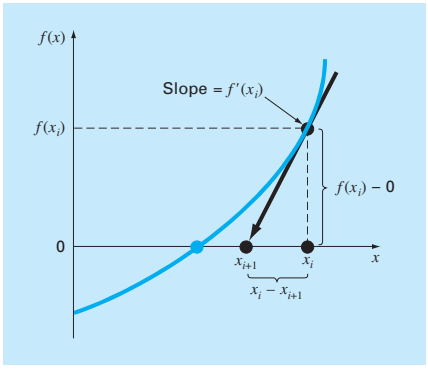
۳.۲ روش نیوتن ـ رافسون (Newton-Raphson Method)

دو روشی که تا اینجا بررسی شد، با انتخاب دو کران یک بازه‌ی جستجو شروع می‌شد. اما روش نیوتن ـ رافسون با انتخاب تنها یک نقطه آغاز می‌شود. منحنی معادله را در اطراف این نقطه‌ی آغازین (x_i) با خط مماس بر منحنی تقریب می‌زنیم. برای پیدا کردن ریشه‌ی معادله، بجای پیدا کردن محل برخورد منحنی با محور x، محل برخورد خط مماس که آن را تقریب منحنی در نظر گرفته‌ایم را، با



شکل ۵.۲: توضیح تصویری از یکی از نقاط ضعف روش نابجایی (منبع: کتاب چاپرا)

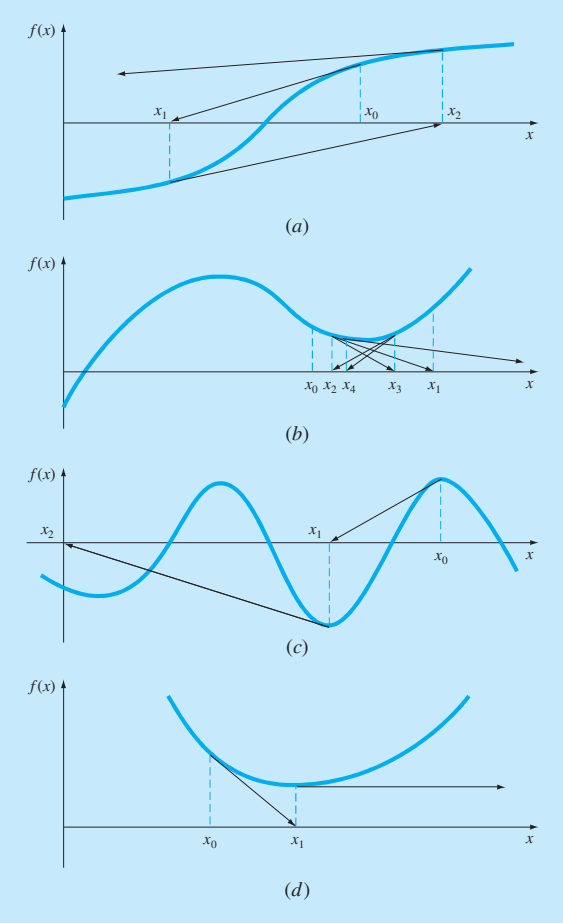
محور x پیدا می‌کنیم. این نقطه‌ی جدید (x_{i+1}) را به عنوان تقریبی از ریشه‌ی معادله در نظر می‌گیریم. در مرحله‌ی بعد منحنی را با خط مماس در این نقطه‌ی جدید تقریب می‌زنیم و محل برخورد خط مماس با محور x را به عنوان تقریبی از ریشه‌ی معادله جستجو می‌کنیم. شکل (۶.۲) بطور نمادین مراحل روش نیوتن–رافسون را نشان می‌دهد. همانطور که در شکل دیده می‌شود، خط مماس، با محور x و خط عمود بر محور x در نقطه‌ی x_i مثلث قائم الزاویه بوجود می‌آورد. با استفاده از این مثلث و تعریف شیب مماس $f'(x_i)$، نقطه‌ی x_{i+1} بدست می‌آید. رابطه‌ی (۵.۲) نقطه‌ی x_{i+1} را بر حسب مقادیر و شیب منحنی در نقطه‌ی x_i بدست می‌آورد.



شکل ۶.۲: توضیح تصویری روش نیوتن–رافسون (منبع: کتاب چاپرا)

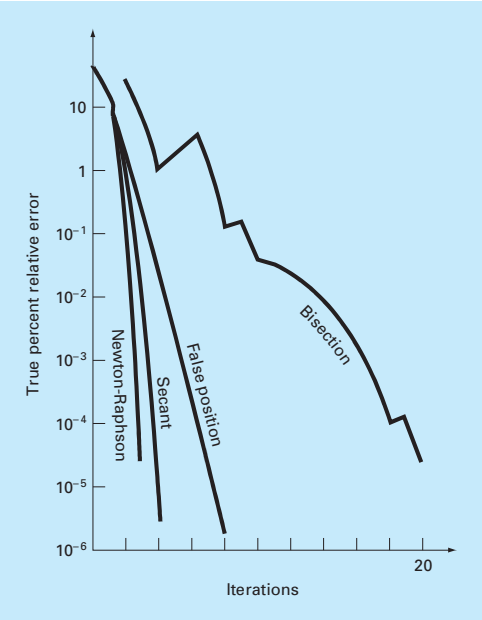
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (۵.۲)$$

روش نیوتن–رافسون از متداول‌ترین روش‌های حل معادلات غیر خطی است که نسبت به روش‌های دیگر سرعت همگرایی نسبتا بالاتری دارد. اما مانند هر روش دیگر، این روش نیز دارای نقاط ضعفی می‌باشد. بطور مثال اگر در مرحله‌ای از حل، شیب منحنی در ریشه‌ی تقریبی بسیار کوچک و صفر باشد، از آنجاییکه شیب در رابطه‌ی (۵.۲) در مخرج کسر قرار دارد، ریشه‌ی تقریبی مرحله‌ی بعد با سرعت زیاد از ریشه‌ی واقعی دور و روش واگرا می‌شود. شکل (۷.۲) چند نمونه از این حالت را نشان می‌دهد.



شکل ۷.۲: رفتار واگرایی روش نیوتن–رافسون در نقاطی که شیب منحنی صفر است (منبع: کتاب چاپرا)

در شکل (۸.۲) روش‌های دوبخشی، نابجایی، وتری و نیوتن–رافسون از نظر سرعت همگرایی با هم مقایسه شده‌اند. همانطور که در شکل دیده می‌شود سرعت همگرایی روش نیوتن–رافسون از ۳ روش دیگر بیشتر می‌باشد.



شکل ۸.۲: مقایسه‌ی سرعت همگرایی در روش‌های دوبخشی، نابجایی، وتری و نیوتن–رافسون (منبع: کتاب چاپرا)

۴.۲ روش نقطه‌ی ثابت (Fixed Point Method)

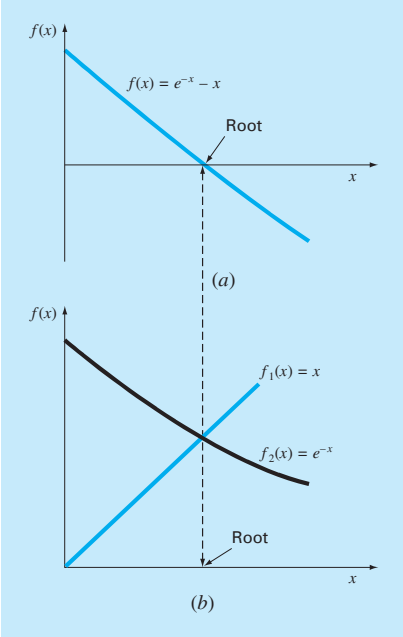
در روش نقطه‌ی ثابت معادله‌ی اصلی مد نظر ریشه یابی را به نحوی تغییر شکل می‌دهیم که منحنی اصلی به تفاضل یک خط (نیمساز ربع اول و سوم) و یک منحنی $g(x)$ تبدیل شود. سپس در هر مرحله همانطور که در روابط (۶.۲) نشان داده شده است، ریشه‌ی تقریبی از روی منحنی جدید بازاء ریشه‌ی مرحله قبل بدست می‌آید.

$$f(x) = 0 \quad \rightarrow \quad f(x) = f_1(x) - f_2(x) = 0$$
$$f_1(x) = x \quad f_2(x) = g(x) \quad \rightarrow \quad x - g(x) = 0 \quad \rightarrow \quad x = g(x)$$

$$x_{i+1} = g(x_i), \qquad \text{if } |g'(x_i)| < 1 \tag{۶.۲}$$

همانطور که در شکل (۹.۲) نشان داده شده است، برای یافتن ریشه‌ی معادله، بجای پیدا کردن محل برخورد منحنی اصل $y = f(x)$ با محور x، هدف به یافتن محل برخورد خط نیمساز $y = x$ و منحنی جدید $y = g(x)$ تبدیل می‌شود. در ابتدا با استفاده از تابع $g(x)$ بازاء یک نقطه‌ی آغازین x_0 نقطه‌ی جدید $x_1 = g(x_0)$ را که محل برخورد منحنی $g(x)$ با نیمساز است، به عنوان اولین ریشه‌ی تقریبی بدست می‌آوریم. سپس با تکرار این مرحله، نقطه به نقطه ریشه‌های تقریبی را پیدا می‌کنیم و گام به گام به ریشه واقعی نزدیک می‌شویم. در شکل (۱۰.۲) چهار حالت مختلف از برخورد منحنی $y = g(x)$ و نیمساز $y = x$ را نشان می‌دهد. در دو قسمت a و b قدرمطلق شیب منحنی

از شیب نیمساز کمتر و در دو قسمت c و d از نیمساز بیشتر است. همانطور که در شکل مشاهده می‌شود در دو قسمت a و b روش همگرا است و در دو قسمت c و d روش واگرا می‌باشد. شرط $|g'(x_i)| < 1$ برای همگرایی در روابط (۶.۲) آورده شده است. در قسمت بعد به اثبات این قضیه و نیز در حالت کلی بررسی شرط همگرایی روش‌های مختلف می‌پردازیم.



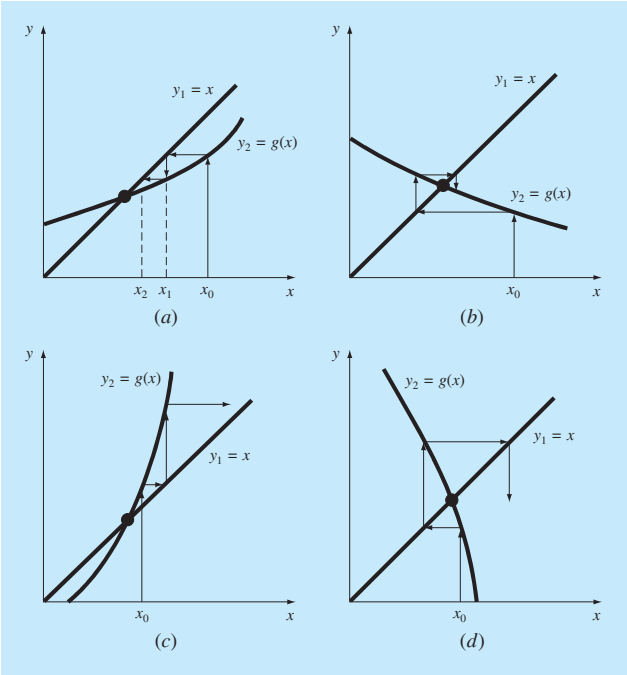
شکل ۹.۲: توضیح تصویری مراحل روش نقطه‌ی ثابت (منبع: کتاب چاپرا)

۵.۲ بررسی هم‌گرایی یک دنباله

دنباله‌ای از اعداد حقیقی x_n را در نظر بگیرید. فرض کنید این دنباله همگرا به a باشد یعنی $\lim_{n\to\infty} x_n = a$ و رابطه‌ی زیر برای مقادیر مثبت c و p برقرار باشد، آنگاه دنباله‌ی $\{x_n\}$ همگرایی مرتبه‌ی p دارد.

$$\lim_{n\to\infty} \left| \frac{x_{n+1} - a}{(x_n - a)^p} \right| = \lim_{n\to\infty} \left| \frac{e_{n+1}}{(e_n)^p} \right| < c \qquad (۷.۲)$$

پس برای پیدا کردن مرتبه همگرایی، همواره می‌بایست خطای گام $(i+1)$ ام را برحسب تابعی از خطای گام (i) ام بدست آورد و به این منظور از بسط تیلور استفاده می‌کنیم. در مثال زیر همگرایی روش نقطه‌ی ثابت بررسی می‌شود.



شکل ۲.۱۰: رفتار واگرایی روش نقطه‌ی ثابت در هنگامی که قدر مطلق شیب منحنی $g(x)$ در محل ریشه‌ی واقعی بزرگتر از ۱ باشد.(منبع: کتاب چاپرا)

مثال شماره: ۱

همگرایی روش نقطه‌ی ثابت: همانطور که در متن گفته شد باید خطای مرحله‌ی $n+1$ ام را بر حسب مرحله‌ی n ام محاسبه نماییم. برای این منظور از دو طرف رابطه‌ی تکرار نقطه‌ی ثابت، ریشه واقعی را کم می‌کنیم و بسط مرتبه اول تابع $g(x)$ را حول ریشه واقعی بدست می‌آوریم. دقت می‌کنیم که خود ریشه واقعی در رابطه‌ی تکرار نقطه ثابت صدق می‌کند.

$$\begin{aligned} x_{n+1} - x_r &= g(x_n) - x_r \\ &= (g(x_r) + g'(x_r)(x_n - x_r)) - x_r \\ &= x_r + g'(x_r)(x_n - x_r) - x_r \\ &= g'(x_r)(x_n - x_r) \\ e_{n+1} &= g'(x_r)e_n \quad \rightarrow \quad \frac{e_{n+1}}{e_n} = g'(x_r) \end{aligned}$$

برای شرط همگرایی، انتظار داریم قدرمطلق خطا گام به گام کمتر شود. یعنی در هر مرحله قدر مطلق خطا از گام قبلی کمتر باشد. پس اولا روش نقطه‌ی ثابت تنها به شرطی همگرا است که $|g'(x_r)| < 1$ و ثانیا در این حالت مرتبه همگرایی $p = 1$ است.

مثال شماره: ۲

همگرایی روش نیوتن ـ رافسون: مجددا برای برای بررسی همگرایی خطای مرحله‌ی $n+1$ ام را محاسبه می‌کنیم.

$$\begin{aligned} e_{n+1} &= x_{n+1} - x_r \\ &= x_n - \frac{f(x_n)}{f'(x_n)} - x_r \\ &= x_n - \frac{f(x_r) + f'(x_r)(x_n - x_r) + \frac{1}{2}f''(x_r)(x_n - x_r)^2}{f'(x_n)} - x_r \end{aligned}$$

و از آنجاییکه ریشه واقعی در معادله صدق می‌کند یعنی $f(x_r) = 0$ و نیز شیب منحنی $f(x)$ در ریشه و نقطه‌ی x_n به شرط آنکه به حد کافی به ریشه نزدیک شده باشیم، بسیار به هم نزدیک هستند یعنی $f'(x_r) \approx f'(x_n)$ رابطه‌ی خطاها به شکل زیر ساده می‌شود:

$$\begin{aligned} e_{n+1} &\approx x_n - (x_n - x_r) - \frac{f''(x_r)(x_n - x_r)^2}{2f'(x_n)} - x_r \\ e_{n+1} &\approx -\frac{f''(x_r)}{2f'(x_n)}e_n^2 \quad \rightarrow \quad \frac{e_{n+1}}{e_n^2} \approx \left| -\frac{f''(x_r)}{2f'(x_n)} \right| \end{aligned}$$

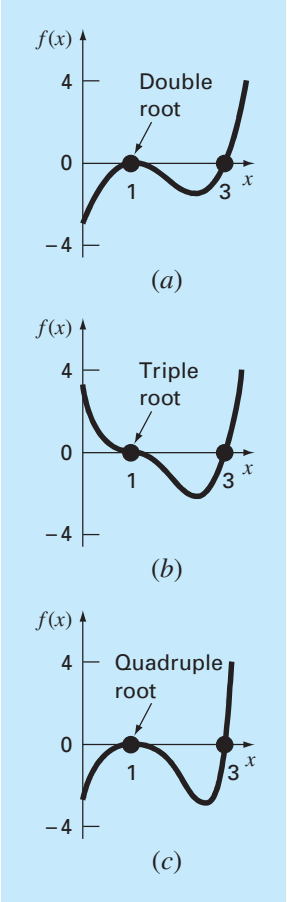
پس همگرایی روش نیوتن از مرتبه ۲ است.

۶.۲ ریشه‌های چندگانه

معادلاتی را در نظر بگیرید که ریشه چندگانه دارند. بطور مثال شکل (۱۱.۲) منحنی سه نمونه از این نوع معادلات را نشان می‌دهد. همانطور که از شکل مشخص است، در محل ریشه‌های چندگانه‌ی این معادلات علاوه بر خود تابع، شیب تابع نیز صفر می‌شود و منحنی بر محور x مماس می‌شود. این خاصیت باعث می‌شود در روش‌هایی مانند نیوتن ـ رافسون که مشتق تابع در مخرج کسر قرار می‌گیرد باعث واگرایی سیستم شود و نیز سرعت همگرایی روش به شدت کاهش یابد. در این نوع مسائل اثبات می‌شود که همواره تابع $f(x)$ زودتر از $f'(x)$ صفر می‌شود، لذا در هر مرحله چک می‌کنیم اگر مقدار تابع $f(x)$ صفر شد، در برنامه دیگر مشتق تابع محاسبه نشود. برای رفع مشکل سرعت

همگرایی نیز می توان روش نیوتن–رافسون را برای پیدا کردن ریشه‌ی m گانه یک معادله به روش زیر بهبود داد.

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)} \qquad (۸.۲)$$



شکل ۱۱.۲: توضیح تصویری از ریشه‌های چندگانه‌ی یک معادله.(منبع: کتاب چاپرا)

۷.۲ دستگاه معادلات غیرخطی

در این قسمت قصد داریم روش‌هایی را بررسی کنیم که چند معادله جبری مورد نظر را توام حل نمایند و نقاطی را پیدا کنند که هم‌زمان در این معادلات صدق نمایند. ما دستگاه معادلاتی را بررسی می‌کنیم که به تعداد معادلاتی که دارند، دارای متغیر مجهول هستند. بطور مثال دو معادله‌ی زیر را در نظر بگیرید.

$$\begin{aligned} u(x,y) &= x^2+y^2-25=0 \\ v(x,y) &= y^2-x-5=0 \end{aligned} \tag{۹.۲}$$

برای ترسیم این دو منحنی باید y را برحسب x بدست آوریم. با توجه به توان y برای هر معادله دو منحنی برای بالا و پایین محور x حاصل می‌شود.

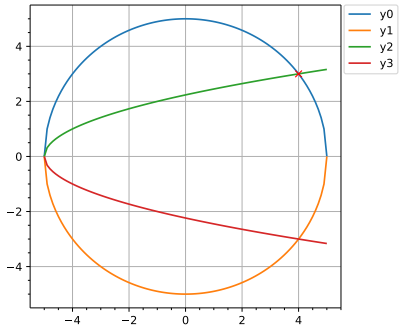
$$\begin{aligned} y &= \pm\sqrt{25-x^2} \\ y &= \pm\sqrt{5+x} \end{aligned} \tag{۱۰.۲}$$

در کد زیر این منحنی‌ها را رسم می‌کنیم. همانطور که دیده می‌شود این منحنی‌ها در سه نقطه‌ی $\{(0,0),(4,3),(4,-3)\}$ همدیگر را قطع می‌کنند. به عنوان صورت مساله هدف پیدا کردن محل برخورد منحنی‌ها در ناحیه اول است، یعنی $(x=4,y=3)$ که در هر دو معادله‌ی دستگاه (۹.۲) صدق می‌کند. این نقطه در شکل با علامت X قرمز رنگ مشخص شده است. در این فصل با دو روش نقطه‌ی ثابت و نیوتن‌ـ رافسون (ژاکوبی) حل می‌کنیم.

مثال شماره: ۳

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-5,5,100)
y0 = np.sqrt(25-x**2)
y1 = -np.sqrt(25-x**2)
y2 = np.sqrt(5+x)
y3 = -np.sqrt(5+x)

fig, ax = plt.subplots(1,1,figsize=(5,5))

ax.plot(x,y0,label='y0')
ax.plot(x,y1,label='y1')
ax.plot(x,y2,label='y2')
ax.plot(x,y3,label='y3')
ax.plot(4,3,'x r')

ax.legend(bbox_to_anchor=(1.01,1), borderaxespad=0,loc='upper left')
```

```
ax.minorticks_on()
ax.grid()

plt.show()
```



۱.۷.۲ روش نقطه‌ی ثابت

مانند حل معادله تک متغیره، معادله باید به نحوی تغییر شکل پیدا کند که در هر معادله یک مجهول برحسب تابعی از الباقی مجهولات نوشته شود. در روابط (۱۱.۲) این تبدیل برای یک دستگاه دو معادله، دو مجهول را نشان می‌دهد.

$$\begin{cases} u(x,y)=0 \\ v(x,y)=0 \end{cases} \Rightarrow \begin{cases} x_{i+1}=\phi(x_i,y_i) \\ y_{i+1}=\psi(x_i,y_i) \end{cases} \qquad (۱۱.۲)$$

روش با یک دسته مقادیر آغازین (x_0,y_0) شروع می‌شود. سپس با استفاده از روابط (۱۱.۲) نقطه‌ی جدید (x_1,y_1) و با تکرار، نقاط بعدی از یک دنباله که انتظار داریم به ریشه‌ی دستگاه منتهی شود، را بدست می‌آوریم. به عنوان شرط هم‌گرایی این دنباله به ریشه‌ی واقعی، رابطه‌ی زیر در بازه‌ی مورد بررسی و نزدیک به ریشه‌ی واقعی باید برقرار باشد.

$$\begin{cases} \left|\frac{\partial\phi}{\partial x}\right|+\left|\frac{\partial\phi}{\partial y}\right|<1 \\ \\ \left|\frac{\partial\psi}{\partial x}\right|+\left|\frac{\partial\psi}{\partial y}\right|<1 \end{cases} \qquad (۱۲.۲)$$

مثال شماره: ۴

دستگاه معادلات غیرخطی (۹.۲) را با روش نقطه ثابت حل نمایید.
ابتدا معادلات را به شکل (۱۱.۲) به نحوی تغییر می‌دهیم که در شرایط (۱۲.۲) صدق نماید.

$$\begin{cases} u(x,y) = x^2 + y^2 - 25 = 0 \\ v(x,y) = y^2 - x - 5 = 0 \end{cases} \Rightarrow \begin{cases} x_{i+1} = \phi(x_i, y_i) = \sqrt{25 - y_i^2} \\ y_{i+1} = \psi(x_i, y_i) = \sqrt{5 + x_i} \end{cases}$$

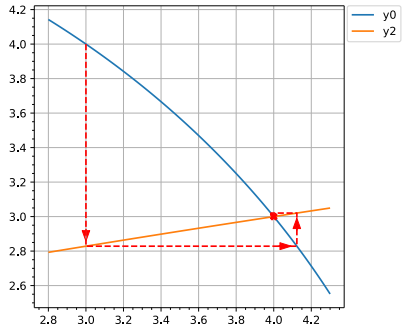
$$\begin{cases} \partial_x \phi(x_i, y_i) = 0 \quad \partial_y \phi(x_i, y_i) = \left| \frac{y_i}{\sqrt{25 - y_i^2}} \right| \\ \partial_x \psi(x_i, y_i) = \frac{1}{2\sqrt{|5+x|}} \quad \partial_y \psi(x_i, y_i) = 0 \end{cases}$$

با توجه به شکل رسم شده، نقطه‌ی برخورد در ناحیه اول ($x \geq 0$) و نیز پایین‌تر از خط ($y = 3.5$) قرار دارد. پس شرط (۱۲.۲) برقرار می‌باشد. حال با استفاده از کد زیر ریشه را پیدا می‌کنیم. همانطور که در کد دیده می‌شود از نقطه‌ی $(x_0, y_0) = (3.00, 4.00)$ آغاز می‌کنیم و آنقدر مراحل را تکرار می‌کنیم که خطای مطلق تقریبی در هر دوی x و y کمتر از 0.005 باشد تا مطمئن شویم فاصله‌ی ریشه‌های تقریبی از محل واقعی برخورد کمتر از دو رقم اعشار می‌باشد. در منحنی رسم شده، مشاهده می‌کنیم که در هر مرحله نقطه‌ی بدست آمده از یک منحنی به منحنی بعدی انتقال می‌یابد. مسیر حرکت نقاط با فلش مشخص شده است.

```
import matplotlib.pyplot as plt
import numpy as np

def phi(x,y):
    return np.sqrt(25-y**2)

def psi(x,y):
    return np.sqrt(x+5)

def makeArrow(ax,x0,y0,x1,y1):
    delta = 0.01
    dx = delta*(x1-x0)/np.sqrt((x1-x0)**2+(y1-y0)**2)
    dy = delta*(y1-y0)/np.sqrt((x1-x0)**2+(y1-y0)**2)

    ax.arrow(x1-10*dx,y1-10*dy,dx,dy,head_width=0.04,
        head_length=0.06,color='red')

x0,y0 = 3,4
```

```
roots = np.array([[x0,y0]])
while True:
    x = phi(x0,y0)
    y = psi(x0,y0)
    roots = np.vstack((roots,np.array([x,y])))
    if abs(x-x0) < 0.005 and abs(y-y0) < 0.005:
        break
    x0,y0=x,y

for i,(xi,yi) in enumerate(roots):
    print(f"x{i}: {xi:.2f}\ty{i}: {yi:.2f}")

x = np.linspace(2.8,4.3,100)
y0 = np.sqrt(25-x**2)
y1 = -np.sqrt(25-x**2)
y2 = np.sqrt(5+x)
y3 = -np.sqrt(5+x)

fig, ax = plt.subplots(1,1,figsize=(5,5))

ax.plot(x,y0,label='y0')
ax.plot(x,y2,label='y2')
ax.plot(4,3,'Xr')
ax.plot(roots[:,0],roots[:,1],'--r')
makeArrow(ax,roots[0,0],roots[0,1],roots[1,0],roots[1,1])
makeArrow(ax,roots[1,0],roots[1,1],roots[2,0],roots[2,1])
makeArrow(ax,roots[2,0],roots[2,1],roots[3,0],roots[3,1])

ax.legend(bbox_to_anchor=(1.01,1), borderaxespad=0,loc='upper left')
ax.minorticks_on()
ax.grid()

plt.show()
```

```
x0: 3.00        y0: 4.00
x1: 3.00        y1: 2.83
x2: 4.12        y2: 2.83
x3: 4.12        y3: 3.02
x4: 3.98        y4: 3.02
x5: 3.98        y5: 3.00
x6: 4.00        y6: 3.00
x7: 4.00        y7: 3.00
```



۲.۷.۲ روش نیوتن–رافسون یا ژاکوبی

به عنوان روش دوم برای حل دستگاه معادلات غیرخطی روش نیوتن–رافسون یا ژاکوبی را بررسی می‌کنیم. مانند معادله تک متغیره، مساله را با یک ریشه تقریبی آغازین شروع می‌کنیم و در هر مرحله با ریشه‌ی مرحله قبل، ریشه‌ی تقریبی جدید را پیدا می‌کنیم. اختلاف ریشه‌ی قدیم با جدید از حاصلضرب تابع معادله در معکوس مشتق آن بدست می‌آید. با این تفاوت که در اینجا مساله چند متغیره و چند معادله هم‌زمان وجود دارند لذا با تعاریف زیر مساله شکل برداری پیدا می‌کند و فرم برداری روش نیوتن–رافسون را بکار می‌بریم. در حالت برداری مشتق توابع معادل ماتریس ژاکوبی خواهد شد و معکوس مشتق تابع معادل معکوس این ماتریس خواهد بود.

$$\begin{cases} u(x,y)=0 \\ v(x,y)=0 \end{cases} \Rightarrow \begin{bmatrix} \vec{X}=\begin{bmatrix} x \\ y \end{bmatrix} \\ \\ \vec{F}=\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} \end{bmatrix} \tag{۱۳.۲}$$

$$\vec{X}_{i+1}=\vec{X}_i-J_i^{-1}.\vec{F}_i \Rightarrow \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix}=\begin{bmatrix} x_i \\ y_i \end{bmatrix}-J_i^{-1}.\begin{bmatrix} u(x_i,y_i) \\ v(x_i,y_i) \end{bmatrix}$$

$$J=\begin{bmatrix} \partial_x u & \partial_y u \\ \partial_x v & \partial_y v \end{bmatrix} \tag{۱۴.۲}$$

مثال شماره: ۵

دستگاه معادلات غیرخطی (۹.۲) را با روش نیوتن–رافسون حل نمایید.

با استفاده از کد زیر ریشه را پیدا می‌کنیم. همانطور که در کد دیده می‌شود مجدد از نقطه‌ی $(x_0, y_0) = (3.00, 4.00)$ آغاز می‌کنیم و آنقدر مراحل را تکرار می‌کنیم که خطای مطلق تقریبی در هر دوی x و y کمتر از 0.005 باشد تا مطمئن شویم فاصله‌ی ریشه‌های تقریبی از محل واقعی برخورد کمتر از دو رقم اعشار می‌باشد. همانطور که در نتیجه کد دیده می‌شود، روش نیوتن رافسون از سرعت همگرایی بیشتری برخوردار است و در ۳ گام به قدر کفایت (۲ رقم اعشار) به ریشه واقعی نزدیک می‌شویم.

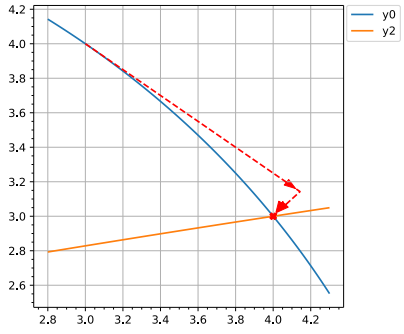
$$\begin{cases} u(x,y) = x^2 + y^2 - 25 = 0 \\ v(x,y) = y^2 - x - 5 = 0 \end{cases} \Rightarrow J = \begin{bmatrix} 2x & 2y \\ -1 & 2y \end{bmatrix}$$

$$\Rightarrow J^{-1} = \begin{bmatrix} \frac{2y}{4xy+2y} & -\frac{2y}{4xy+2y} \\ \frac{1}{4xy+2y} & \frac{2x}{4xy+2y} \end{bmatrix}$$

```
import matplotlib.pyplot as plt
import numpy as np

def Jinv(x,y):
    ji = [[(2*y)/(2*y + 4*x*y), -(2*y)/(2*y + 4*x*y)],
    [1/(2*y + 4*x*y), (2*x)/(2*y + 4*x*y)]]
    return np.array(ji)

def F(x,y):
    return np.array([x**2+y**2-25,y**2-x-5]).reshape((-1,1))

x0,y0 = 3,4
roots = np.array([[x0,y0]])
while True:
    X0 = np.array([x0,y0]).reshape((-1,1))
    X = X0 - np.dot(Jinv(x0,y0),F(x0,y0))
    x,y = X[0,0],X[1,0]
    roots = np.vstack((roots,np.array([x,y])))
    if abs(x-x0) < 0.005 and abs(y-y0) < 0.005:
        break
```

```
    x0,y0=x,y

for i,(xi,yi) in enumerate(roots):
    print(f"x{i}: {xi:.2f}\ty{i}: {yi:.2f}")

x = np.linspace(2.8,4.3,100)
y0 = np.sqrt(25-x**2)
y1 = -np.sqrt(25-x**2)
y2 = np.sqrt(5+x)
y3 = -np.sqrt(5+x)

fig, ax = plt.subplots(1,1,figsize=(5,5))

ax.plot(x,y0,label='y0')
ax.plot(x,y2,label='y2')
ax.plot(4,3,'Xr')
ax.plot(roots[:,0],roots[:,1],'--r')
makeArrow(ax,roots[0,0],roots[0,1],roots[1,0],roots[1,1])
makeArrow(ax,roots[1,0],roots[1,1],roots[2,0],roots[2,1])
makeArrow(ax,roots[2,0],roots[2,1],roots[3,0],roots[3,1])

ax.legend(bbox_to_anchor=(1.01,1), borderaxespad=0,loc='upper left')
ax.minorticks_on()
ax.grid()

plt.show()
```

```
x0: 3.00        y0: 4.00
x1: 4.14        y1: 3.14
x2: 4.00        y2: 3.00
x3: 4.00        y3: 3.00
```



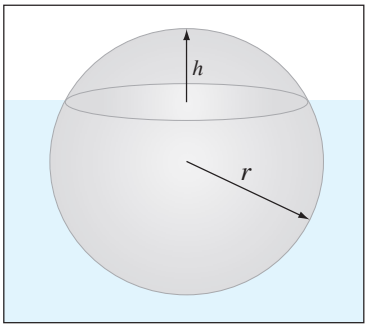
تمرین شماره: ۱

طبق اصل ارشمیدس ، نیروی شناوری برابر است با وزن سیال جابجا شده توسط قسمت غوطه‌ور شده یک جسم. برای کره ای که در شکل زیر به تصویر کشیده شده است، ارتفاع h قسمتی از کره که بالای سطح آب می‌ماند را از دو روش دوبخشی و نابجایی و نیز با استفاده از مقادیر زیر با دقت ۲ رقم اعشار محاسبه کنید.

$$r = 1\ [m]$$
$$\rho_s = 200\ [Kg/m^3]$$
$$\rho_w = 1000\ [Kg/m^3]$$

که ρ_s چگالی کره و ρ_w چگالی آب است. توجه داشته باشید که برای کره، حجم قسمت بیرون از آب را با فرمول زیر می‌توان محاسبه کرد.

$$V = \frac{\pi h^2}{3}(3r - h)$$

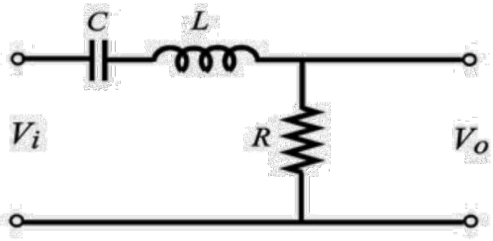


تمرین شماره: ۲

شکل زیر یک مدار فیلتر برای عبور سیگنال‌هایی با محدوده‌ی فرکانس خاصی را نشان می‌دهد. در این فیلتر نسبت ولتاژ خروجی به ورودی از رابطه‌ی

$$RV = \left|\frac{V_o}{V_i}\right| = \frac{\omega RC}{\sqrt{(1-\omega^2 LC)^2 + (\omega RC)^2}}$$

بدست می‌آید. که در آن ω فرکانس ولتاژ ورودی است. اگر $R = 1000\Omega$ ، $L = 11mH$ و $C = 8\mu F$ باشد. محدوده‌ی فرکانسی را به نحوی محاسبه نمایید که $RV \geq 0.87$ باشد. مساله را با دو روش دوبخشی و نابجایی با دقت ۲ رقم معنی‌دار حل نمایید. (راهنمایی: از رابطه و مقادیر داده شده یک نامعادله ایجاد کنید و با حل این نامعادله برحسب ω بازه‌ی فرکانس‌های مدنظر را پیدا کنید.)



تمرین شماره: ۳

الف) جذر عدد ۹ را به روش تکرار نقطه‌ی ثابت و نیوتن_رافسون، با ارائه‌ی یک تابع تکرار مناسب، با دقت ۲ رقم اعشار محاسبه نمایید. (راهنمایی: برای محاسبه‌ی جزر ۹ باید معادله $x^2 - 9 = 0$ را حل نمایید.)
ب) از آنجاییکه جذر عدد ۹ را دقیقا می‌دانیم، در هر مرحله می‌توانیم خطای واقعی را محاسبه کنیم. با الهام از شکل (۸.۲) ، با استفاده از رسم منحنی خطا نسبت به تعداد تکرار مراحل، سرعت همگرایی هر دو روشی که استفاده کرده‌اید را مقایسه نمایید. (برای رسم منحنی از پایتون استفاده کنید.)

تمرین شماره: ۴

معادله $P(x) = \tan(\pi x) - \sqrt{a}\sec(\pi x) + b = 0$ را در نظر بگیرید.
الف) مقدار a و b را چنان پیدا کنید که معادله ریشه‌ی مضاعف $x = 0.25$ داشته باشد.
ب) ریشه مذکور را با روشی بدست آورید که مرتبه همگرایی حداقل از مرتبه ۲ داشته باشد. از نقطه‌ی $x_0 = -0.25$ شروع کنید و آنقدر مراحل را تکرار کنید که با حداقل ۳ رقم معنی‌دار به ریشه‌ی واقعی رسیده باشید.

تمرین شماره: ۵

دستگاه معادلات غیرخطی زیر را با روش تکرار نیوتن و نقطه‌ی ثابت با حدس اولیه $x_0 = y_0 = 1.2$ را حل نمایید. و پاسخ را با دقت ۳ رقم معنی‌دار بدست آورید .

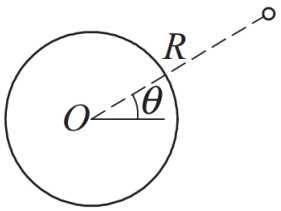
$$\begin{cases} y = -x^2 + x + 0.75 \\ y + 5xy = x^2 \end{cases}$$

تمرین شماره: ۶

مسیر چرخش یک ماهواره حول زمین از رابطه‌ی زیر داده می‌شود:

$$R = \frac{C}{1 + e\sin(\theta + \alpha)}$$

که در این رابطه (R, θ) همانطور که در شکل نشان داده شده است، مختصات قطبی ماهواره و C، e و α ثابت می‌باشند.



اگر ماهواره در ۳ نقطه‌ی جدول زیر مشاهده شده باشد، مشخص کنید ماهواره در چه θای به کمترین R می‌رسد و مقدار R را در این نقطه محاسبه نمایید. همه محاسبات را با روش نیوتن ـ رافسن و تا ۳ رقم معنی‌دار انجام دهید.

θ	$-30°$	$0°$	$30°$
$R(km)$	6870	6728	6615

راهنمایی: با استفاده از جدول داده شده ۳ معادله ایجاد کنید تا ابتدا ۳ ثابت C، e و α را بدست آورید. سپس با روش تحلیلی از R نسبت به θ مشتق بگیرید و مشتق را مساوی صفر

قرار دهید و معادله حاصل را برای θ حل کنید تا θای را پیدا کنید که در آن R کمینه می‌شود.

فصل ۳

دستگاه معادلات خطی

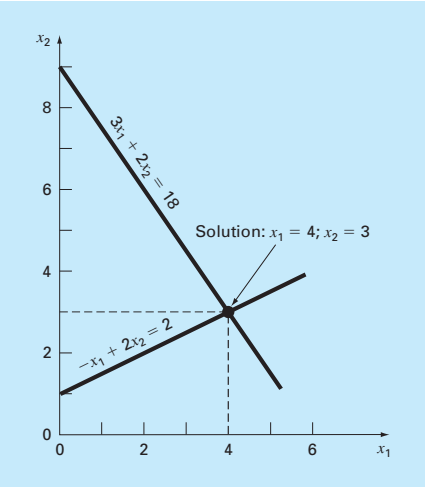
در این فصل به بررسی دستگاه معادلات خطی می‌پردازیم. منظور یک دسته از معادلات جبری است که می‌بایست توام با هم حل شوند و به پاسخ‌هایی دست یابیم که در همه‌ی معادلات مذکور صدق کنند. خطی بودن در اینجا به این معنی است که مجهولات همه از درجه یک می‌باشند. در ساده‌ترین شکل، دو معادله و دو مجهول زیر را در نظر بگیرید:

$$\begin{cases} 3x_1 + 2x_2 &= 18 \\ -x_1 + 2x_2 &= 2 \end{cases} \qquad (۱.۳)$$

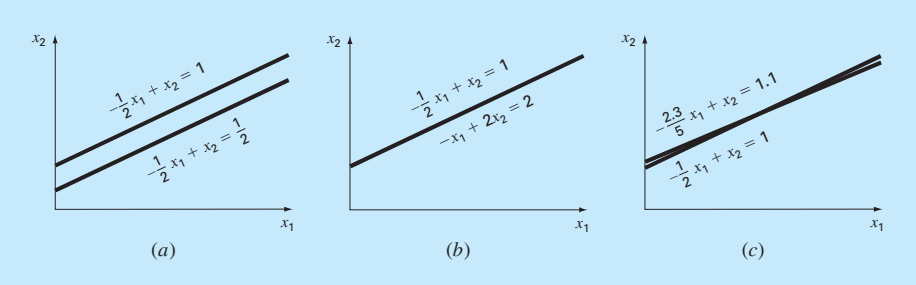
از نظر هندسی هر یک از معادلات معرف یک خط در صفحه‌ی $x_2 - x_1$ می‌باشد. و حل این دستگاه بدست آوردن محل برخورد این خطوط است. در شکل (۱.۳) این دو خط و محل برخورد آن‌ها به عنوان حل دستگاه معادلات (۱.۳) نشان داده شده است. در ابعاد بالاتر مثلا در ۳ معادله و ۳ مجهول، هدف از حل دستگاه پیدا کردن محل برخورد ۳ خط در یک فضا و مختصات ۳ بعدی می‌باشد.

در اینکه خطوط همدیگر را بتوانند در یک نقطه قطع کنند و یا نه، شرایط مختلفی می‌تواند بوجود آید. در شکل (۲.۳) این شرایط برای حالت دو معادله دو مجهول نشان داده شده است. در قسمت (a) دیده می‌شود که دو خط با هم موازی هستند و همدیگر را قطع نمی‌کنند. در نتیجه دستگاه متناظر آن پاسخی ندارد. قسمت (b) دو خط روی هم منطبق هستند. پس دستگاه دارای بی‌نهایت پاسخ است. و در قسمت (c) می‌بینیم که دو خط همدیگر را در یک نقطه قطع می‌کنند اما شیب دو خط بسیار به هم نزدیک است و با کوچکترین تغییری در معادلات، محل تقاطع دو خط جابجایی زیادی خواهد داشت.در نتیجه پاسخ این دستگاه بسیار حساس می‌باشد و خطاهای محاسبات می‌تواند روی پاسخ تاثیر زیادی داشته باشد.

اگر معادلات را به شکل ماتریسی (۲.۳) بنویسیم، این شرایط را از روی ماتریس ضرایب $[A]$ و $[B]$ می‌توانیم تشخیص دهیم. حالت (a) وقتی رخ می‌دهد که سطرهای ماتریس $[A]$ با هم یکسان باشند و اگر سطرهای بردار $[B]$ نیز با هم یکی باشند آنگاه حالت قسمت (b) رخ می‌دهد و دو خط



شکل ۱.۳: مفهوم حل دستگاه معادلات خطی(منبع: کتاب چاپرا)



شکل ۲.۳: مفهوم انواع دستگاه معادلات خطی از نظر داشتن و یا نداشتن پاسخ(منبع: کتاب چاپرا)

روی هم منطبق می‌باشند. در نهایت اگر اختلاف سطرها در هر ستون ماتریس $[A]$ و نیز سطرهای بردار $[B]$ خیلی به هم نزدیک باشند، حالت قسمت (c) بوجود می‌آید. در حالت‌های (a) و (b) دترمینان ماتریس ضرایب $[A]$ صفر می‌شود و در حالت (c) تقریبا صفر می‌شود.

$$\left\{ \begin{array}{rcl} a_{11}x_1 + a_{12}x_2 & = & b_1 \\ a_{21}x_1 + a_{22}x_2 & = & b_2 \end{array} \right. \Rightarrow \left[\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right] . \left[\begin{array}{c} x_1 \\ x_2 \end{array} \right] = \left[\begin{array}{c} b_1 \\ b_2 \end{array} \right]$$

$$\Rightarrow \quad [A] \,.\, [X] = [B] \qquad (۲.۳)$$

برای حل معادله‌ی ماتریسی (۲.۳) کافیست دو طرف تساوی را در ماتریس معکوس ضرایب $[A]$ ضرب نماییم.

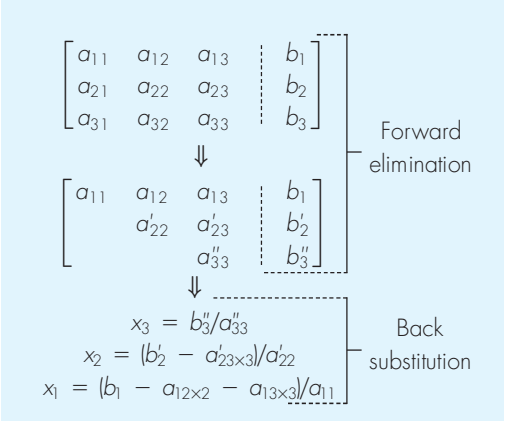
$$[A]^{-1} \,.\, ([A] \,.\, [X]) = [A]^{-1} \,.[B] \Rightarrow [X] = [A]^{-1}.[B] \qquad (۳.۳)$$

پس عمده‌ی مطالب این فصل مربوط به یافتن بردار $[X]$ و نیز ماتریس معکوس $[A]^{-1}$ است. برای حالتی که تعداد معادلات و مجهولات محدود به ۲ می‌باشد، از تکنیک‌های ساده‌ای همچون **روش**

کرامر می‌توانیم استفاده کنیم اما هنگامی با تعداد زیاد معادله و مجهول مواجه هستیم از روش‌های عددی مانند روش حذفی گاوس و روش تکرار گاوس ـ سایدل استفاده می‌کنیم.

۱.۳ روش حذفی گاوس

در این روش با استفاده از عملیات جبری روی سطرهای ماتریس $[A]$ و بردار $[B]$ سعی می‌کنیم ماتریس $[A]$ را بشکل بالا مثلثی تبدیل می‌کنیم و در نهایت بسادگی می‌توانیم بردار $[X]$ را بدست آوریم. شکل (۳.۳) مراحل این عملیات را در یک تصویر کلی نشان می‌دهد. با جزئیات این مراحل بکمک یک مثال آشنا می‌شویم. دستگاه معادلات (۴.۳) را در نظر بگیرید.



شکل ۳.۳: توضیح تصویری از روش حذفی گاوس(منبع: کتاب چاپرا)

$$\begin{cases} 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 &= 19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4 \end{cases} \qquad (۴.۳)$$

ماتریس ضرایب $[A]$ و بردار ثابت $[B]$ را در کنار هم قرار می‌دهیم و یک ماتریس از ترکیب آنها که ستون‌های اول آن از ماتریس ضرایب و ستون آخر از بردار ثابت می‌باشد، ایجاد می‌کنیم. ماتریسی که به این نحو از افزودن دو ماتریس یا بردار ایجاد می‌شود اصطلاحاً **ماتریس افزوده (Augmented Matrix)** گفته می‌شود. اگر سطر اول را در $a_{21}/a_{11} = 0.1/3$ ضرب کنیم و از سطر دوم کم کنیم و جایگذین سطر دوم کنیم درایه‌ی سطر دوم، ستون اول ماتریس $[A]$ صفر خواهد شد. این تکنیک را ادامه می‌دهیم و با ضرب سطر اول در $a_{31}/a_{11} = 0.3/3$ و کم کردن از سطر سوم و جایگذینی در سطر سوم، درایه‌ی سطر سوم و ستون اول هم صفر خواهد شد. حال درایه‌های ماتریس افزوده تغییر پیدا کرده است. این درایه‌ها را با a'_{ij} و b'_i برای درایه‌های متناظر با ماتریس $[A]$ و بردار $[B]$ نمایش می‌دهیم. برای صفر کردن درایه‌ی سطر سوم و ستون دوم باید از سطر دوم ماتریس افزوده‌ی جدید

بجای سطر اول استفاده کنیم. کافیست سطر دوم را در $a'_{32}/a'_{22} = (-0.190)/7.003$ ضرب و از سطر سوم کم و جایگذین سطر سوم کنیم. به این طریق گام به گام ماتریس ضرایب بالا مثلثی می‌شود.

$$\left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & 19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{array}\right]$$

$$\Rightarrow \begin{array}{rcl} R2 & \leftarrow & (R2 - 0.1/3\ R1) \\ R3 & \leftarrow & (R3 - 0.3/3\ R1) \end{array} \Rightarrow \left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0 & 7.003 & -0.293 & -19.562 \\ 0 & -0.190 & 10.020 & 70.615 \end{array}\right]$$

$$\Rightarrow R3 \leftarrow (R3 - (-0.190/7.003)\ R2) \Rightarrow \left[\begin{array}{ccc|c} 3 & -0.1 & -0.2 & 7.85 \\ 0 & 7.003 & -0.293 & -19.562 \\ 0 & 0 & 10.012 & 70.084 \end{array}\right] \quad (۶.۲)$$

همانطور که دیده می‌شود ماتریس ضرایب بالا مثلثی شده و به تناسب ماتریس ثابت‌ها نیز تغییر یافته است. حال از سطر آخر که تنها یک درایه‌ی غیر صفر مربوط به ماتریس ضرایب $[A]$ دارد، شروع می‌کنیم و مجهول مربوط به این سطر را با تقسیم درایه‌ی آخر بردار ثابت $[B]$ بر همان درایه‌ی غیر صفر ماتریس ضرایب $[A]$ بدست می‌آوریم. با داشتن این مجهول، می‌توانیم با استفاده از سطر یکی مانده به آخر، مجهول مربوط به آن سطر و گام به گام تمام مجهول‌ها را محاسبه نماییم. پس برای مثال بالا، مجهولات را به این همین نحو محاسبه می‌نماییم.

$$\begin{aligned} \rightarrow x_3 &= \frac{70.084}{10.012} = 7.000 \\ \rightarrow x_2 &= \frac{-19.562}{7.003} - \frac{-0.293}{7.003}(7.000) = -2.500 \\ \rightarrow x_1 &= \frac{7.85}{3} - \frac{-0.2}{3}(7.000) - \frac{-0.1}{3}(-2.500) = 3.000 \end{aligned} \quad (۶.۳)$$

همانطور که دیده می‌شود این روش بسیار ساده و به راحتی قابل برنامه نویسی و تعمیم به تعداد زیاد معادلات و مجهولات است. اما در عین حال در چند حالت نیز با مشکل جدی مواجه می‌شود. در ضرایبی که برای صفر کردن درایه‌های مثلث پایین استفاده شد، درایه‌ی قطر اصلی در مخرج آمده بود. پس هنگامی که درایه‌های روی قطر اصلی صفر باشند، این روش با مشکل مواجه می‌شود. مشکل بعدی خطای گرد کردن است. از آنجاییکه در این روش محاسبات از مراحل قبلی روی هر مرحله اثر می‌گذارد، در هنگامیکه تعداد معادلات و مجهولات خیلی زیاد است، خطای گرد کردن از هر مرحله به مرحله بعد انتقال و در نهایت با انباشت زیاد خطای گرد کردن مواجه می‌شویم. و اگر معادلات **بد-وضع (Ill-Conditioned)** باشند، یعنی هنگامی که دترمینان ماتریس ضرایب بسیار کوچک است، پاسخ بدست آمده از این روش می‌تواند کاملا اشتباه و دور از ریشه‌ی واقعی باشد.

برای بهبود دادن پاسخ‌ها می‌توانیم سطرهای ماتریس افزوده را در هر مرحله جابجا کنیم تا همواره درایه‌های روی قطر اصلی از بقیه درایه‌ها در ستون خود بزرگتر باشد. به این عمل **محورگیری جزئی** گفته می شود. البته می‌توان ستون‌ها را نیز با هم جابجا نمود تا درایه‌ی روی قطر اصلی علاوه بر بزرگتر بودن از الباقی درایه‌های هم ستون خود، از بقیه‌ی درایه‌ها در سطر خود نیز بزرگتر باشد، که به آن **محورگیری کلی** گفته می‌شود. معمولا محورگیری کلی زیاد استفاده نمی‌شود زیرا هنگام جابجا کردن ستون‌ها باید ترتیب مجهول‌ها را نیز به تناسب ستون‌ها، جابجا نمود و این پیچیدگی برنامه نویسی را زیاد می‌کند. **مقیاس کردن** هر سطر ماتریس افزوده، در پاسخ و ترتیب مجهولات تغییری ایجاد نمی‌کند. پس بجای محورگیری کلی، می‌توان ابتدا هر سطر را چنان مقیاس نمود تا بزرگترین درایه آن سطر عدد ۱ باشد و سپس محورگیری جزئی نمود. در ضمن روش مقیاس کردن بخصوص در مسائلی که واحد سنجش مجهولات متفاوت باشد می‌تواند بسیار مفید باشد. در مثال زیر با نحوه‌ی مقیاس کردن و محورگیری جزئی آشنا می‌شوید.

$$\begin{cases} 2x_1 + 100000x_2 &= 100000 \\ x_1 + x_2 &= 2 \end{cases} \tag{۷.۳}$$

به ظاهر برای ستون اول ماتریس افزوده داریم $a_{11} > a_{21}$. اما در سطر اول a_{11} از بقیه درایه‌ها بزرگتر نیست.لذا ابتدا سطر ۱ و ۲ را مقیاس می‌کنیم. بعد از مقیاس متوجه می‌شویم که باید محورگیری انجام می‌دهیم.

$$\left[\begin{array}{cc|c} 2 & 100000 & 100000 \\ 1 & 1 & 2 \end{array}\right]$$

$$\text{scaling: } R1 \leftarrow 0.00001\,R1 \;\Rightarrow\; \left[\begin{array}{cc|c} 0.00002 & 1 & 1 \\ 1 & 1 & 2 \end{array}\right]$$

$$\text{pivoting: } \begin{array}{l} R1 \leftarrow R2 \\ R2 \leftarrow R1 \end{array} \Rightarrow \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0.00002 & 1 & 1 \end{array}\right]$$

$$\Rightarrow R2 \leftarrow R2 - 0.00002R1 \Rightarrow \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 0.99998 & 0.99996 \end{array}\right]$$

$$\Rightarrow x_2 = 0.99996/0.99998 = 0.99998 \Rightarrow x_1 = 2 - 0.99998 = 1.00002$$

باید دقت داشت که در بعضی موارد مقیاس کردن خود موجب خطای گرد کردن می‌شود. لذا در چنین مواردی سطر را مقیاس می‌کنیم که تشخیص دهیم به محورگیری نیاز می‌باشد و یا خیر اما بلافاصله قبل از محورگیری، مقیاس معکوس می‌کنیم تا دستگاه به ضرایب اول باز گردد.

مثال شماره: ۱

بقای جرم ایجاب می‌کند که در واحد زمان مقدار افزایش جرم سیال در یک مخزن برابر اختلاف جرم خروجی از ورودی به مخزن باشد. در حالت پایدار که جرم درون مخزن ثابت است، جرم خروجی در واحد زمان (نرخ جرم خروجی) با جرم ورودی در واحد زمان (نرخ جرم ورودی) برابر است.

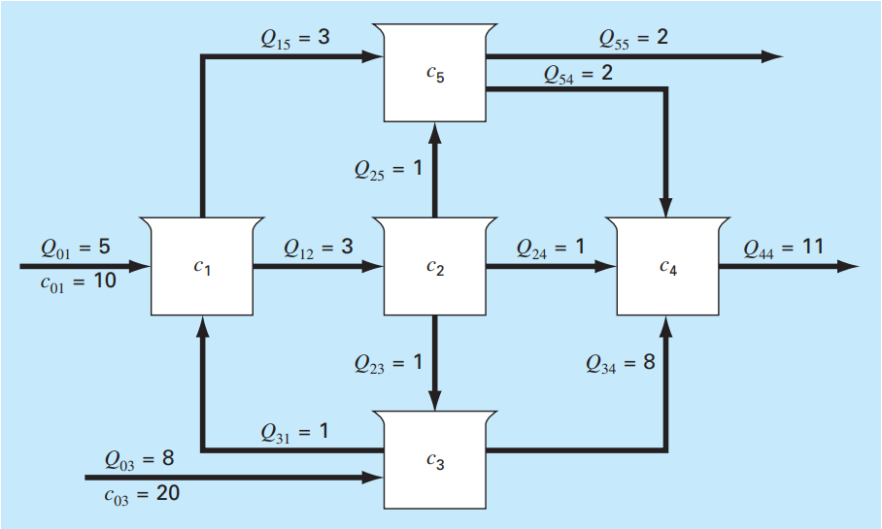
$$\frac{\Delta m_c}{\Delta t} = \frac{\Delta m_{in}}{\Delta t} - \frac{\Delta m_{out}}{\Delta t} = 0 \Rightarrow \frac{\Delta m_{in}}{\Delta t} = \frac{\Delta m_{out}}{\Delta t}$$

$$\dot{m}_{in} = \dot{m}_{out}$$

نرخ جرم جابجا شده‌ی یک سیال ($\dot{m}$) برابر چگالی جرمی (c) در نرخ حجم (Q) جابجا شده‌ی سیال است. پس برای هر مخزن می‌توانیم بنویسیم:

$$\sum \dot{m}_{in} = \sum \dot{m}_{out} \Rightarrow \sum c_{in} Q_{in} = \sum c_{out} Q_{out}$$

در شکل زیر ۵ مخزن شیمیایی نشان داده شده است که نرخ‌های حجمی ورودی و خروجی به مخزن و نیز برای سیال‌های ورودی از بیرون سیستم چگالی حجمی در شکل نشان داده شده است. با استفاده از نوشتن روابط بقای جرم برای هر مخزن، چگالی جرمی هر مخزن را محاسبه نمایید. سیال خروجی از هر مخزن همان چگالی سیال درون مخزن را دارد و سیال ورودی، چگالی سیال مخزنی که از آن خارج شده است.



پاسخ:
همانطور که در شکل دیده می‌شود، بطور مثال سیال با چگالی $c_{01} = 10$ و نرخ حجمی $Q_{01} = 5$ از خارج سیستم و نیز چگالی c_3 و نرخ حجمی $Q_{31} = 1$ از مخزن ۳ به مخزن ۱ وارد و با چگالی c_1 و نرخ حجمی $Q_{12} = 3$ و $Q_{15} = 3$ از مخزن ۱ خارج و به مخزن‌های ۲ و ۵ وارد می‌شود.

$$Q_{01}c_{01} + Q_{31}c_3 = Q_{12}c_1 + Q_{15}c_1 \Rightarrow 50 + c_3 = 3c_1 + 3c_1 \Rightarrow 6c_1 - c_3 = 50$$

با نوشتن روابط برای هر ۵ مخزن معادلات زیر بدست می‌آید.

$$\begin{cases} 6c_1 - c_3 = 50 \\ -3c_1 + 3c_2 = 0 \\ -c_2 + 9c_3 = 160 \\ -c_2 - 8c_3 + 11c_4 - 2c_5 = 0 \\ -3c_1 - c_2 + 4c_5 = 0 \end{cases}$$

این معادلات را بشکل ماتریسی بازنویسی می‌کنیم.

$$\begin{bmatrix} 6 & 0 & -1 & 0 & 0 \\ -3 & 3 & 0 & 0 & 0 \\ 0 & -1 & 9 & 0 & 0 \\ 0 & -1 & -8 & 11 & -2 \\ -3 & -1 & 0 & 0 & 4 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 50 \\ 0 \\ 160 \\ 0 \\ 0 \end{bmatrix}$$

برای حل این دستگاه معادلات خطی از روش حذفی گاوس استفاده می‌کنیم. به این منظور ابتدا چک می‌کنیم عناصر روی محور از الباقی عناصر در ستون خود بزرگتر باشند. همانطور که مشاهده می‌شود، خوشبختانه، همه عناصر روی قطر از عناصر هم ستون در سطرهای پایین‌تر خود بزرگتر هستند و نیاز به محورگیری نیست.

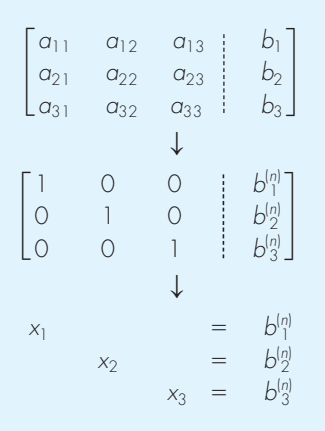
$$\begin{cases} 6 > 3 \\ 3 > 1 \\ 9 > 8 \\ 11 > 0 \end{cases}$$

```
import numpy as np

def upper_triangular_matrix(a,b):
    n = len(b)
    for i in range(1,n):
        for j in range(i):
            lam = a[i,j]/a[j,j]
            a[i] = a[i]-lam*a[j]
            b[i] = b[i]-lam*b[j]
    return a,b

def back_substituation(a,b):
    n = len(b)
    x = b[:]
    for k in range(n-1,-1,-1):
        x[k]=(b[k] - np.dot(a[k,k+1:n],x[k+1:n]))/a[k,k]
    return x

def gauss_elimination(a,b):
    a,b = upper_triangular_matrix(a,b)
    x = back_substituation(a,b)
    return x

a = np.array([[6,0,-1,0,0],
              [-3,3,0,0,0],
              [0,-1,9,0,0],
              [0,-1,-8,11,-2],
              [-3,-1,0,0,4]],dtype=float)

b = np.array([50,0,160,0,0],dtype=float)

c = gauss_elimination(a,b)

print(f'Concentration in the reactors are: \
\nc1: {c[0]:.1f} \nc2: {c[1]:.1f} \nc3: {c[2]:.1f} \
\nc4: {c[3]:.1f} \nc5: {c[4]:.1f}')
```

Concentration in the reactors are:
c1: 11.5
c2: 11.5
c3: 19.1
c4: 17.0
c5: 11.5

۲.۳ روش گاوس–جوردن (Gauss-Jordan) برای معکوس ماتریس

روش گاوس–جوردن کاملا مانند روش حذفی گاوس است با این تفاوت بعد از بالا مثلثی کردن ماتریس ضرایب، مراحل را ادامه می‌دهیم و مانند درایه‌های مثلث پایین که با عملیات جبری بین سطرها صفر شدند، درایه‌های مثلث بالا را نیز صفر کرده و ماتریس ضرایب را قطری می‌کنیم. با این کار مرحله پیدا کردن مجهولات ساده می‌شود و نیاز نیست از آخرین مجهول شروع کنیم و به عقب، یکی یکی مجهولات را بدست آوریم و هر محهولی را مستقل از باقی مجهولات می‌توانیم محاسبه کنیم. شکل (۴.۳) این مراحل را با یک نمای کلی نشان می‌دهد.



شکل ۴.۳: توضیح تصویری از روش گاوس–جوردن (منبع: کتاب چاپرا)

البته می‌توان اثبات کرد که روش گاوس جوردن نسبت به حذفی گاوس تعداد عملیات بیشتری نیاز دارد و در نتیجه خطای گرد کردن بزرگتری بوجود می‌آورد. لذا معمولا از همان روش حذفی گاوس برای حل دستگاه معادلات خطی استفاده می‌کنیم. اما یک کاربرد بسیار خوب روش گاوس–جوردن محاسبه‌ی ماتریس معکوس یک ماتریس می‌باشد. این کاربرد و روش محاسبه را بکمک مثال زیر نشان می‌دهیم. فرض کنید می‌خواهیم معکوس ماتریس $[A]$ را محاسبه کنیم.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{۸.۳}$$

ماتریس افزوده را بکمک ماتریس ضرایب $[A]$ و ماتریس واحد $[I]$ ایجاد می‌کنیم. سپس به روش گاوس‌ـجوردن ماتریس ضرایب را قطری و به ماتریس واحد تبدیل می‌کنیم. بعد از این عملیات ماتریس واحد به ماتریس معکوس $[A]$ تبدیل می‌شود.

$$\left[\, A \,|\, I \,\right] \Rightarrow \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 3 & 4 & 0 & 1 \end{array}\right] \Rightarrow \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & -2 & -3 & 1 \end{array}\right]$$
$$\Rightarrow \left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 0 & 1 & 3/2 & -1/2 \end{array}\right] \Rightarrow \left[\begin{array}{cc|cc} 1 & 0 & -2 & 1 \\ 0 & 1 & 3/2 & -1/2 \end{array}\right]$$

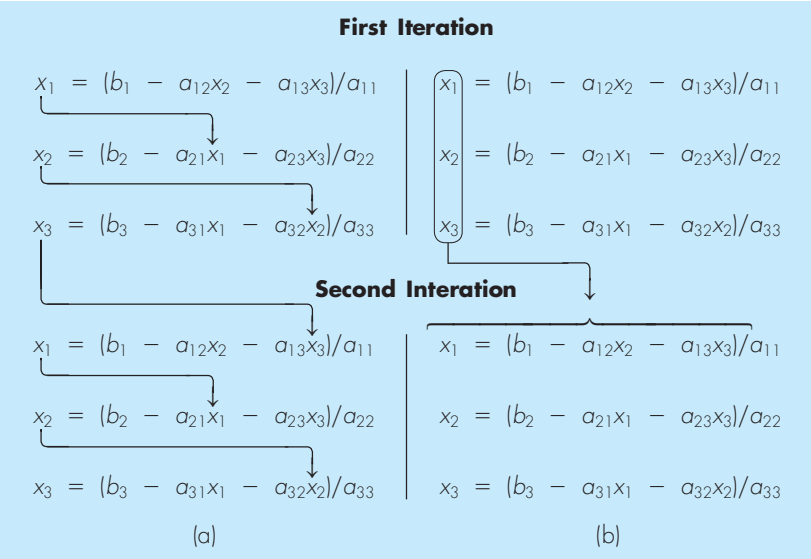
پس ماتریس معکوس $[A]$ می‌شود:

$$[A]^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} \tag{۹.۳}$$

۳.۳ روش گاوس‌ـسایدل (Gauss-Seidel)

با روش گاوس‌ـسایدل که مبتنی بر تکرار است، قبلا در محاسبه‌ی دستگاه معادلات غیرخطی تحت عنوان روش نقطه‌ی ثابت آشنا شدیم. از هر معادله یک مجهول را برحسب مجهولات دیگر محاسبه می‌کنیم. با انتخاب یک مقدار آغازین برای مجهولات در هر گام مقدار جدیدی برای آن‌ها پیدا می‌کنیم. شکل (۵.۳) نمای کلی مراحل گاوس‌ـسایدل (قسمت a) و ژاکوبی (قسمت b) را نشان می‌دهد. همانطور که در تصویر دیده می‌شود، در روش ژاکوبی در هر گام همه مجهولات با استفاده از مقادیر مجهولات در گام قبلی محاسبه می‌شوند و مقادیر جدید که در هر گام محاسبه می‌شود در همان گام برای محاسبه به کار نمی‌رود و برای محاسبه در گام بعدی ذخیره می‌گردد. در حالیکه در روش گاوس‌ـسایدل در هر گام هنگامی که مجهولی مقدار جدید گرفت از همین مقدار برای محاسبه‌ی مجهولات بعدی در همین گامی که در آن قرار داریم، استفاده می‌شود. سرعت روش گاوس‌ـسایدل از روش ژاکوبی بیشتر است.

شرط همگرایی روش گاوس‌ـسایدل از شرط $|g'(x)| < 1$ برای نقطه‌ی ثابت می‌آید و مانند آن قابل اثبات است.بعد از محاسبه‌ی مشتقات و مرتب کردن مقادیر در دو طرف نامساوی، شرط همگرایی روش گاوس‌ـسایدل به شکل زیر ساده می‌شود. این شرط **غالب قطری** نامیده می‌شود. همانطور که دیده می‌شود، روش گاوس‌ـسایدل هنگامی همگرا است که قدرمطلق درایه‌ی روی قطر اصلی از جمع قدرمطلق بقیه درایه‌های هم سطر خود بزرگتر باشد.



شکل ۵.۳: نمای کلی از روش تکراری گاوس–سایدل (a) و ژاکوبی (b)((منبع: کتاب چاپرا)

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \qquad (۱۰.۳)$$

در انتها از مثالی که به روش حذفی گاوس حل نمودیم استفاده می‌کنیم و با روش گاوس–سایدل نیز مقادیر مجهول آن را بدست می‌آوریم. قبل از بدست آوردن هر مجهول از یک معادله، شرط غالب قطری را چک می‌کنیم تا از همگرایی روش اطمینان حاصل کنیم.

مثال شماره: ۲

پاسخ دستگاه معادلات خطی زیر را با روش گاوس–سایدل با دقت ۳ رقم اعشار بدست آورید.

$$\begin{cases} 3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 = -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 = 71.4 \end{cases} \Rightarrow \begin{cases} 3 > 0.1 + 0.2 \\ 7 > 0.1 + 0.3 \\ 10 > 0.3 + 0.2 \end{cases}$$

$$\Rightarrow \begin{cases} x_1 = \frac{1}{3}(0.1x_2 + 0.2x_3 + 7.85) \\ x_2 = \frac{1}{7}(-0.1x_1 + 0.3x_3 - 19.3) \\ x_3 = \frac{1}{10}(-0.3x_1 + 0.2x_2 + 71.4) \end{cases}$$

```python
import matplotlib.pyplot as plt
import numpy as np

x1,x2,x3 = 0,0,0
es = 0.0005
i = 1
print(f" \t",end='')
print(f"x1: {x1:0.3f}\t", end='')
print(f"x2: {x2:0.3f}\t", end='')
print(f"x3: {x3:0.3f}")

while True:
    print(f"iteration: {i}\t",end='')
    er_con = 1

    x1_ = 1/3 * (0.1*x2 + 0.2*x3 + 7.85)
    print(f"x1: {x1_:0.3f}\t", end='')
    if not abs(x1 - x1_) < es:
        er_con = 0
    x1 = x1_

    x2_ = 1/7 * (-0.1*x1 + 0.3*x3 - 19.3)
    print(f"x2: {x2_:0.3f}\t", end='')
    if not abs(x2 - x2_) < es:
        er_con = 0
    x2 = x2_

    x3_ = 1/10 * (-0.3*x1 + 0.2*x2 + 71.4)
    print(f"x3: {x3_:0.3f}")
    if not abs(x3 - x3_) < es:
        er_con = 0
    x3 = x3_

    i += 1
```

```
    if er_con == 1:
        break
```

```
                    x1: 0.000       x2: 0.000       x3: 0.000
iteration: 1        x1: 2.617       x2: -2.795      x3: 7.006
iteration: 2        x1: 2.991       x2: -2.500      x3: 7.000
iteration: 3        x1: 3.000       x2: -2.500      x3: 7.000
iteration: 4        x1: 3.000       x2: -2.500      x3: 7.000
```

تمرین شماره: ۱

پاسخ دستگاه معادلات خطی زیر را با روش حذفی گاوس بدست آورید.

$$\begin{cases} -x_1 & -x_2 & +2x_3 & = & -5 \\ 2x_1 & -2x_2 & +2x_3 & = & 4 \\ 2x_1 & -x_2 & +x_3 & = & -1 \end{cases}$$

تمرین شماره: ۲

پاسخ دستگاه معادلات خطی زیر را با روش گاوس ـ سایدل با دقت ۲ رقم اعشار بدست آورید. مقادیر نقطه‌ی آغازین به انتخاب شما است.

$$\begin{cases} 10x_1 & +2x_2 & -x_3 & = & 27 \\ x1 & +x2 & +5x3 & = & -21.5 \\ -3x_1 & -6x_2 & +2x_3 & = & -61.5 \end{cases}$$

تمرین شماره: ۳

معکوس ماتریس زیر را محاسبه نماید.

$$[A] = \begin{bmatrix} 8 & -2 & -1 & 0 & 0 \\ -2 & 9 & -4 & -1 & 0 \\ -1 & -3 & 7 & -1 & -2 \\ 0 & -4 & -2 & 12 & -5 \\ 0 & 0 & -7 & -3 & 15 \end{bmatrix}$$

تمرین شماره: ۴

بازاء چه مقداری از r ، حل عددی دستگاه معادلات خطی زیر، با روشهاي تکراري همگرا خواهد شد. سپس بازاء $r = 0.125$ پاسخ دستگاه خطی زیر را به روش گاوس_سایدل و حدس اولیه‌ی

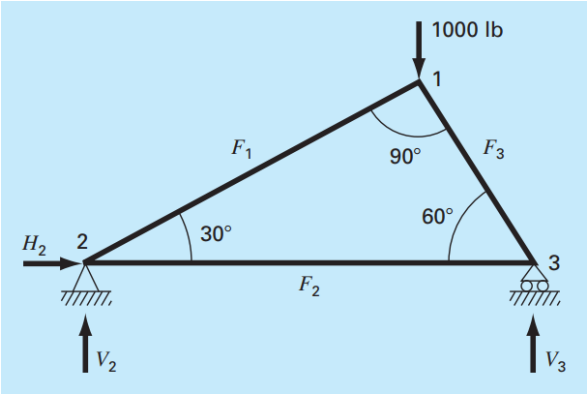
$$x_1, x_2, x_3, x_4 = 1.0, 0.8, 8, 0.00$$

با تقریب ۴ رقم اعشار محاسبه نمایید.

$$[A] = \begin{bmatrix} 1-3r & -r & 0 & 0 \\ -r & 1-4r & -r & 0 \\ 0 & -r & 1-4r & -r \\ 0 & 0 & -r & 1-3r \end{bmatrix} \quad [B] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

تمرین شماره: ۵

در شکل زیر نیروهای $F_1, F_2, F_3, H_2, V_2, V_3$ را با استفاده از **معکوس کردن ماتریس ضرایب** محاسبه نمایید.



راهنمایی: برای تعادل استاتیکی قاب، می‌توانید از معادلات بین نیروهای داده شده روی نقاط ۱ و ۲ و ۳ به ترتیب در راستای افقی و قائم از روابط زیر استفاده کنید.

$$
\begin{cases}
\sum F_H = 0 = F_1 \cos 30^o - F_3 \cos 60^o \\
\sum F_V = 0 = F_1 \sin 30^o + F_3 \sin 60^o - 1000 \\
\\
\sum F_H = 0 = -F_2 - F_1 \cos 30^o + H_2 \\
\sum F_V = 0 = -F_1 \sin 30^o + V_2 \\
\\
\sum F_H = 0 = F_2 + F_3 \cos 60^o \\
\sum F_V = 0 = -F3 \sin 60^o + V_3
\end{cases}
$$

فصل ۴

برازش منحنی

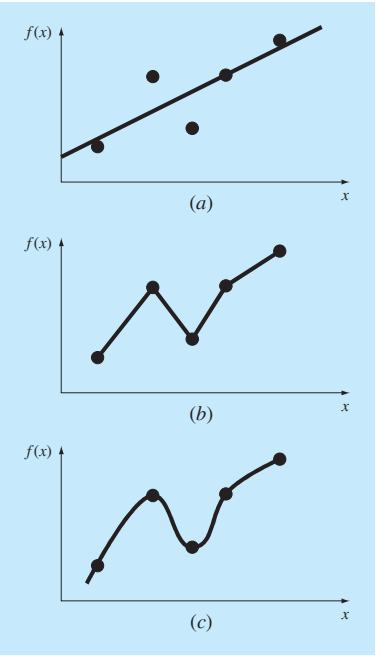
فرض می‌کنیم n داده به عنوان نتیجه‌ی یک آزمایش و یا حاصل یک محاسبه‌ی پیچیده داریم. انجام آزمایش و محاسبات مجدد بدلیل هزینه‌ی بالا امکان پذیر نیست و در عین حال ما نیاز به داده‌های بیشتر در بازه‌ی آزمایش داریم. و یا نیاز به شکل توابع تحلیلی این داده‌ها برای عملیات و استنباط‌های ریاضی داریم. در شکل (۱.۴) سه رویکرد مختلف در برخورد با این مساله نشان داده شده است.

- **رگرسیون (Regression):** در نگاه اول فرض می‌کنیم داده‌های موجود دارای خطا هستند اما مدل ریاضی که توصیف کننده‌ی داده‌ها هستند در اختیار است و تنها باید پارامترهای مدل از روی داده‌های موجود مشخص شوند، شکل (۱.۴ ـ الف).
- **درونیابی خطی:** در نگاه دوم برای داده‌ها خطایی در نظر نمی‌گیریم و فرض می‌کنیم داده‌ها در حد کفایت دقیق هستند و در ضمن مدل ریاضی برای داده‌ها در اختیار نداریم، شکل (۱.۴ ـ ب). در این حالت در ساده‌ترین روش، فاصله بین داده‌ها را با خط تقریب می‌زنیم.
- **درونیابی چندجمله‌ای:** در نهایت در صورتی‌که داده‌ها در حد کفایت دقیق می‌باشند و البته مدلی در اختیار نداریم، یک چندجمله‌ای درجه‌ی $(n-1)$ به عنوان مدل برای داده‌ها درنظر می‌گیریم و ضرایب چندجمله‌ای را چنان تعیین می‌کنیم که داده‌های موجود در چندجمله‌ای صدق نمایند، شکل (۱.۴ ـ ج).

۱.۴ رگرسیون (Regression) با استفاده از حداقل مربعات

در نظر بگیرید مجموعه‌ای از n داده‌ی دو تایی $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ را در اختیار داریم. و مدلی که این داده‌ها را توصیف می‌کند، یک چندجمله‌ای درجه‌ی m است.

$$y = f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \qquad (۱.۴)$$



شکل ۱.۴: اشکال مختلف درون‌یابی و برازش منحنی.

برازش تابع مذکور $y = f(x)$ بر روی داده‌های موجود به این معنی است که ضرایب چند جمله‌ای را به نحوی پیدا کنیم که بازاء هر x_i در داده‌ها مدل منجر به y_i شود. اما از آنجاییکه بر روی داده‌ها خطا وجود دارد انتظار نداریم خروجی مدل و داده‌ی اندازه‌گیری شده‌ی y_i دقیقا با هم برابر شوند. لذا ضرایب را به نحوی پیدا می‌کنیم که مجموع مربع اختلاف این دو مقدار، بر روی تمام داده‌ها، حداقل شود (روش حداقل مربعات).

$$S_r = \sum_{i=1}^{n} \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_m x_i^m)\right)^2 \qquad (۲.۴)$$

برای پیدا کردن ضرایبی که بازاء آنها تابع S_r حداقل شود، باید از این تابع نسبت به ضرایب مشتق گرفت و این مشتق را مساوی صفر قرار داد. از آنجاییکه $m + 1$ ضریب مجهول وجود دارد

$(\{a_0, \ldots, a_m\})$ با مشتق نسبت به آنها $m+1$ معادله نیز بوجود می‌آید.

$$\begin{aligned}
\frac{\partial S_r}{\partial a_0} &= -2\sum_{i=1}^{n}\left(y_i - (a_0 + a_1x_i + a_2x_i^2 + \cdots + a_mx_i^m)\right) = 0 \\
\frac{\partial S_r}{\partial a_1} &= -2\sum_{i=1}^{n} x_i\left(y_i - (a_0 + a_1x_i + a_2x_i^2 + \cdots + a_mx_i^m)\right) = 0 \\
\frac{\partial S_r}{\partial a_2} &= -2\sum_{i=1}^{n} x_i^2\left(y_i - (a_0 + a_1x_i + a_2x_i^2 + \cdots + a_mx_i^m)\right) = 0 \qquad (۳.۴) \\
&\vdots
\end{aligned}$$

با ساده کردن این $m+1$ معادله به دستگاه معادلات خطی زیر می‌رسیم. با استفاده از روش‌های ارائه شده در فصل دستگاه معادلات خطی (فصل: ۳) می‌توانیم $m+1$ ضریب چندجمله‌ای (۱.۴) را بدست آورد.

$$\begin{array}{llll}
(n)a_0 & +(\sum x_i)a_1 & +(\sum x_i^2)a_2 & +\ldots = \sum y_i \\
(\sum x_i)a_0 & +(\sum x_i^2)a_1 & +(\sum x_i^3)a_2 & +\ldots = \sum x_iy_i \\
(\sum x_i^2)a_0 & +(\sum x_i^3)a_1 & +(\sum x_i^4)a_2 & +\ldots = \sum x_i^2y_i \\
 & & \vdots &
\end{array} \qquad (۴.۴)$$

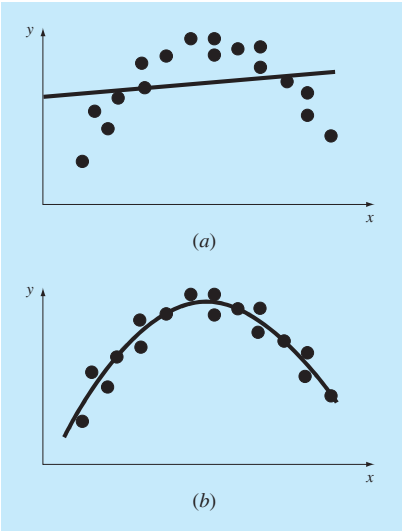
بعد از محاسبه‌ی ضرایب مدل، باید بررسی شود که آیا مدل استفاده شده به خوبی می‌تواند داده‌ها را توصیف نماید یا خیر. برای این منظور پارامتری که **ضریب همبستگی (Correlation Coefficient)** نامیده می‌شود و با رابطه‌ی زیر داده می‌شود را محاسبه می‌کنیم.

$$r^2 = \frac{S_t - S_r}{S_t} \qquad (۵.۴)$$

که در آن S_r مجموع مربع اختلاف مدل و داده‌ها است که با رابطه‌ی (۲.۴) داده می‌شود و S_t مجموع مربع اختلاف متغیر وابسته‌ی داده‌ها y_i از میانگین آن‌ها $\bar{y}$ می‌باشد:

$$S_t = \sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad (۶.۴)$$

مقدار ضریب همبستگی r^2 می‌تواند از 0 تا 1 تغییر کند. عدد 0 به این معنی است که مدل قادر به توصیف داده‌ها نیست و مقدار 1 معرف یک مدل مناسب برای توصیف داده‌ها می‌باشد. بطور



شکل ۲.۴ : برازش دو مدل نامناسب (خطی) و مناسب (سهمی) برای توصیف داده‌ها.

مثال برای توصیف داده‌های نشان داده شده در شکل (۲.۴) از دو مدل متفاوت (خطی و سهمی) استفاده شده است. همانطور که دیده می‌شود مدل خطی قادر به توصیف داده‌ها نمی‌باشد. پس ضریب همبستگی r^2 آن از مدل سهمی بسیار کوچکتر و به صفر نزدیک است.

مثال شماره: ۱

فرض کنید در یک آزمایش ۶ مقدار مستقل x و وابسته‌ی y اندازه‌گیری $(n = 6)$ و این اندازه‌گیری‌ها در جدول زیر آورده شده است. همچنین می‌دانیم مدل و رابطه‌ی بین متغیرهای مستقل و وابسته در این آزمایش سهمی (چندجمله‌ای درجه ۲) است. ضرایب این چند جمله‌ای درجه ۲ را بدست آورید.

5	4	3	2	1	0	x_i
61.1	40.9	27.2	13.6	7.7	2.1	y_i

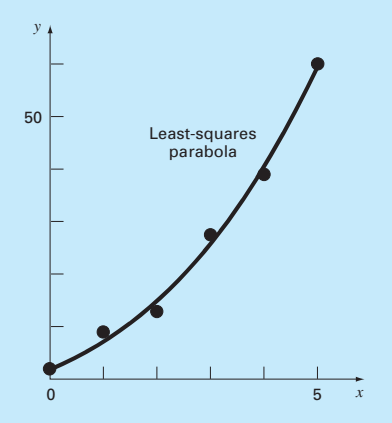
پاسخ: برای بدست آوردن ضرایب، همانطور که در متن توضیح داده شد باید مربع اختلاف مدل و داده، حداقل شود که با توجه به معادلات (۴.۴) جدول زیر را توانهای مختلف و مورد نیاز متغیرهای مستقل x و وابسته‌ی y تشکیل می‌دهیم و با استفاده از جمع ستون‌ها دستگاه سه معادله و سه مجهول مورد نظر را ایجاد می‌کنیم:

x	y	x^2	x^3	x^4	x * y	x^2 * y
0	2.1	0	0	0	0	0
1	7.7	1	1	1	7.7	7.7
2	13.6	4	8	16	27.2	54.4
3	27.2	9	27	81	81.6	244.8
4	40.9	16	64	256	163.6	654.4
5	61.1	25	125	625	305.5	1527.5
sum:	15	152.6	55	225	979	585.6	2488.8
num:	6					

$$\begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 152.6 \\ 585.6 \\ 2488.8 \end{Bmatrix}$$

در نهایت با حل این دستگاه ضرایب a_1, a_0 و a_2 را بدست می‌آوریم.

$$y = a_0 + a_1 x + a_2 x^2 = 2.47857 + 2.35929x + 1.86071x^2$$



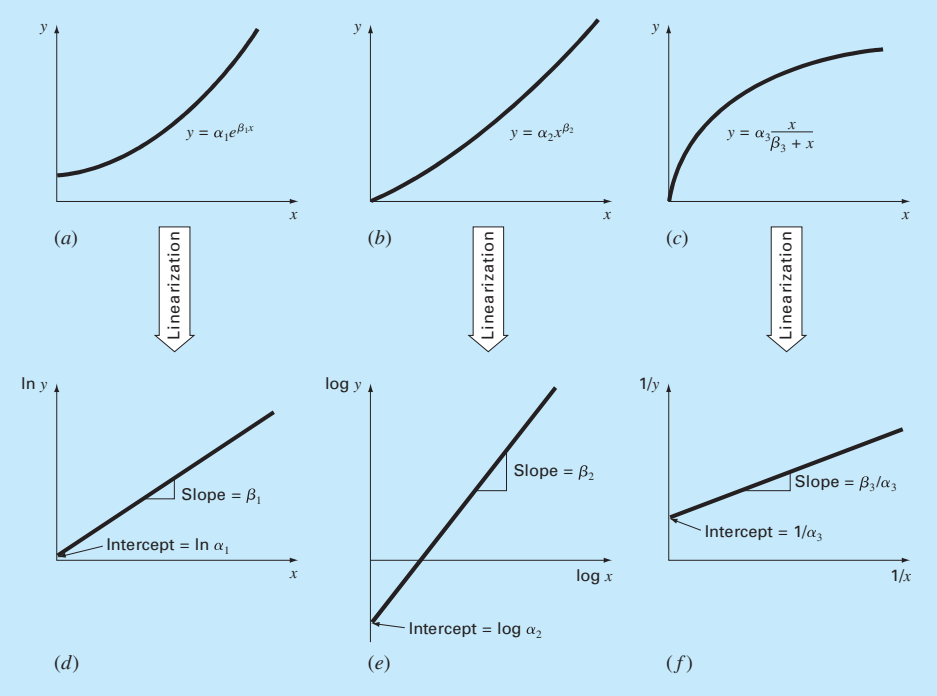
برای اطمینان از کیفیت برازش منحنی بدست آمده مقدار r^2 را محاسبه می‌کنیم.

x_i	y_i	$(y_i - \bar{y})^2$	$(y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$
0	2.1	544.44	0.14332
1	7.7	314.47	1.00286
2	13.6	140.03	1.08158
3	27.2	3.12	0.80491
4	40.9	239.22	0.61951
5	61.1	1272.11	0.09439
Σ	152.6	2513.39	3.74657

$$r^2 = \frac{2513.39 - 3.74657}{2513.39} = 0.99851$$

همانطور که دیده می‌شود ضریب همبستگی r^2 برای برازش بسیار به عدد ۱ نزدیک است و این نشان می‌دهد که مدل بخوبی داده‌ها را توصیف می‌کند.

مدل‌های چندجمله‌ای از هر توان mی که باشند، ضرایب آن‌ها از توان 1 و خطی هستند و معادلات بکار رفته برای بدست آوردن ضرایب منجر به حل دستگاه‌های معادلات خطی می‌شوند. اما همه‌ی داده قابل توصیف با مدل‌های چندجمله‌ای نیستند و باید با مدل‌های پیچیده‌تر و غیرخطی توصیف و بیان شوند. و در حالت کلی پارامترهای این مدل‌ها نیز غیرخطی خواهند بود. که برای بدست آوردن آن‌ها باید از روش‌های حل دستگاه معادلات غیرخطی استفاده کرد. بطور مثال اگر داده‌های یک آزمایش رفتار نمایی داشته باشند و با مدل $y = a\exp(bx)$ بیان شوند، مساوی صفر قرار دادن مشتقات S_r نسبت به a و b منجر به دستگاه غیرخطی می‌شود. اما می‌توان با تبدیل‌هایی بعضی از این معادلات را خطی نمود و با تبدیل x و y مدل را به شکل $Y = A + BX$ تبدیل نمود. بطور مثال شکل (۳.۴) سه نوع از مدل‌هایی که قابل خطی شدن می‌باشند را نشان می‌دهد.



شکل ۳.۴: خطی‌سازی مدل‌های غیرخطی با استفاده از تبدیل متغیرهای مستقل و وابسته.

معادلات (۷.۴) تبدیل‌های مورد نیاز برای خطی کردن سه مدل غیرخطی در شکل (۳.۴) را نشان می‌دهد. همانطور که دیده می‌شود برای مدل‌های نمایی و توانی با لگاریتم گرفتن و در مدل اشباع نرخ رشد با معکوس کردن، مدل‌ها خطی می‌شوند.

$$\begin{array}{lll} \text{Exponential:} & y = ae^{bx} & \rightarrow \ \ln y = \ln a + bx \\ \text{Power:} & y = ax^{b} & \rightarrow \ \ln y = \ln a + b \ln x \\ \text{Saturation Growth Rate:} & y = a\frac{x}{b+x} & \rightarrow \ \frac{1}{y} = \frac{1}{a} + (\frac{b}{a})\frac{1}{x} \end{array} \qquad (۷.۴)$$

مثال شماره: ۲

میزان غلظت دارو در جریان خون بر حسب زمان به خوبی با مدل زیر بیان می‌شود.

$$y = ate^{bt}$$

که در آن زمان برحسب ساعت از لحظه‌ی ورود دارو به جریان خون و میزان غلظت برحسب نانوگرم بر میلی‌لیتر اندازه‌گیری می‌شود. در یک آزمایش غلظت داروی نورفلکستین (Nor-fluoxetine) در جریان خون یک بیمار در هر ساعت اندازه‌گیری و در جدول زیر داده شده است.

hour	concentration (ng/ml)
1	8.0
2	12.3
3	15.5
4	16.8
5	17.1
6	15.8
7	15.2
8	14.0

نیمه عمر این دارو در جریان خون بیمار را محاسبه نمایید. نیمه عمر دارو مدت زمانی است که طول می‌کشد تا سطح غلظت دارو به نصف مقدار بیشینه‌ی آن در جریان خون می‌رسد.

پاسخ: برای خطی‌سازی ابتدا از دو طرف رابطه لگاریتم می‌گیریم:

$$\begin{aligned} y = ate^{bt} \ \rightarrow \ \ln y &= \ln a + \ln t + bt \\ \ln y - \ln t &= bt + \ln a \end{aligned}$$

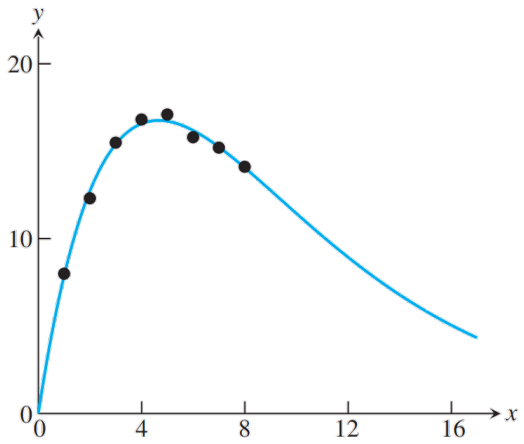
$$\begin{cases} Y &= \ln y - \ln t \\ X &= t \\ a' &= \ln a \end{cases} \rightarrow \ Y = bX + a'$$

t	y	Y = ln y - ln t	X	X^2	X Y
1.00	8.00	2.08	1.00	1.00	2.08
2.00	12.30	1.82	2.00	4.00	3.63
3.00	15.50	1.64	3.00	9.00	4.93
4.00	16.80	1.44	4.00	16.00	5.74
5.00	17.10	1.23	5.00	25.00	6.15
6.00	15.80	0.97	6.00	36.00	5.81
7.00	15.20	0.78	7.00	49.00	5.43
8.00	14.00	0.56	8.00	64.00	4.48
sum:		10.51	36.00	204.00	38.24
num:	8.00				

$$\begin{pmatrix} 8. & 36. \\ 36. & 204. \end{pmatrix} \cdot \begin{pmatrix} b \\ a' \end{pmatrix} = \begin{pmatrix} 10.51 \\ 38.24 \end{pmatrix} \rightarrow \begin{pmatrix} a' \\ b \end{pmatrix} = \begin{pmatrix} 2.28 \\ -0.215 \end{pmatrix}$$
$$a = e^{a'} = 9.78$$

بعد از محاسبه‌ی پارامترها منحنی مدل به شکل زیر بدست می‌آید.

$$y = 9.78\, t\, e^{-0.215\, t}$$



برای پیدا کردن نیمه عمر ابتدا بیشینه منحنی را پیدا می‌کنیم و سپس زمانی که به نصف این مقدار می‌رسد را $\tau = 12.46$ ساعت بدست می‌آوریم.

$$\frac{d}{dt}(y) = 9.78e^{-0.215t} - 2.1027e^{-0.215t}t = 0$$
$$\rightarrow \begin{cases} t = 4.65 \\ y_{max} = 16.73 \end{cases} \rightarrow \tau = 12.46\ (h)$$

مثال شماره: ۳

جدول زیر میزان گاز دی‌اکسیدکربن برحسب میکرومول موجود در یک مول هوا که در انتهای هر ماه در یک مرکز مطالعاتی اندازه‌گیری شده است را از ژانویه سال ۲۰۱۷ تا انتهای سال ۲۰۱۹ نشان می‌دهد. از آنجاییکه میزان دی‌اکسیدکربن در طول سال دارای نوسان و در عین‌حال در کل در حال افزایش است، می‌توانیم با معادله‌ی زیر میزان گاز دی‌اکسیدکربن موجود در هوا نسبت به زمان را مدل نمود:

$$y = c_1 + c_2 t + c_3 \sin(2\pi t)$$

که در آن $c_1\, c_2\, c_3$ ثابت‌هایی هستند که با برازش منحنی روی داده‌های جدول بدست خواهند آمد.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2017	400.0	400.9	402.2	402.5	402.1	401.6	400.8	400.0	398.7	398.4	399.1	399.8
2018	400.8	401.9	403.0	403.4	403.1	402.7	401.9	401.1	400.2	399.9	399.9	400.7
2019	401.5	402.6	403.9	404.1	403.8	403.8	403.6	402.2	401.4	400.8	400.8	401.1

پاسخ:
ابتدا مدل پیشنهاد شده را بر روی داده‌ها برازش می‌کنیم. مدل نسبت به ضرایب کاملا خطی هستند، پس نیازی به خطی سازی نیست. با مشتق گرفتن نسبت به ضرایب می‌توانیم معادلات لازم را بوجود آوریم تا از طریق آنها ضرایب مجهول را محاسبه نماییم.

$$y = c_1 + c_2 t + c_3 \sin(2\pi t) \qquad \Rightarrow \qquad S = \sum (y_i - (c_1 + c_2 t_i + c_3 \sin(2\pi t_i)))^2$$

$$\partial_{c_1} S = \sum -2(y_i - (c_1 + c_2 t_i + c_3 \sin(2\pi t_i))) = 0$$

$$\partial_{c_2} S = \sum -2t_i(y_i - (c_1 + c_2 t_i + c_3 \sin(2\pi t_i))) = 0$$

$$\partial_{c_3} S = \sum -2\sin(2\pi t_i)(y_i - (c_1 + c_2 t_i + c_3 \sin(2\pi t_i))) = 0$$

$$\begin{bmatrix} N & \sum t_i & \sum \sin(2\pi t_i) \\ \sum t_i & \sum t_i^2 & \sum t_i \sin(2\pi t_i) \\ \sum \sin(2\pi t_i) & \sum t_i \sin(2\pi t_i) & \sum \sin(2\pi t_i)^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum t_i y_i \\ \sum \sin(2\pi t_i) y_i \end{bmatrix}$$

با استفاده از داده‌ها درایه‌های ماتریس ضرایب و ماتریس ثابت‌ها را محاسبه می‌کنیم.

$$\begin{cases} N = 36 \\ \sum t_i = 55.50 \\ \sum \sin(2\pi t_i) = 0.00 \\ \sum t_i^2 = 112.54 \\ \sum t_i \sin(2\pi t_i) = -5.60 \\ \sum \sin(2\pi t_i)^2 = 18.00 \\ \sum y_i = 14850.3 \\ \sum t_i y_i = 22942.55 \\ \sum \sin(2\pi t_i) y_i = 6.90 \end{cases} \Rightarrow \begin{bmatrix} 36.00 & 55.50 & 0.00 \\ 55.50 & 112.54 & -5.60 \\ 0.00 & -5.60 & 18.00 \end{bmatrix} \begin{bmatrix} c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} 14850.30 \\ 22942.55 \\ 6.90 \end{bmatrix}$$

و در نهایت با حل معادلات می‌توانیم ضرایب مجهول را بدست آوریم:

$$\Rightarrow \quad \begin{bmatrix} c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} 409.42 \\ 2.00 \\ 1.01 \end{bmatrix}$$

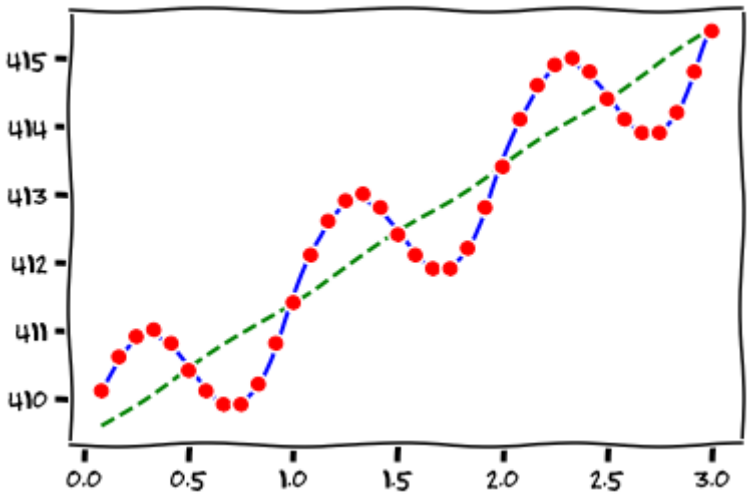
در ادامه برای محاسبه‌ی نرخ افزایش CO_2 کافیست یک خط بر روی داده‌ها برازش دهیم.

$$\begin{cases} N = 36 \\ \sum t_i = 55.50 \\ \sum t_i^2 = 112.54 \\ \sum y_i = 14850.3 \\ \sum t_i y_i = 22942.55 \end{cases} \Rightarrow \begin{bmatrix} 36.00 & 55.50 \\ 55.50 & 112.54 \end{bmatrix} \begin{bmatrix} c1 \\ c2 \end{bmatrix} = \begin{bmatrix} 14850.30 \\ 22942.55 \end{bmatrix}$$

$$\Rightarrow \quad \begin{bmatrix} c1 \\ c2 \end{bmatrix} = \begin{bmatrix} 409.42 \\ 2.00 \end{bmatrix}$$

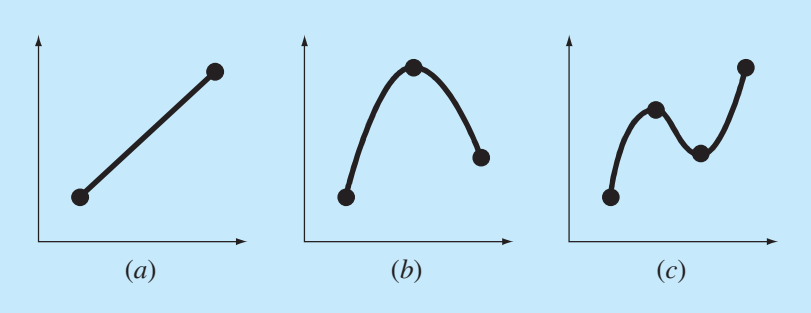
از آنجاییکه میانگین تابع sin در هر مضربی از دوره تناوب آن صفر می‌شود، عملا در برازش خط روی داده‌ها همان قسمت خطی مدل بدست می‌آید. نرخ افزایش CO_2 شیب این خط برازش شده است.

$$\frac{\Delta CO_2}{\Delta t} = 2.00 \ (\mu mole/mole/year)$$


415
414
413
412
411
410
0.0
0.5
1.0
1.5
2.0
2.5
3.0

۲.۴ درونیابی (Interpolation)

همانطور که در ابتدای این فصل آورده شد، هدف ما بیان داده‌ها از طریق یک تابع تحلیلی است. در بخش قبل شکل کلی این تابع (مدل) را می‌دانستیم و از روی داده‌های پارامت‌های مجهول مدل را محاسبه نمودیم. و در این بخش فرض می‌کنیم مدل برای ما شناخته شده نیست، لذا از یک چندجمله‌ای بجای مدل استفاده می‌کنیم. و در ضمن فرض ما بر این است که خطای داده‌ها قابل صرفنظر کردن است و دقت استفاده شده در جمع‌آوری داده‌ها در حد کفایت مساله ما می‌باشد. لذا تمام داده‌ها در مدل صدق می‌کنند و منحنی مدل از نقاط داده‌ها عبور می‌کند. درجه چند جمله‌ای که به عنوان مدل انتخاب کرده‌ایم بستگی به تعداد نقاطی دارد که تصمیم داریم در درون‌یابی درگیر نماییم. در شکل (؟؟) بطور مثال درونیابی منحنی با استفاده از ۲ و ۳ و ۴ نقطه نشان داده شده است. همانطور که دیده می‌شود از ۲ نقطه فقط یک خط می‌توان عبور داد و نیز از ۳ نقطه یک منحنی درجه ۲ و به همین نحو از n نقطه یک منحنی درجه‌ی $n-1$ عبور می‌کند.



شکل ۴.۴: درون‌یابی با استفاده از ۲ و ۳ و ۴ نقطه

برای محاسبه‌ی ضرایب چند جمله‌ای در این بخش از دو روش نیوتن و لاگرانژ استفاده می‌کنیم.

۱.۲.۴ روش تفاضلات تقسیم‌شده‌ی نیوتن

در این روش تابع $f(x)$ را با چند جمله‌ای درجه‌ی n تقریب می‌زنیم و این چندجمله‌ای را نیز به شکل معادله‌ی (۸.۴) بازنویسی می‌کنیم.

$$\begin{aligned} f(x) \approx P_n(x) \quad &= a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \\ &\cdots + a_n(x-x_0)(x-x_1)\dots(x-x_{n-1}) \end{aligned} \tag{۸.۴}$$

که در آن نقاط $\{(x_0, y_0 = f(x_0)), (x_1, y_1 = f(x_1)), \dots, (x_n, y_n = f(x_n))\}$ از نتایج یک آزمایش و محاسبه‌ی دقیقتر دانسته فرض می‌شوند. تمام جملات این چندجمله‌ای در نقطه‌ی x_0 صفر خواهند شد و فقط ضریب a_0 باقی می‌ماند.

$$a_0 = f(x_0)$$

و در نقطه‌ی x_1 فقط دو جمله‌ی اول باقی می‌ماند که در آن هم فقط a_1 مجهول است و به سادگی قابل محاسبه می‌باشد.

$$f(x_1) = a_0 + a_1(x_1 - x_0) \rightarrow a_1 = \frac{f(x_1) - a_0}{(x_1 - x_0)} = \frac{f(x_1) - f(x_0)}{(x_1 - x_0)}$$

همانطور که دیده می‌شود برای محاسبه‌ی a_0 به یک نقطه و برای محاسبه‌ی a_1 به نقطه‌ی دوم نیاز داریم و به این ترتیب برای محاسبه‌ی ضریب a_n به تعداد $n+1$ نقطه نیاز داریم که با یک عملیات جبری روی این نقاط و مقادیر تابع در این نقاط، ضرایب مذکور محاسبه می‌شوند.

$$\begin{aligned} a_0 &= f[x_0] \\ a_1 &= f[x_1, x_0] \\ &\vdots \\ a_n &= f[x_n, \ldots, x_0] \end{aligned}$$

عملگرهای $f[\ldots]$ را **تفاضلات تقسیم‌شده‌ی محدود (Finite Divided Difference)** می‌نامیم. و این عملگرها را به شکل زیر تعریف می‌کنیم:

$$\begin{aligned} a_0 &= f[x_0] = f(x_0) \\ a_1 &= f[x_1, x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\ a_2 &= f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} \\ &\vdots \\ a_n &= f[x_n, x_{n-1}, \ldots, x_0] = \frac{f[x_n, \ldots, x_1] - f[x_{n-1}, \ldots, x_0]}{x_n - x_0} \end{aligned} \tag{۹.۴}$$

همانطور که دیده می‌شود، با استفاده از این عملگر ، عملیات پیدا کردن ضرایب چندجمله‌ای (۸.۴) به شکل کلی و ساده‌ای خلاصه می‌شود. و چندجمله‌ای را می‌توان به شکل زیر نوشت:

$$\begin{aligned} f(x) \approx P_n(x) \; &= f[x_0] + f[x_1, x_0](x - x_0) + f[x_2, x_1, x_0](x - x_0)(x - x_1) + \\ &\cdots + f[x_n, x_{n-1}, \ldots, x_0](x - x_0)(x - x_1)\ldots(x - x_{n-1}) \end{aligned} \tag{۱۰.۴}$$

در جدول نشان داده شده در شکل (۵.۴) ترتیب محاسبه‌ی عملگرها تا مرتبه‌ی ۳ نشان داده شده است.

i	x_i	$f(x_i)$	First	Second	Third
0	x_0	$f(x_0)$	$f[x_1, x_0]$	$f[x_2, x_1, x_0]$	$f[x_3, x_2, x_1, x_0]$
1	x_1	$f(x_1)$	$f[x_2, x_1]$	$f[x_3, x_2, x_1]$	
2	x_2	$f(x_2)$	$f[x_3, x_2]$		
3	x_3	$f(x_3)$			

شکل ۵.۴: در این جدول نحوه‌ی وابستگی هر عملگر به عملگرهای مرتبه‌ی پایین‌تر نشان داده شده است.

مثال شماره: ۴

با استفاده از نقاط $\{x_0, x_1, x_2, x_3\} = \{1, 4, 6, 5\}$ تابع $\ln$ را با یک چندجمله‌ای درجه‌ی سه تقریب بزنید و مقدار تقریبی $\ln 2 = 0.693147$ را بکمک چندجمله‌ای بدست آمده، محاسبه نمایید.

پاسخ:

$$f[x_1, x_0] = \frac{1.386294 - 0}{4 - 1} = 0.4620981$$

$$f[x_2, x_1] = \frac{1.791759 - 1.386294}{6 - 4} = 0.2027326$$

$$f[x_3, x_2] = \frac{1.609438 - 1.791759}{5 - 6} = 0.1823216$$

$$f[x_2, x_1, x_0] = \frac{0.2027326 - 0.4620981}{6 - 1} = -0.05187311$$

$$f[x_3, x_2, x_1] = \frac{0.1823216 - 0.2027326}{5 - 4} = -0.02041100$$

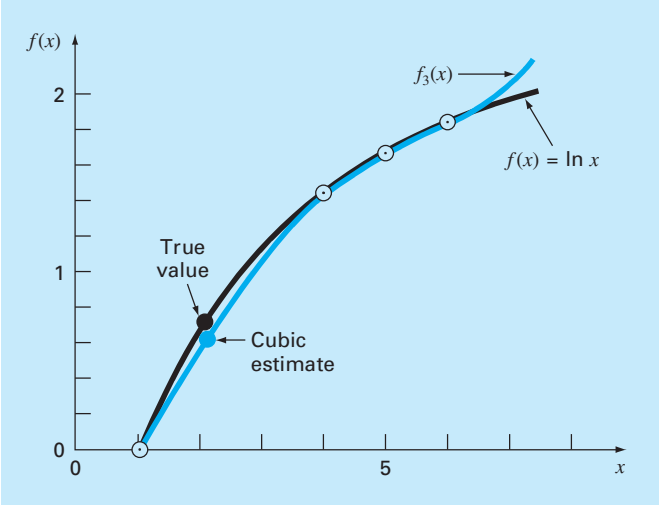
$$f[x_3, x_2, x_1, x_0] = \frac{-0.02041100 - (-0.05187311)}{5 - 1} = 0.007865529$$

$$f_3(x) = 0 + 0.4620981(x - 1) - 0.05187311(x - 1)(x - 4) + 0.007865529(x - 1)(x - 4)(x - 6)$$

$$f_3(2) = 0.6287686$$

x0	1	0	0.462098	-0.05187	0.007866
x1	4	1.386294	0.202733	-0.02041	
x2	6	1.791759	0.182322		
x3	5	1.609438			

با استفاده از شکل (۵.۴) جدول بالا را ایجاد و در نهایت ردیف اول چهار ستون آخر ضرایب چندجمله‌ای درجه‌ی ۳ را مشخص می‌کند. در شکل زیر مقدار واقعی و تقریبی $\ln 2$ نشان داده شده است.



ساختار رابطه‌ی (۱۰.۴) شبیه به بسط تیلور است. با مقایسه‌ی بسط تیلور با این رابطه پی‌می‌بریم که عملگر دو نقطه‌ای $f[x_1, x_0]$ مانند مشتق مرتبه‌ی اول و به همین ترتیب عملگر n نقطه‌ای $f[x_{n-1}, x_{n-2}, \ldots, x_0]$ مانند مشتق مرتبه‌ی nام تابع $f(x)$ در بسط تیلور عمل می‌کنند. همانطور که قبلا دیده‌ایم، خطای برشی بسط تیلور مرتبه‌ی nام از رابطه‌ی زیر بدست می‌آید.

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x_{i+1} - x_i)^{n+1}$$

که از درجه‌ی $n+1$ام و در آن ξ نقطه‌ای بین نقاط x_i و x_{i+1} است. به تناظر آن برای خطای دورنیابی از درجه‌ی nام می‌توانیم رابطه‌ی زیر را بنویسیم:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n)$$

که در آن ξ نقطه‌ای در بین داده‌ها است. فرمول جایگزین برای این خطا رابطه‌ی زیر است.

$$R_n = f[x, x_n, x_{n-1}, \ldots, x_0](x - x_0)(x - x_1)\ldots(x - x_n)$$

که در آن $f[x, x_n, \ldots, x_0]$ عملگر $n+1$ام تفاضلات تقسیم‌شده‌ی محدود است و در آن مقدار $f(x)$ نامشخص است و به این دلیل نمی‌توان از این طریق خطای درونیابی را محاسبه کرد. اما اگر

یک نقطه‌ی بیشتر از داده‌ها در دسترس باشد بطور مثال $f(x_{n+1})$، می‌توان حد بالای خطای درونیابی از درجه‌ی n را با رابطه‌ی زیر تخمین زد.

$$\begin{aligned} R_n &\approx f[x_{n+1}, x_n, x_{n-1}, \ldots, x_0](x - x_0)(x - x_1)\ldots(x - x_n) \\ &= P_{n+1}(x) - P_n(x) \end{aligned} \tag{۱۱.۴}$$

یکی از اهداف درونیابی که در ابتدای فصل بیان شد، بیان داده‌ی موجود به زبان توابع تحلیلی است تا از طریق آنها قادر به تفسیر و استنباط ریاضی داده‌ها باشیم. به این منظور اگر فاصله‌ی نقاط در داده‌ها یکسان باشد یعنی

$$x_1 - x_0 = x_2 - x_1 = \cdots = h$$

روابطه تفاضلات تقسیم‌شده را می‌توان به شکل ساده‌تری نوشت. در این راستا ابتدا عملگرهای **تفاضلات پیشرو (Forward Differences)** را با روابط زیر تعریف می‌کنیم.

$$\begin{array}{lll} \Delta^0 f_0 &= f(x_0) & = f_0 \\ \Delta^1 f_0 &= \Delta^0 f_1 - \Delta^0 f_0 & = f_1 - f_0 \\ \Delta^2 f_0 &= \Delta^1 f_1 - \Delta^1 f_0 & = f_2 - 2f_1 - f_0 \\ & \vdots & \\ \Delta^n f_0 &= \Delta^{n-1} f_1 - \Delta^{n-1} f_0 & = f_n - \binom{n}{1} f_{n-1} + \binom{n}{2} f_{n-2} + \cdots + (-1)^n f_0 \end{array} \tag{۱۲.۴}$$

و همچنین عملگرهای **تفاضلات پسرو (Backward Differences)** را با روابط زیر تعریف می‌کنیم.

$$\begin{aligned} \nabla^0 f_n &= f(x_n) = f_n \\ \nabla^1 f_n &= \nabla^0 f_n - \nabla^0 f_{n-1} \\ &\vdots \\ \nabla^n f_n &= \nabla^{n-1} f_n - \nabla^{n-1} f_{n-1} \end{aligned} \tag{۱۳.۴}$$

با استفاده از این دو عملگر جدید، تفاضلات تقسیم‌شده را می‌توانیم به شکل زیر که بسیار شبیه به تعریف مشتق مرتبه‌ی nام است ساده نماییم.

$$\begin{aligned} f[x_n, x_{n-1}, \ldots, x_0] &= \frac{\Delta^n f_0}{n! h^n} \\ f[x_n, x_{n-1}, \ldots, x_{n-k}] &= \frac{\nabla^k f_n}{k! h^k} \end{aligned} \tag{۱۴.۴}$$

و در نهایت چون فواصل نقاط داده‌ها با هم یکسان می‌باشند، می‌توانیم با تغییر متغیر

$$s = \frac{x - x_0}{h} \rightarrow (x - x_1) = h(s - 1)$$

چندجمله‌ای درجه‌ی n را به شکل زیر برحسب تفاضلات پیشرو ساده نماییم.

$$P_n(s) = \Delta^0 f_0 + s\Delta^1 f_0 + s(s-1)\frac{\Delta^2 f_0}{2!} + \cdots + s(s-1)\ldots(s-(n-1))\frac{\Delta^n f_0}{n!} \quad (۱۵.۴)$$

و همچنین با تغییر متغیر

$$s = \frac{x - x_4}{h} \rightarrow (x - x_3) = h(s + 1)$$

چندجمله‌ای مذکور را برحسب تفاضلات پسرو به شکل زیر ساده نماییم.

$$P_n(s) = \nabla^0 f_4 + s\nabla^1 f_n + s(s+1)\frac{\nabla^2 f_n}{2!} + \cdots + s(s+1)\ldots(s+(n-1))\frac{\nabla^n f_n}{n!} \quad (۱۶.۴)$$

مثال شماره: ۵

درونیابی تابع $y = x\cos x$ با استفاده از چندجمله‌ای درجه‌ی ۴ در بازه‌ی $x \in [0, 2\pi]$:

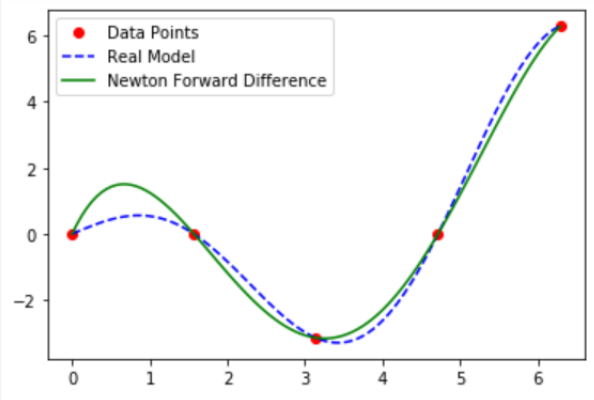
پاسخ:

```
# %% NEWTON FORWARD DIFFERENCE METHOD
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return x*np.cos(x)

def fac(m):
    f = 1
    for i in range(1,m+1):
        f *= i
```

```
    return f

x0,x1 = 0,2*np.pi
n = 5
h = (x1-x0)/(n-1)
x = np.linspace(x0,x1,n)

F = []

f0 = [fi for fi in f(x)]
F.append(f0)

for _ in range(n-1):
    f0 = F[-1]
    f1 = [f0[i+1]-f0[i] for i in range(len(f0)-1)]
    F.append(f1)

for i in range(len(F)):
    for j in range(len(F[i])):
        print(f" {F[i][j]:.2f} ",end='')
    print("")

y = ""
for i in range(len(F)):
    tmp = f"+({F[i][0]:.2f}/fac({i}))"
    y += tmp
    for j in range(i):
        tmp = f"*(s - {j})"
        y += tmp

print(f"P{n-1} = {y}")

xx = np.linspace(x0,x1,100)

s = (xx - x0)/h
P = eval(y)
```

```
plt.plot(x,f(x),"or",label="Data Points")
plt.plot(xx,f(xx),"--b",label="Real Model")
plt.plot(xx,P,"-g",label="Newton Forward Difference")
plt.legend()
plt.show()
```

```
 0.00  0.00  -3.14  -0.00  6.28
 0.00  -3.14  3.14  6.28
 -3.14  6.28  3.14
 9.42  -3.14
 -12.57
P4 = +(0.00/fac(0))+(0.00/fac(1))*(s - 0)+(-3.14/fac(2))*(s - 0)*(s - 1)
+(9.42/fac(3))*(s - 0)*(s - 1)*(s - 2)+(-12.57/fac(4))*(s - 0)*(s - 1)*
(s - 2)*(s - 3)
```



و اگر مجدد بجای s برحسب x چندجمله‌ای را بنویسیم به شکل زیر خواهد شد:

$$\begin{aligned} p_4(x) &= (0.000) + (0.000)(x-0) + (-0.637)(x-0)(x-1.57) \\ &\quad +(0.405)(x-0)(x-1.57)(x-3.14) \\ &\quad +(-0.086)(x-0)(x-1.57)(x-3.14)(x-4.71) \end{aligned}$$

مثال شماره: ۶

با استفاده از روش تفاضلات پیشرونده و پسرونده نیوتن، یک مدل چندجمله‌ای درجه ۳ که داده‌های جدول زیر را توصیف کند بدست آورید.

X	Y
0.50	0.00
1.00	0.11
1.50	1.42
2.00	7.11

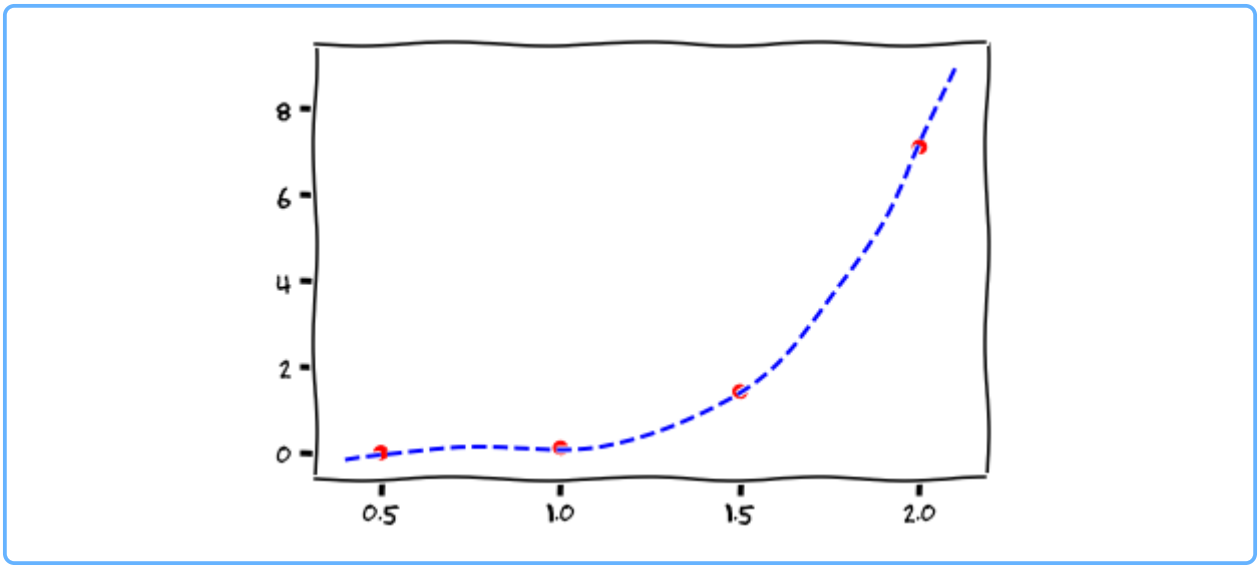
پاسخ:

X	Y				
0.50	0.00	$\Delta^0 f(x_0)$ = 0.00			
			$\Delta^1 f(x_0)$ = 0.11		
1.00	0.11	0.11		$\Delta^2 f(x_0)$ = 1.2	
			1.31		$\Delta^3 f(x_0)$ = 3.18
1.50	1.42	1.42		4.38	
			5.69		
2.00	7.11	7.11			

$$P_3(s) = (\Delta^0 f_0) + (\Delta^1 f_0)s + (\Delta^2 f_0)\frac{1}{2}s(s-1) + (\Delta^3 f_0)\frac{1}{6}s(s-1)(s-2)$$

$$= 0.11s + 0.6s(s-1) + 0.53s(s-1)(s-2)$$

که در آن $s = (x - x_0)/(x_1 - x_0) = 2(x - 0.5)$ است.



۲.۲.۴ روش لاگرانژ

همانطور که در شکل (۴.۴ـ الف) نشان داده می‌شود از ۲ نقطه فقط یک خط با به عبارتی چندجمله‌ای درجه‌ی ۱ عبور می‌کند و نمی‌توانیم از این ۲ نقطه خط دیگری عبور دهیم. به عبارت دیگر معادله‌ی هر خطی که این ۲ نقطه در آنها صدق کند، بعد از ساده کردن به معادله اول تبدیل خواهد شد. به همین منوال از n نقطه تنها یک چند جمله‌ای درجه‌ی $n-1$ عبور می‌کند و اگر با روش‌ها و الگوریتم‌های مختلف چندجمله‌ای‌های مختلفی که همه از درجه‌ی $n-1$ می‌باشند، بدست آوریم که تمامی این n نقطه عبور می‌کنند، در نهایت با ساده کردن همه به یک چندجمله‌ای درجه‌ی $n-1$ یکسان تبدیل می‌شوند و تمام ضرایب متناظر چندجمله‌ای‌ها یکسان خواهد شد. پس تفاوتی نمی‌کند که از چه روشی چندجمله‌ای‌های درونیاب را بدست آوریم. لیکن در این بخش، به دلیل جنبه‌های آموزشی و کاربرد آن در بخش‌های دیگر همچون انتگرالگیری عددی، با روش لاگرانژ برای بدست آوردن چندجمله‌ای‌های درونیاب آشنا می‌شویم. روابط (۱۷.۴) به اختصار روش لاگرانژ برای تشکیل چندجمله‌ای درونیاب درجه‌ی n که از $n+1$ نقطه‌ی $\{(x_0, f(x_0)), \ldots, (x_n, f(x_n))\}$ عبور می‌کند را نشان می‌دهند. همانطور که دیده می‌شود، $n+1$ تابع لاگرانژ L_i که تمامی چندجمله‌ای از درجه n می‌باشند را با ضرایب $f(x_i)$ با هم جمع می‌کنیم تا چندجمله‌ای $P_n(x)$ را برای تقریب تابع $f(x)$ بدست آوریم. توابع لاگرانژ که بطور مثال در روابط (۱۷.۴) تابع 1ام آن L_1 نشان داده شده است، از حاصلضرب n کسر تشکیل می‌شوند.

$$f(x) \cong P_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x) = f(x_0)L_0(x) + \cdots + f(x_n)L_n(x)$$

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

$$\text{e.g.} \quad L_1(x) = \frac{(x - x_0)(x - x_2)\ldots(x - x_n)}{(x_1 - x_0)(x_1 - x_2)\ldots(x_1 - x_n)} \tag{۱۷.۴}$$

تعداد توابع لاگرانژ $n+1$ و همه چندجمله‌ای از درجه‌ی n می‌باشند و هر یک تنها از یک نقطه‌ی $n+1$ داده عبور می‌کند. بطور مثال در شکل (۶.۴) برای تقریب درجه‌ی ۲ تابع $f(x)$ که به ۳ نقطه نیاز است، ۳ تابع لاگرانژ که هر یک از یکی از نقاط عبور می‌کنند و نیز هر یک از درجه‌ی ۲ می‌باشند (سهمی) را نشان می‌دهد. باید دقت کرد که هر تابع در ۲ نقطه نیز محور x را قطع می‌کند. تابع L_0 در نقطه‌ی $x = x_0$ تابع $f(x)$ و در نقاط $\{x_1, x_2\}$ محور x را قطع می‌کند. به همین منوال تابع L_1 در نقطه‌ی x_1 تابع $f(x)$ و در نقاط $\{x_0, x_2\}$ محور x را قطع می‌کند. پس می‌توانیم در حالت کلی ببینیم که:

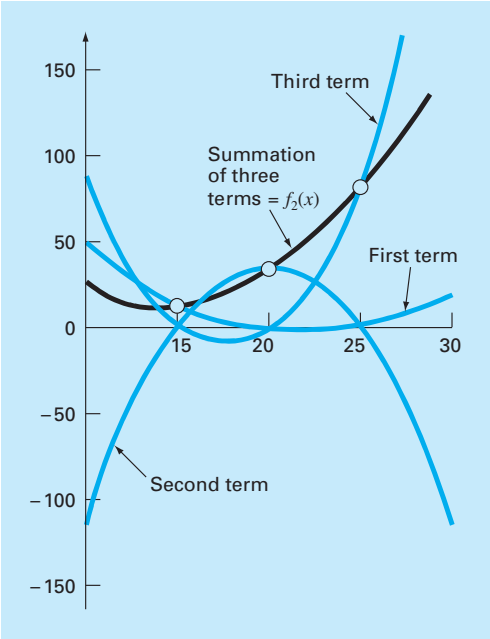
$$L_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{۱۸.۴}$$

مثال شماره: ۷

درونیابی تابع $y = x\cos x$ با استفاده از چندجمله‌ای درجه‌ی ۴ در بازه‌ی $x \in [0, 2\pi]$:

پاسخ:

```
# %% LAGRANGE METHOD FOR INTERPOLATION
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return x*np.cos(x)

x0,x1 = 0,2*np.pi
n = 5
```

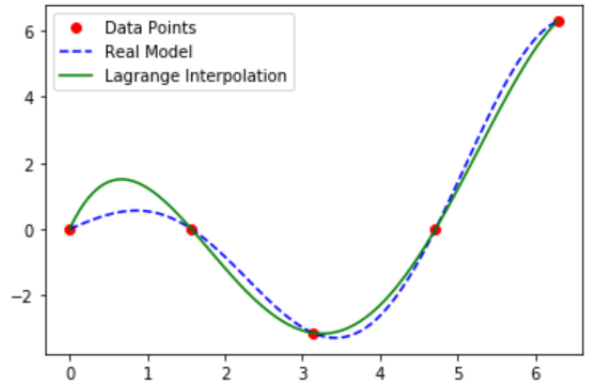


شکل ۶.۴: تابع لاگرانژ اول، دوم و سوم برای یک تقریب درجه‌ی ۲ تابع $f(x)$.

```
h = (x1-x0)/(n-1)
x = np.linspace(x0,x1,n)

L = ""
for j in range(n):
    tmp = f"f(x[{j}])"
    for i in range(n):
        if i != j:
            tmp += f"*(xx - x[{i}])/(x[{j}] - x[{i}])"
    L += "+" + tmp

xx = np.linspace(x0,x1,100)

print(L)
P = eval(L)

plt.plot(x,f(x),"or",label="Data Points")
plt.plot(xx,f(xx),"--b",label="Real Model")
plt.plot(xx,P,"-g",label="Lagrange Interpolation")
```

```
plt.legend()
plt.show()
```

```
P4 = +f(x[0])*(xx - x[1])/(x[0] - x[1])*(xx - x[2])/(x[0] - x[2])*(xx -
x[3])/(x[0] - x[3])*(xx - x[4])/(x[0] - x[4])+f(x[1])*(xx - x[0])/(x[1]
- x[0])*(xx - x[2])/(x[1] - x[2])*(xx - x[3])/(x[1] - x[3])*(xx - x
[4])/(x[1] - x[4])+f(x[2])*(xx - x[0])/(x[2] - x[0])*(xx - x[1])/(x[2]
- x[1])*(xx - x[3])/(x[2] - x[3])*(xx - x[4])/(x[2] - x[4])+f(x[3])*(xx
- x[0])/(x[3] - x[0])*(xx - x[1])/(x[3] - x[1])*(xx - x[2])/(x[3] - x
[2])*(xx - x[4])/(x[3] - x[4])+f(x[4])*(xx - x[0])/(x[4] - x[0])*(xx -
x[1])/(x[4] - x[1])*(xx - x[2])/(x[4] - x[2])*(xx - x[3])/(x[4] - x[3])
```



همانطور که دیده می‌شود منحنی درجه ۴ بدست آمده از روش لاگرانژ دقیقا منحنی درجه ۴ است که از روش تفاضلات پیشرونده نیوتن در مثال قبل (۵) بدست آوردیم.

تمرین شماره: ۱

فرض کنید میزان غلظت داروی خاصی در جریان خون یک بیمار طی 10 ساعت اندازه‌گیری می‌شود و در جدول زیر آورده شده است. برای درمان بیمار میزان غلظت دارو در جریان خون بیمارنباید از $4(ng/ml)$ کمتر و از $17(ng/ml)$ بیشتر باشد. اگر مدل توصیف کننده‌ی سطح غلظت دارو در طول زمان با رابطه‌ی $y = a\ t\ e^{b\,t}$ داده شود، مشخص کنید بیمار هر چند ساعت یک بار باید از دارو استفاده کند تا غلظت دارو در جریان خون بیمار در این سطح مطلوب باقی می‌ماند. در صورت نیاز به حل معادله جبری از روش نیوتن–رافسن استفاده نمایید.

hour	concentration (ng/ml)
1	6.2
2	9.5
3	12.3
4	13.9
5	14.6
6	13.5
7	13.3
8	12.7
9	12.4
10	11.9

تمرین شماره: ۲

یک مرکز تحقیقاتی اقلیمی سطح گستره‌ی یخ در قطب شمال را در سال‌های ۲۰۱۵ و ۲۰۱۶ هر ماه اندازه‌گیری و در جدول زیر بر حسب میلیون کیلومتر مربع ($10^6 km^2$) گزارش کرده است.

month	2015	2016
Jan.	13.75	13.64
Feb.	14.51	14.32
Mar.	14.49	14.53
Apr.	13.98	13.83
May	12.69	12.08
Jun.	11.05	10.60
Jul.	8.83	8.13
Aug.	5.66	5.60
Sept.	4.68	4.72
Oct.	7.79	6.45
Nov.	10.11	9.08
Dec.	12.33	12.09

اگر مقدار گستره‌ی سطح یخ در قطب شمال از مدل

$$y = c_1 + c_2 t + c_3 \sin 2\pi t + c_4 \cos 2\pi t$$

پیروی کند که در آن زمان t بر حسب سال از ابتدای ژانویه ۲۰۱۵ و سطح یخ y بر حسب $(10^6 km^2)$ سنجیده می‌شود، پارامترهای مدل را بدست آورید. نرخ کاهش متوسط سالیانه‌ی سطح یخ‌های قطب شمال را بدست آورید. برای محاسبه‌ی نرخ متوسط، کافیست یک خط از داده‌ها برازش دهید. شیب این خط معرف نرخ متوسط تغییرات است.

تمرین شماره: ۳

الف - با استفاده از جدول داده‌های زیر مقدار $f(2.8)$ را از روش چندجمله‌ای درونیاب نیوتن مرتبه ۱ تا ۳ بدست آورید. نقاط مورد استفاده‌ی درونیابی را به نحوی انتخاب کنید که محاسبه‌ی شما بیشترین دقت ممکن را داشته باشد.

x	1.6	2	2.5	3.2	4	4.5
f(x)	2	8	14	15	8	2

ب - مساله‌ی قسمت قبل را با روش لاگرانژ مرتبه‌ی ۱ تا ۳ حل نمایید و مجددا دقت نمایید که نقاطی را برای درونیابی انتخاب کنید تا به بیشترین دقت ممکن برسید.

تمرین شماره: ۴

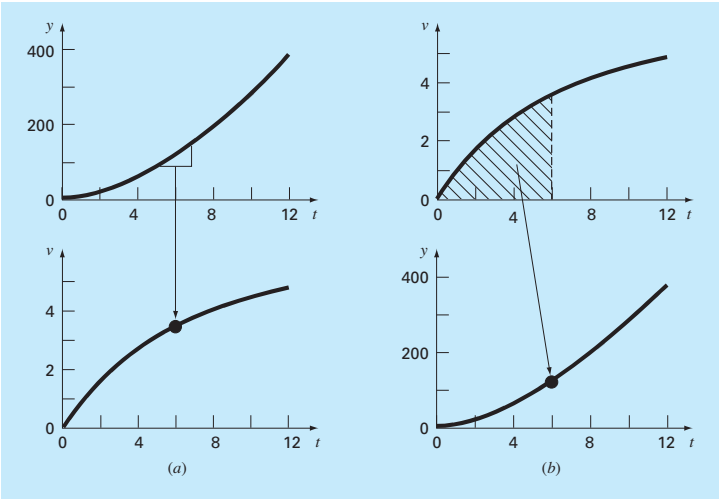
جدول داده‌های زیر از یک آزمایش بسیار دقیق بدست آمده است. بهترین روش را انتخاب کنید تا y را در نقطه‌ی $x = 3.5$ محاسبه نمایید. نشان دهید که یک چندجمله‌ای جواب دقیق این مساله است و درجه‌ی این چند جمله‌ای را محاسبه کنید.

x	0	1.8	5	6	8.2	9.2	12
y	26	16.415	5.375	3.5	2.015	2.54	8

فصل ۵

مشتقگیری و انتگرال‌گیری

دو تا از پرکاربردترین ابزار ریاضی در محاسبات علمی و مهندسی مشتقگیری و انتگرال‌گیری است. مشتق‌گیری و انتگرالگیری دو عملگر معکوس هم می‌باشند. یعنی وقت با عمل مشتقگیری از یک تابع به یک تابع جدید می‌رسیم با عمل انتگرالگیری از تابع جدید می‌توانیم به تابع اول برگردیم. در عمل مشتق‌گیری نرخ تغییرات موضعی یک کمیت را محاسبه می‌کنیم و در عمل انتگرالگیری سطح زیر منحنی یک کمیت را محاسبه می‌کنیم. شکل (۱.۵) ارتباط این دو عملگر با هم را نشان می‌دهد.



شکل ۱.۵: ارتباط عمل مشتق‌گیری و انتگرال‌گیری یک تابع.

مانند تمام روش‌های عددی که تا اینجا مطالعه کردیم، بدلیل محدودیت‌های روش‌های تحلیلی ناگزیر به استفاده از روش‌های عددی هستیم. در عمل توابع بسیار زیادی هستند که عمل انتگرال‌گیری از آن‌ها به روش تحلیلی غیرممکن است. البته در مورد مشتق این قضیه قدری متفاوت است و بیشتر زمانی مجبور به استفاده از روش‌های مشتق‌گیری عددی می‌شویم که شکل تحلیلی تابع را در اختیار نداشته باشیم که توضیح خواهیم داد ابتدا با یک برازش منحنی را ایجاد و سپس از آن مشتق می‌گیریم.

مواردی هم وجود دارد که حتی مقادیر عددی تابع هم در دسترس نیست و فقط معادله دیفرانسیل حاکم بر مساله را می‌شناسیم و به کمک مشتقات عددی معادلات دیفرانسیل را به معادلات جبری تبدیل و سپس این معادلات جبری را با روش‌هایی که جلوتر آموختیم می‌توانیم حل نماییم. در این فصل ابتدا با انتگرالگیری عددی که دارای پاسخ‌های دقیق‌تری هستند، آشنا می‌شویم و سپس مشتق‌گیری عددی را که بیشتر در حل معادلات دیفرانسیل کاربرد دارد بررسی می‌کنیم.

۱.۵ انتگرال‌گیری عددی

در حالت کلی برای انتگرال‌گیری عددی، ما دو روش نیوتن_کاتس و گاوس را بررسی می‌کنیم.

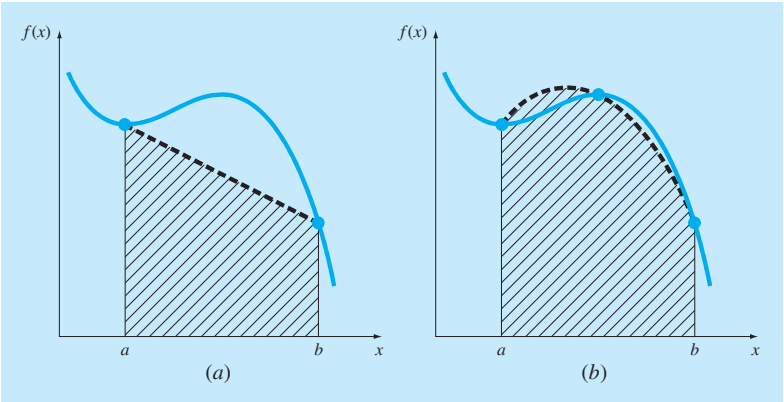
۱.۱.۵ روش نیوتن_کاتس (Newton-Cotes)

در روش نیوتن_کاتس، تابع زیر انتگرال را از طریق روش‌های درون‌یابی که مطالعه کردیم، با یک چندجمله‌ای درجه‌ی n تقریب می‌زنیم.

$$\begin{aligned} \int_a^b f(x)dx &\cong \int_a^b P_n(x)dx \\ P_n(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n \end{aligned} \qquad (۱.۵)$$

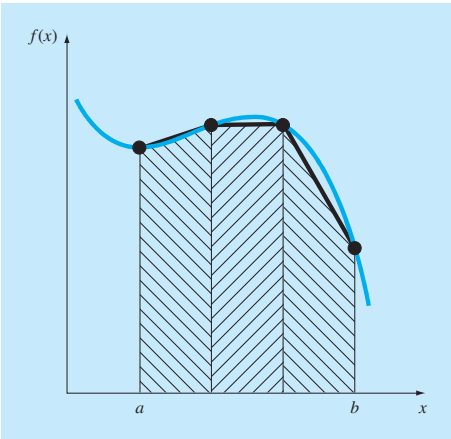
بطور مثال در شکل (۲.۵) تقریب یک تابع با یک چندجمله‌ای درجه یک (خط) و درجه‌ی دو (سهمی) را نشان می‌دهد. همانطور که در شکل مشاهده می‌شود، سطح زیر منحنی چندجمله‌ای با سطح زیر منحنی تابع اصلی اختلاف دارد و این اختلاف خطای محاسبه‌ی انتگرال‌گیری عددی است. با افزایش درجه‌ی چندجله‌ای خطای محاسبه‌ی انتگرال کاهش می‌یابد ولی این مستلزم هزینه‌ی محاسباتی بالاتر است و درضمن بستگی به تعداد داده‌های موجود در بازه‌ی انتگرال‌گیری دارد. در قسمت درون‌یابی توابع آموختیم که چندجمله‌ای درجه‌ی nای که از $n+1$ نقطه عبور می‌کند، یکتا است و بستگی به روش بدست آوردن این چندجمله ندارد. لذا درشکل (۲.۵) اگر تابع اصلی (آبی رنگ) یک چندجمله‌ای درجه‌ی یک یعنی یک خط بود، تقریب درجه یک (قسمت a شکل) انتگرال عددی پاسخ دقیق داشت و خطای آن صفر می‌شد و اگر تابع اصلی درجه‌ی دو بود، تقریب درجه‌ی دو (قسمت b شکل) پاسخ دقیق داشت.

درحالت کلی در یک روش انتگرال‌گیری عددی، کمیتی به نام **درجه دقت (Degree of Precision)** تعریف می‌کنیم که معرف میزان عملکرد روش مذکور است. درجه‌ی دقت یک روش انتگرال‌گیری عددی، بزرگترین عدد صحیح k است که تمام چندجمله‌ای‌ها با درجه‌ی k و کمتر، با آن روش عددی پاسخ کاملا دقیق دهند. بطور مثال روش نیوتن_کاتس با استفاده از دو نقطه (ذوزنقه‌ای) که درقسمت a شکل (۲.۵) نشان داده شده است، قطعا قادر به انتگرال‌گیری دقیق یک سهمی که درجه



شکل ۲.۵: انتگرال عددی از طریق تقریب منحنی با چندجمله‌ای درجه یک (خط) و دو (سهمی).

$n = 2$ دارد، نیست. پس روش ذوزنقه‌ای درجه‌ی دقت $k = 1$ دارد. در بازه‌های کوچک‌تر تقریب یک منحنی با درجه‌ی بالا، بوسیله‌ی خط بسیار دقیق‌تر از تقریب آن منحنی در بازه‌های بزرگتر است. پس یک راه برای بالابردن دقت روش نیوتن_کاتس تقسیم بازه‌ی انتگرال‌گیری به بازه‌های کوچکتر است تا بجای بالا بردن درجه‌ی چندجمله‌ای و افزایش پیچیدگی محاسبات، با ترکیب چندجمله‌ای‌های درجه‌ی پایین‌تر اما در بازه‌های کوچک‌تر استفاده کنیم که به آن روش **مرکب (Complex)** می‌گوییم. در شکل (۳.۵) انتگرال‌گیری با روش ذوزنقه‌ای مرکب را نشان می‌دهد. در مقایسه با قسمت a شکل (۲.۵) ، همانطور که دیده می‌شود، بازه به ۳ قسمت تقسیم شده است و سطح هاشورخورده با خطای بسیار کمتری، سطح زیر منحنی اصلی (آبی رنگ) را تقریب می‌زند.



شکل ۳.۵: روش مرکب برای کاهش خطای روش‌های با درجه‌ی دقت کمتر.

۲.۱.۵ قاعده‌ی ذوزنقه‌ای (Trapezoidal Rule)

ساده‌ترین شکل روش نیوتن_کاتس، استفاده از یک خط (چندجمله‌ای درجه یک) برای تقریب منحنی اصلی است. با این تقریب خطی، سطح زیر منحنی با سطح یک ذوزنقه تقریب زده می‌شود. روابط

(۲.۵) جزییات این تقریب و نیز انتگرال‌گیری از آن را نشان می‌دهد. در تقریب منحنی از روش نیوتن پیشرو برای داده‌های با فاصله‌ی یکسان استفاده کرده‌ایم. همانطور که دیده می‌شود خطای تقریب از مرتبه‌ی $\mathcal{O}(h^2)$ است و بعد از انتگرال‌گیری دقت افزایش می‌یابد و از مرتبه‌ی $\mathcal{O}(h^3)$ می‌شود.

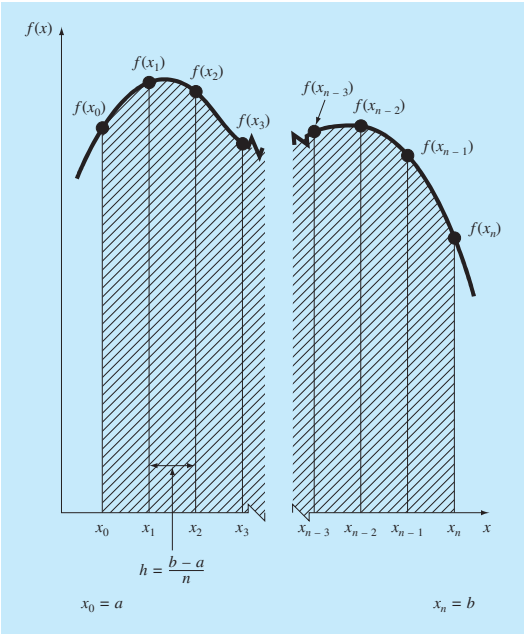
$$
\begin{aligned}
I &= \int_{x_0}^{x_1} f(x)dx \cong \int_{x_0}^{x_1} P_1(x)dx \\
I &= \int_{x_0}^{x_1} \left(a_0 + a_1(x-a) + \frac{f''(\xi)}{2}(x-a)(x-2a) \right) dx \\
&= \int_0^1 \left(\Delta^0 f_0 + s\Delta^1 f_0 + s(s-1)h^2 \frac{f''(\xi)}{2} \right) h ds \quad \text{where } s = \frac{x-x_0}{h} \\
&= h\left[s\Delta^0 f_0 + \frac{s^2}{2}\Delta^1 f_0 + h^2 f''(\xi)\left(\frac{s^3}{6} - \frac{s^2}{4} \right) \right]_0^1 \\
&= h\left[\Delta^0 f_0 + \frac{1}{2}\Delta^1 f_0 - \frac{1}{12}h^2 f''(\xi) \right] \qquad (۲.۵)
\end{aligned}
$$

بعد از ساده کردن روابط، شکل کلی قاعده‌ی ذوزنقه‌ای و خطای برشی آن به شکل معادله‌ی (۳.۵) خلاصه می‌شود. در این رابطه x_0 و x_1 نقاط ابتدا و انتهای بازه است. یادآوری می‌شود که بازه یک تکه می‌باشد و به زیر بازه تقسیم نشده است. همچنین ξ نقطه‌ای در این بازه یعنی بین x_0 و x_1 می‌باشد. خطای برشی بدست آمده معرف کران بالای خطا می‌باشد. لذا برای ξ محلی را انتخاب می‌کنیم که مشتق مرتبه‌ی دو تابع اصلی یعنی $f''(x)$ در آن نقطه بیشینه باشد. کمیت h نیز معرف پهنای بازه یعنی $h = x_1 - x_0$ می‌باشد.

$$
I = \underbrace{h\frac{f(x_0)+f(x_1)}{2}}_{\text{قاعده ذوزنقه‌ای}} \underbrace{-\frac{1}{12}h^3 f''(\xi)}_{\text{خطای برشی}} \qquad (۳.۵)
$$

همانطور که گفته شد، برای کاهش خطای روش ذوزنقه‌ای می‌توانیم بازه را به قسمت‌های باریک‌تر که پهنای کمتری دارند تقسیم کنیم. و روش را برای هر قسمت مستقلا اعمال نماییم. با این کار پهنای هر قسمت بسته به تعداد تقسیمات کاهش پیدا می‌کند، یعنی $h = (x_1 - x_0)/n$. و از آنجاییکه خطای برشی روش ذوزنقه‌ای از مرتبه‌ی $\mathcal{O}(h^3)$ است، با توان سوم تعداد تقسیمات بازه یعنی n کاهش می‌یابد. همانطور که در شکل (۴.۵) مشاهده می‌شود بعد از تقسیمات نقطه‌ی ابتدای بازه x_0 و نقطه‌ی انتهایی x_n خواهد شد. فاصله‌ی نقاط با هم مساوی می‌باشند.

$$
h = \frac{(x_n - x_0)}{n} = x_1 - x_0 = x_2 - x_1 = \cdots = x_n - x_{n-1}
$$



شکل ۴.۵: روش ذوزنقه‌ای مرکب و تقسیم بازه به n قسمت مساوی.

بعد از تقسیم، انتگرال کل می‌شود مجموع n انتگرال با پهنای h جدید و هر انتگرال را با روش ذوزنقه‌ای محاسبه می‌کنیم. روابط (۴.۵) جزییات این محاسبات را نشان می‌دهند. برای محاسبه‌ی خطای برشی، نقطه‌ی ξ_i در بازه‌ی iام یعنی $x_{i-1} \le \xi_i \le x_i$ انتخاب می‌شود.

$$\begin{aligned} I &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx \\ &= \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] - \frac{h^3}{12}\sum_{i=1}^{n} f''(\xi_i) \end{aligned} \qquad (۴.۵)$$

با ساده کردن روابط و بیان آن‌ها برحسب نقطه‌ی ابتدایی x_0، انتهایی x_n و تعداد تقسیمات n به رابطه‌ی (؟؟) می‌رسیم. در این رابطه مشاهده می‌کنیم که انتگرال را می‌توان بشکل حاصلضرب پهنا در میانگین تابع در نقاط تقسیم کننده‌ی بازه نوشت. در محاسبه‌ی کران بالای خطای برشی، بیشینه‌ی مشتق دوم تابع در n زیر قسمت بوجود آمده بدست می‌آوریم و سپس همه را با هم جمع می‌کنیم. از آنجاییکه مشتق دوم می‌تواند در هر قسمت با قسمت‌های دیگر بسیار متفاوت باشد، کار صحیحی نیست که بیشینه‌ی مشتق دوم بین قسمت‌های مختلف را جایگذین همه‌ی قسمت‌ها کنیم. و استفاده از میانگین آن‌ها به مقدار درست خطای برشی نزدیک‌تر است. که در نهایت به رابطه‌ی (۶.۵) منجر می‌شود. بر اساس این رابطه برای محاسبه‌ی کران بالای خطای برشی، ابتدا میانگین مشتق دوم تابع اصلی را در تمام بازه بدست می‌آوریم. البته دقت می‌کنیم در صورت استفاده از میانگین، خطای برشی، بجای توان ۳، با مربع تعداد تقسیمات کاهش می‌یابد.

$$I = \underbrace{\overbrace{(x_n - x_0)}^{\text{پهنا}} \overbrace{\frac{f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)}{2n}}^{\text{میانگین تابع}}}_{\text{قاعده مرکب ذوزنقه‌ای}} \underbrace{- \frac{(x_n - x_0)^3}{12n^3} \sum_{i=1}^{n} f''(\xi_i)}_{\text{خطای برشی}} \quad (۵.۵)$$

$$\bar{f''} = \frac{\sum_{i=1}^{n} f''(\xi_i)}{n} \rightarrow E_a = -\frac{(x_n - x_0)^3}{12n^2} \bar{f''} \quad (۶.۵)$$

مثال شماره: ۱

انتگرال زیر را با روش ذوزنقه‌ای مرکب محاسبه نمایید. محاسبات را با تعداد تقسیمات بازه از ۲ قسمت تا ۳۰ قسمت تکرار نمایید و خطای نسبی برشی را با خطای نسبی واقعی در هر تکرار مقایسه نمایید. مقدار واقعی انتگرال $I_{exact} = 1.640533$ می‌باشد.

$$I = \int_0^{0.8} \left(0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5\right) dx$$

از آنجاییکه خطای روش ذوزنقه‌ای مرکب از رابطه‌ی (۶.۵) بدست می‌آید و این کران بالای خطا می‌باشد،

$$f''(x) = 8000x^3 - 10800x^2 + 4050x - 400 \rightarrow \bar{f''} = \frac{1}{0.8} \int_0^{0.8} f''(x)dx = -60$$

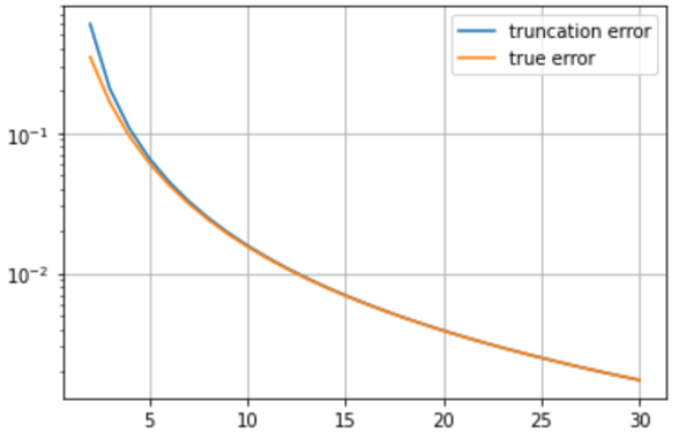
$$E_{truncation} = -\frac{(0.8)^3}{12 \times n^2}(-60)$$

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f(x):
    return 0.2 + 25*x - 200*x**2 + 675*x**3 - 900*x**4 + 400*x**5

x0,xn = 0,0.8

II = [0]
Etrunc_ = [0]
Etrue_ = [0]
Iexact = 1.640533
N = 31

for n in range(1,N):
    x = np.linspace(x0,xn,n+1)

    I = f(x0) + f(xn)
    for i in range(1,n):
        I += 2 * f(x[i])

    I = (xn-x0)/(2*n)*I
    Etrunc = abs((0.8)**3*60/12/n**2/I)
    Etrue = abs((I - Iexact)/Iexact)
    II.append(I)
    Etrunc_.append(Etrunc)
    Etrue_.append(Etrue)
    if n > 1:
        print(f"{n}\t{I:0.2f}\t{Etrunc:0.3f}\t{Etrue:0.3f}")

nn = list(range(2,N))
plt.plot(nn,Etrunc_[2:],label="truncation error")
plt.plot(nn,Etrue_[2:],label="true error")
plt.legend()
plt.yscale('log')
plt.grid(True)
plt.show()
```

در این پلات خطای نسبی برشی با رنگ آبی و خطای نسبی واقعی با رنگ نارنجی نمایش داده شده است. محور y در مقیاس لگاریتمی نمایش داده شده است تا منحنی‌ها قابل مقایسه باشند. محور x نیز معرف تعداد تقسیمات بازه می‌باشد. در جدول نیز ستون‌ها از چپ به ترتیب تعداد تقسیمات، مقدار انتگرال محاسبه شده با این تعداد تقسیمات خطای نسبی برشی و ستون آخر خطای نسبی واقعی است. در جدول از تقسیمات بازه به ۲ قسمت تا ۵ و بعد از ۲۶تا نمایش داده شده است.

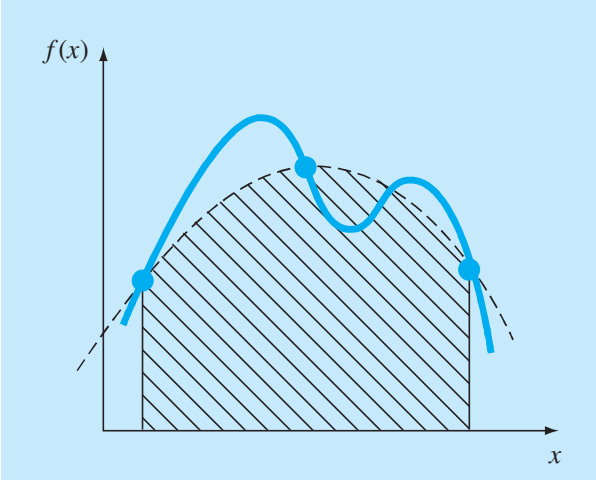
2	1.07	0.599	0.349
3	1.37	0.208	0.165
4	1.48	0.108	0.095
5	1.54	0.066	0.061
26	1.64	0.002	0.002
27	1.64	0.002	0.002
28	1.64	0.002	0.002
29	1.64	0.002	0.002
30	1.64	0.002	0.002



۳.۱.۵ قاعده‌ی سیمپسون (Simpson's Rule)

در تقریب بعدی، در روش نیوتن_کاتس بجای استفاده از ۲ نقطه، باید از ۳ نقطه استفاده کنیم و منحنی اصلی را با یک چندجمله‌ای درجه‌ی ۲ تقریب بزنیم. شکل (۵.۵) این تقریب را نشان می‌دهد. رابطه‌ی (۷.۵) جزییات محاسبه‌ی این تقریب و انتگرال‌گیری از آن را نشان می‌دهد. در این محاسبات بازه به دو قسمت مساوی تقسیم شده است و پهنای هر قسمت h می‌باشد. دقت کنید که در محاسبه‌ی تقریب، هر چند از ۳ نقطه استفاده می‌کنیم، اما از آنجاییکه انتگرال خطای برشی مرتبه‌ی $\mathcal{O}(h^3)$ صفر می‌شود، خطای یک مرتبه بالاتر یعنی $\mathcal{O}(h^4)$ نیز آورده شده است. در نهایت بعد از محاسبه‌ی انتگرال نهایی رابطه‌ی (۷.۵) و ساده کردن روابط، قاعده‌ی سیمپسون به شکل رابطه‌ی (۸.۵) خلاصه می‌شود.

$$
\begin{aligned}
I &= \int_{x_0}^{x_2} f(x)dx \cong \int_{x_0}^{x_2} P_2(x)dx \\
I &= \int_{x_0}^{x_2} \left(a_0 + a_1(x-x_0) + a_2(x-x_0)(x-2x_0) + \frac{1}{3!}f^{(3)}(\xi')(x-x_0)(x-2x_0)(x-3x_0) \right. \\
&\qquad \left. + \frac{1}{4!}f^{(4)}(\xi)(x-x_0)(x-2x_0)(x-3x_0)(x-4x_0) \right) dx \\
I &= \int_0^2 \left(\Delta^0 f_0 + s\Delta^1 f_0 + \frac{1}{2}s(s-1)\Delta^2 f_0 + s(s-1)(s-2)\frac{h^3}{3!}f^{(3)}(\xi') \right. \\
&\qquad \left. + s(s-1)(s-2)(s-3)\frac{h^4}{4!}f^{(4)}(\xi) \right) hds \quad \text{where} \quad s = \frac{x-x_0}{h}
\end{aligned}
\tag{۷.۵}
$$



شکل ۵.۵: انتگرال نیوتن‌ـ کاتس با استفاده از ۳ نقطه (روش سیمپسون)

$$
I = \underbrace{\overbrace{2h}^{\text{پهنا}} \overbrace{\left[\frac{f(x_0) + 4f(x_1) + f(x_2)}{6} \right]}^{\text{میانگین وزنی}}}_{\text{سیمپسون}} \underbrace{+0 - \frac{1}{90}f^{(4)}(\xi)h^5}_{\text{خطای برشی}} \tag{۸.۵}
$$

در رابطه‌ی (۸.۵) مجدد مشاهده می‌کنیم که انتگرال بصورت حاصلضرب پهنای بازه‌ی انتگرال‌گیری $(2h)$ در میانگین وزنی مقادیر تابع در نقاط استفاده شده در بازه است. برای محاسبه‌ی کران بالای خطای برشی مقدار بیشینه‌ی $f^{(4)}(\xi)$ در سرتاسر بازه محاسبه می‌شود. خطای برشی با توان ۵ پهنای بازه متناسب است. لذا در انتگرال‌هایی که بازه‌ی انتگرال‌گیری وسیع است برای کاهش خطای برشی

از روش سیمپسون مرکب استفاده می‌کنیم. مانند روش ذوزنقه‌ای مرکب، اینجا نیز بازه را به n زیر بازه تقسیم می‌کنیم. از آنجاییکه در هر انتگرال سیمپسون نیاز به ۳ نقطه داریم، پس n باید عدد زوج باشد، تا بتوانیم انتگرال کل را به تعداد $n/2$ انتگرال تجذیه کنیم.

$$\int_{x_0}^{x_n} f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx \qquad (۹.۵)$$

سپس با استفاده از رابطه‌ی (۸.۵) برای هر قسمت انتگرال را محاسبه می‌کنیم. از تجمیع انتگرال‌های زیر بازه‌ها، انتگرال کل را محاسبه می‌نماییم. دقت کنید که در رابطه‌ی (۸.۵) نقاط میانی در زیر بازه‌ها که اندیس‌های فرد دارند با ضریب ۴ باید در نظر گرفته شوند. و در هنگام جمع کردن انتگرال زیر بازه‌ها، نقاط مرزی در هر ۲ بازه‌ی همسایه وارد می‌شوند و لذا ضریب ۲ خواهند گرفت. در نهایت بعد از ساده کردن روابط، برای قاعده‌ی سیمپسون مرکب، به رابطه‌ی (۱۰.۵) می‌رسیم. مجدد در این رابطه مشاهده می‌کنیم که انتگرال عددی حاصلضرب پهنا در میانگین وزنی مقدار تابع در نقاط درون بازه است. همچنین مانند قاعده‌ی ذوزنقه‌ای مرکب، برای محاسبه‌ی خطای برشی، میانگین $f^{(4)}(x)$ در سرتاسر بازه‌ی انتگرال‌گیری را استفاده می‌کنیم.

$$I = \overbrace{(x_n - x_0)}^{\text{پهنا}} \overbrace{\frac{f(x_0) + 4\sum_{i=1,3,5,\ldots}^{n-1} f(x_i) + 2\sum_{i=2,4,6,\ldots}^{n-2} f(x_i) + f(x_n)}{3n}}^{\text{میانگین وزنی}} - \overbrace{\frac{(x_n - x_0)^5}{180n^4} f^{(4)}}^{\text{خطای برشی}}$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{\text{سیمپسون مرکب}}$$

(۱۰.۵)

۴.۱.۵ روش گاوس

در روش ذوزنقه‌ای از مفهوم درونیابی خطی استفاده کردیم. اما همین نتیجه را می‌توانیم با این رویکرد بدست آوریم که هر انتگرال را می‌توانیم بصورت ترکیب خطی با ضرایب نامعین از مقدار تابع در مرزهای بازه‌ی انتگرال‌گیری بنویسیم.

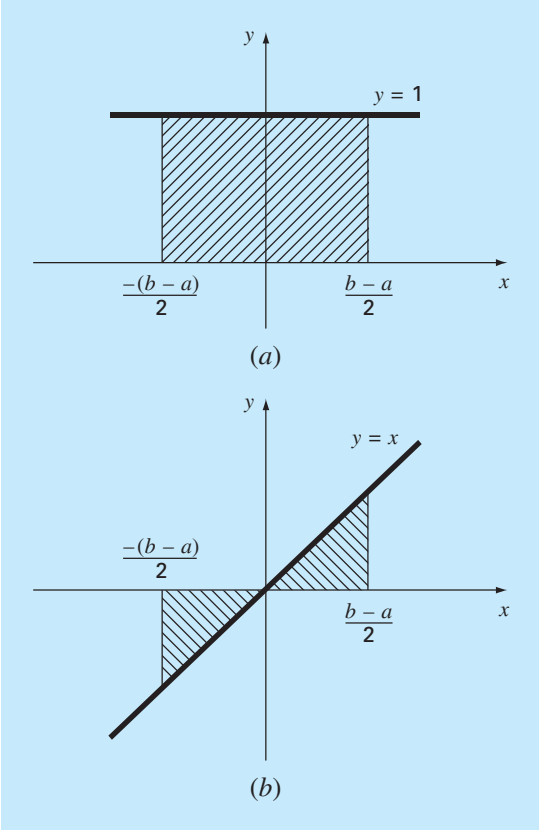
$$I \cong c_0 f(a) + c_1 f(b)$$

برای تعیین ۲ ضریب c_0 و c_1 باید ۲ معادله تشکیل دهیم. با فرض اینکه روش ضرایب نامعین دو نقطه‌ای حداقل دارای درجه دقت ۱ است، فرض می‌کنیم چندجمله‌ای‌های درجه‌ی صفر و یک با این روش مقدار دقیق خواهند داشت. سطوح زیر نمودار توابع درجه‌ی صفر و درجه‌ی یک که در شکل (۶.۵) نشان داده شده است، را با روش تحلیلی و ضرایب نامعین محاسبه می‌کنیم و با هم مساوی

قرار می‌دهیم. همانطور که در رابطه‌ی (۱۱.۵) مشاهده می‌شود با حل این دستگاه، ضرایب نامعین c_0 و c_1 محاسبه می‌شوند. و در نهایت رابطه به همان قاعده‌ی ذوزنقه‌ای که از روش نیوتن_کاتس بدست آورده بودیم، منجر می‌شود.

$$\begin{cases} c_0 + c_1 = \int_{-(b-a)/2}^{(b-a)/2} 1\,dx \\ c_0(-\frac{b-a}{2}) + c_1(\frac{b-a}{2}) = \int_{-(b-a)/2}^{(b-a)/2} x\,dx \end{cases} \Rightarrow c_0 = c_1 = \frac{b-a}{2}$$

$$\Rightarrow I \cong (b-a)\frac{f(a)+f(b)}{2} \quad (۱۱.۵)$$



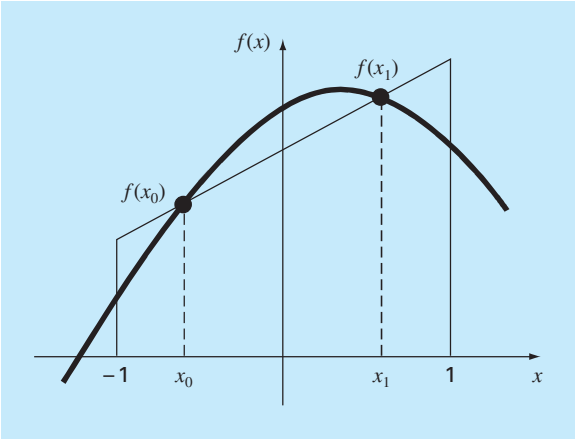
شکل ۶.۵: روش ضرایب نامعین با درجه‌ی دقت یک.

با الهام از ایده‌ی ضرایب نامعین دو نقطه‌ای که منجر به قاعده‌ی ذوزنقه‌ای با درجه‌ی دقت ۱ شد، علاوه بر ضرایب، مکان نقاط درون بازه را نیز نامعین در نظر می‌گیریم. و همانطور که در رابطه‌ی (۱۲.۵) مشاهده می‌شود، انتگرال را می‌توانیم بصورت جمع یک سری با ضرایب نامعین c_i و مقدار تابع در نقاط نامعین x_i بنویسیم. و با مساوی قرار دادن با مقدار محاسبه‌شده از انتگرال تحلیلی همان

چندجمله‌ای‌ها و تشکیل معادلات، ضرایب و نقاط نامعین را تعیین کنیم. به این روش محاسبه‌ی عددی انتگرال‌ها **قاعده‌ی گاوس ـ لژاندر** گفته می‌شود. با مقایسه‌ی بسط (۱۲.۵) و چندجمله‌ای‌های درونیاب لاگرانژ، می‌توانیم پیش‌بینی کنیم که ضرایب c_i انتگرال چندجمله‌ای‌های لاگرانژ هستند. از آنجاییکه با تغییر متغیر، بازه هر انتگرالی را می‌توانیم به -1 تا 1 انتقال دهیم، انتگرال‌ها را در حالت کلی در این بازه محاسبه می‌کنیم.

$$\begin{aligned} I &= \int_{-1}^{1} f(x)dx \cong \sum_{i=1}^{n} c_i f(x_i) \\ c_i &= \int_{-1}^{1} L_i(x)dx, \quad i = 1, \ldots, n \end{aligned} \tag{۱۲.۵}$$

به عنوان مثال، روش گاوس ـ لژاندر را برای حالت ۲ نقطه‌ای محاسبه می‌کنیم. شکل (۷.۵) بازه‌ی انتگرال‌گیری و نقاط درون بازه را نشان می‌دهد. در این حالت این روش بیشتر با نام **تربیع گاوس (Gauss Quadrature)** شناخته می‌شود. همانطور که در روابط (۱۳.۵) نشان داده شده است، تعداد ۴ مجهول c_0, c_1, x_0, x_1 وجود دارد که برای بدست آوردن آنها نیاز به ۴ معادله داریم. این ۴ معادله از انتگرال تحلیلی تابع x^α برای مقادیر $\alpha = 0, 1, 2, 3$ در بازه‌ی -1 تا 1 بدست می‌آیند. در نهایت با حل دستگاه معادلات، مقادیر مجهول محاسبه می‌شوند که در رابطه‌ی (۱۳.۵) آورده شده است. همانطور که مشاهده می‌شود ضرایب با هم مساوی $c_0 = c_1 = 1$ و نقاط درون بازه نیز $x_0, x_1 = \pm 1/\sqrt{3}$ برای فاصله‌ی -1 تا 1 بدست می‌آیند. شکل نهایی قاعده تربیع گاوس با رابطه‌ی ۱۴.۵ داده می‌شود.



شکل ۷.۵: درجه دقت ۲ که قادر است ۲ مجهول را حل نمایید و منجر به یک روش ذوزنقه‌ای با نقاط مجهول x_0 و x_1 می‌شود.

$$I = \int_{-1}^{1} f(x)dx \cong c_0 f(x_0) + c_1 f(x_1)$$

$$\begin{cases} c_0 + c_1 = \int_{-1}^{1} 1dx = 2 \\ c_0 x_0 + c_1 x_1 = \int_{-1}^{1} x dx = 0 \\ c_0 x_0^2 + c_1 x_1^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3} \\ c_0 x_0^3 + c_1 x_1^3 = \int_{-1}^{1} x^3 dx = 0 \end{cases} \Rightarrow \begin{cases} c_0 = c_1 = 1 \\ x_0 = -\frac{1}{\sqrt{3}} \\ x_1 = +\frac{1}{\sqrt{3}} \end{cases} \quad (۱۳.۵)$$

$$I \cong f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) \quad (۱۴.۵)$$

در جدول (۸.۵) قاعده‌ی گاوس ـ لژاندر برای حالت‌های ۲ نقطه‌ای تا ۶ نقطه‌ای آورده شده است. ستون دوم ضرایب نامعین و ستون سوم نقاط درون بازه‌ی $(-1, 1)$ می‌باشند. برای استفاده از جدول باید حدود انتگرال با استفاده از تغییر متغیر (۱۵.۵) به بازه‌ی $(-1, 1)$ انتقال یابد.

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{1}{2}a(1-t) + \frac{1}{2}b(1+t)\right)\left(\frac{b-a}{2}\right)dt \quad (۱۵.۵)$$

مثال شماره: ۲

مقدار واقعی انتگرال زیر برابر $I = 0.386294$ است. با استفاده از روش گاوس ۲ و ۳ نقطه‌ای این انتگرال را محاسبه می‌کنیم.

$$I = \int_1^2 \ln(x)dx$$

ابتدا محدوده‌ی انتگرال را بکمک تغییر متغیر به بازه‌ی $[-1, 1]$ انتقال می‌دهیم.

Points	Weighting Factors	Function Arguments	Truncation Error
2	$c_0 = 1.0000000$ $c_1 = 1.0000000$	$x_0 = -0.577350269$ $x_1 = 0.577350269$	$\cong f^{(4)}(\xi)$
3	$c_0 = 0.5555556$ $c_1 = 0.8888889$ $c_2 = 0.5555556$	$x_0 = -0.774596669$ $x_1 = 0.0$ $x_2 = 0.774596669$	$\cong f^{(6)}(\xi)$
4	$c_0 = 0.3478548$ $c_1 = 0.6521452$ $c_2 = 0.6521452$ $c_3 = 0.3478548$	$x_0 = -0.861136312$ $x_1 = -0.339981044$ $x_2 = 0.339981044$ $x_3 = 0.861136312$	$\cong f^{(8)}(\xi)$
5	$c_0 = 0.2369269$ $c_1 = 0.4786287$ $c_2 = 0.5688889$ $c_3 = 0.4786287$ $c_4 = 0.2369269$	$x_0 = -0.906179846$ $x_1 = -0.538469310$ $x_2 = 0.0$ $x_3 = 0.538469310$ $x_4 = 0.906179846$	$\cong f^{(10)}(\xi)$
6	$c_0 = 0.1713245$ $c_1 = 0.3607616$ $c_2 = 0.4679139$ $c_3 = 0.4679139$ $c_4 = 0.3607616$ $c_5 = 0.1713245$	$x_0 = -0.932469514$ $x_1 = -0.661209386$ $x_2 = -0.238619186$ $x_3 = 0.238619186$ $x_4 = 0.661209386$ $x_5 = 0.932469514$	$\cong f^{(12)}(\xi)$

شکل ۸.۵: جدول ضرایب و نقاط تابع برای روش گاوس–لژاندر ۲ نقطه‌ای تا ۶ نقطه‌ای.

$$I = \int_1^2 \ln(x)dx = \int_{-1}^{1} \underbrace{\ln\left(\frac{t+3}{2}\right)\frac{1}{2}}_{f(t)} dt$$

حال با استفاده از روش‌های ۲ نقطه‌ای و ۳ نقطه‌ای گاوس انتگرال تقریبی را محاسبه می‌کنیم.

$$I_2 \cong f\left(\frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right) = \frac{1}{2}\ln\left(\frac{\frac{-1}{\sqrt{3}}+3}{2}\right) + \frac{1}{2}\ln\left(\frac{\frac{1}{\sqrt{3}}+3}{2}\right) = 0.386595$$

$$I_3 \cong 0.555556\ f(-0.774597) + 0.888889\ f(0.0) + 0.555556\ f(0.774597) = 0.3863$$

۵.۱.۵ انتگرال‌های ناسره (Improper Integral)

برای انتگرال‌هایی که یکی از حدود آنها در بینهایت است اما انتگرال محدود می‌باشد، از تغییر متغیر $x \to 1/t$ می‌توانیم استفاده کنیم تا حد بی‌نهایت را به صفر انتقال دهیم. رابطه‌ی (۱۶.۵) این تغییر متغیر را نشان می‌دهد. در صورتیکه حد دیگر انتگرال با حد بینهایت علامت متفاوت داشته باشد، بازه را به دو قسمت تقسیم می‌کنیم تا در انتگرالی که تغییر متغیر می‌دهیم تغییر علامت نداشته باشیم و پیچیدگی محاسبات کاهش یابد. رابطه‌ی (۱۷.۵) نحوه‌ی تقسیم بازه به دو قسمت را نشان می‌دهد.

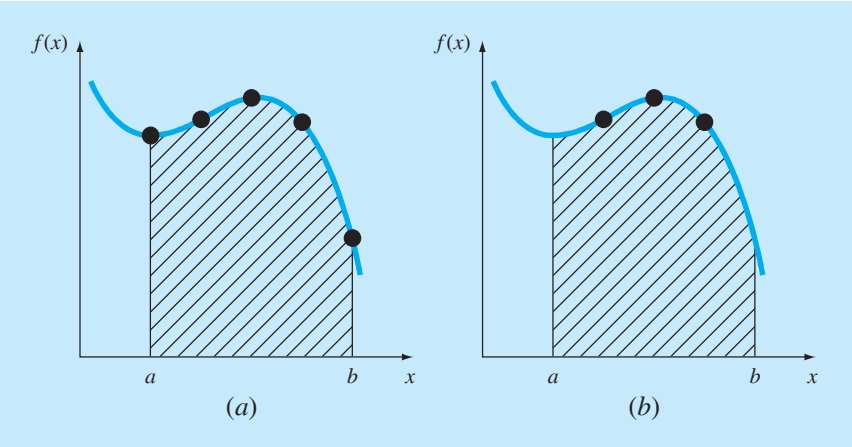
$$\int_a^b f(x)\,dx = \int_{1/a}^{1/b} \frac{1}{t^2} f(\frac{1}{t})\,dt \tag{۱۶.۵}$$

$$\int_{-\infty}^{b} f(x)\,dx = \int_{-\infty}^{-A} f(x)dx + \int_{-A}^{b} f(x)\,dx \tag{۱۷.۵}$$

در بعضی حالات نیز حد انتگرال محدود است اما مقدار تابع در حد انتگرال و یا در نقاط خاصی از دامنه که باید در آن مقدار بگیرد بی‌نهایت و یا تعریف نشده می‌شود و می‌خواهیم به نوعی از این نقاط عبور کنیم. به این منظور از روش **انتگرال‌های باز (Open Integration)** می‌توانیم استفاده کنیم. بطور مثال اگر تابع در نقاط مرزی تعریف نشده باشد، همانطور که در شکل (۹.۵) نشان داده شده است، انتگرال را بکمک نقاط میانی محاسبه می‌کنیم. رابطه‌ی (۱۸.۵) این محاسبات را برای حالت ۳ نقطه‌ی هم فاصله‌ی x_0, x_1, x_2 که تابع در نقاط مرزی x_0, x_2 تعریف نشده است را نشان می‌دهد. برای محاسبه انتگرال، همه‌ی مقادیر را برحسب نقطه‌ی میانی x_1 می‌نویسیم. انتگرال از حاصلضرب پهنا که $2h$ است در میانگین تابع که مقدار تابع در نقطه‌ی میانی است، بدست می‌آید. به این روش **قاعده‌ی نقطه‌ی میانی** گفته می‌شود.

$$I = \int_{x_0}^{x_2} f(x)dx = \int_{x_1-h}^{x_1+h} f(x)dx \cong 2hf(x_1) \tag{۱۸.۵}$$

روش باز را می‌توان به بیش از ۳ نقطه تعمیم داد. بطور مثال رابطه‌ی (۱۹.۵) نحوه‌ی بدست آوردن روش باز برای ۵ نقطه را نشان می‌دهد. برای محاسبه انتگرال، چون تنها در ۳ نقطه‌ی میانی می‌خواهیم تابع را محاسبه کنیم، لذا برای درونیابی از چندجمله‌ای درجه ۲ و بطور مثال از روش لاگرانژ استفاده می‌کنیم. خطای روش مانند قاعده‌ی سیمپسون که از ۳ نقطه استفاده می‌کند از مرتبه‌ی $\mathcal{O}(h^5)$ می‌باشد. به نحوه‌ی نوشتن حدود انتگرال برحسب نقطه‌ی x_1 توجه کنید. در رابطه‌ی نهایی فقط مقدار تابع درنقاط میانی یعنی f_1, f_2, f_3 استفاده شده است و نیاز به مقدار تابع در مرزهای بازه یعنی x_0, x_4 که در آنها تابع تعریف نشده است، نداریم.



شکل ۹.۵: انتگرال به روش بسته (a) و باز (b).

$$
\begin{aligned}
I &= \int_{x_0}^{x_4} f(x)dx = \int_{x_1-h}^{x_1+3h} f(x)dx \\
&\cong \int_{x_1-h}^{x_1+3h} P_2(x)dx \\
&= \int_{x_1-h}^{x_1+3h} \left(f_1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + f_2 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} \right. \\
&\qquad \left. + f_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} \right) dx \\
I &= \frac{4}{3}h(2f_1 - f_2 + 2f_3) + \mathcal{O}(h^5) \qquad (۱۹.۵)
\end{aligned}
$$

مثال شماره: ۳

انتگرال زیر را محاسبه نمایید:

$$I = \int_{-\infty}^{1} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$$

انتگرال را به دو قسمت تقسیم می‌کنیم. به نحوی که قسمت دارای بینهایت تغییر علامت نداشته

باشیم و هر دو حد در ناحیه‌ی xهای منفی باشد. تا در هنگام تغییر متغیر با تغییر علامت مواجه نباشیم. قسمت دارای حد بینهایت را تغییر متغیر $x \to 1/t$ می‌دهیم تا حد بینهایت به صفر تبدیل شود. برای حل این قسمت از روش نقطه‌ی میانی استفاده می‌کنیم. انتگرال بین -2 تا 1 را نیز از سیمپسون استفاده می‌کنیم. در نهایت با جمع نتایج انتگرال کل را محاسبه می‌کنیم.

$$I = \frac{1}{\sqrt{2\pi}} \left(\int_{-\infty}^{-2} e^{-x^2/2} dx + \int_{-2}^{1} e^{-x^2/2} dx \right)$$

$$\begin{aligned}
I_1 &= \int_{-\infty}^{-2} e^{-x^2/2} dx = \int_{-1/2}^{0} \frac{1}{t^2} e^{-1/(2t^2)} dt \\
&\cong \frac{1}{8} \left(f(x_{-7/16}) + f(x_{-5/16}) + f(x_{-3/16}) + f(x_{-1/16}) \right) \\
&\cong \frac{1}{8} (0.3833 + 0.0612 + 0 + 0) = 0.0556 \\
I_2 &= \int_{-2}^{1} e^{-x^2/2} dx \\
&\cong (1 - (-2)) \frac{0.1353 + 4(0.3247 + 0.8825 + 0.8825) + 2(0.6065 + 1) + 0.6065}{3(6)} = 2.0 \\
I &\cong \frac{1}{\sqrt{2\pi}} (0.0556 + 2.0523) = 0.8409
\end{aligned}$$

۲.۵ مشتقگیری عددی

برای مشتق توابع پیچیده و توابعی که با جدول اعداد ارائه می‌شوند، می‌توان از تقریب چندجمله‌ای‌های درونیاب (نیوتن یا لاگرانژ) تابع مذکور استفاده نمود.

$$\frac{df(x)}{dx} \cong \frac{d}{dx} P_n(x) \qquad (۲۰.۵)$$

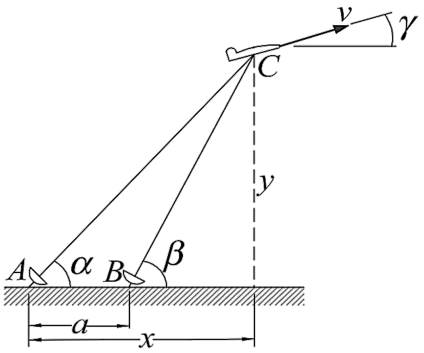
خطای این روش، با مشتق گرفتن از خطای چندجمله‌ای درونیاب بدست می‌آید.

مثال شماره: ۴

رادارهای A و B از هم فاصله‌ی $a = 500\,(m)$ دارند. این رادارها موقعیت هواپیمای C را به ترتیب با زاویه‌ی α و β در هر ثانیه ثبت می‌کنند. که برای ۳ ثانیه در جدول زیر آورده شده است. سرعت هواپیما v و زاویه اوج آن γ را در $t = 10\,(\text{sec})$ محاسبه کنید. مختصات هواپیما از رابطه‌ی زیر محاسبه می‌شود.

$$x = a\frac{\tan\beta}{\tan\beta - \tan\alpha} \qquad y = x\tan\alpha$$

t (s)	9	10	11
α (deg)	54.80	54.06	53.34
β (deg)	65.59	64.59	63.62



پاسخ: ابتدا با استفاده از داده‌های جدول، موقعیت هواپیما را در زمان‌های داده شده محاسبه می‌کنیم. سپس تابع $x(t)$ و $y(t)$ را از درجه‌ی ۲ درونیابی می‌کنیم.

t (s)	9	10	11
x	1401.92	1450.5	1498.64
y	1987.35	2000.84	2013.51

$$\begin{aligned} x(t) &= (1401.92)(\frac{t-10}{9-10})(\frac{t-11}{9-11}) + (1450.5)(\frac{t-9}{10-9})(\frac{t-11}{10-11}) \\ &+ (1498.64)(\frac{t-9}{11-9})(\frac{t-10}{11-10}) \\ x(t) &= 945.124 + 52.7136\,t - 0.217632\,t^2 \\ v_x(t) &= 52.7136 - 0.435264\,t \end{aligned}$$

$$\begin{aligned} y(t) &= (1987.35)(\frac{t-10}{9-10})(\frac{t-11}{9-11}) + (2000.84)(\frac{t-9}{10-9})(\frac{t-11}{10-11}) \\ &+ (2013.51)(\frac{t-9}{11-9})(\frac{t-10}{11-10}) \\ y(t) &= 1828.86 + 21.3132\,t - 0.4115\,t^2 \\ v_y(t) &= 21.3132 - 0.822999\,t \end{aligned}$$

سپس در لحظه‌ی $t = 10$ (sec) سرعت و زاویه‌ی اوج را می‌توان محاسبه کرد.

$$\begin{aligned} \vec{v} &= 48.361\,\hat{e}_x + 13.0832\,\hat{e}_y \quad \Rightarrow v = 50.0985\,(m/s) \\ \tan\gamma &= \frac{v_y}{v_x} \quad \Rightarrow \gamma = 0.0046^\circ \end{aligned}$$

۱.۲.۵ مشتقات با دقت بالا

علاوه بر استفاده از چندجمله‌ای‌های درونیاب، از بسط تیلور توابع نیز می‌توان مشتقات عددی را بدست آورد. این روش به نام **تفاضلات محدود (Finite Difference)** شناخته می‌شود. کاربرد این روش بیشتر برای حل معادلات دیفرانسیل می‌باشد. برای شروع و آشنایی با این روش مشتق مرتبه‌ی اول یک تابع را بدست می‌آوریم. فرض می‌کنیم تابع را بصورت جدول اعداد در نقاط هم فاصله‌ی x_i داریم که مقادیر تابع را به شکل $f_i = f(x_i)$ نمایش می‌دهیم. فاصله‌ی نقاط از هم h می‌باشد. هدف محاسبه‌ی مشتق مرتبه اول در نقطه‌ی x_i است. مقدار تابع در نقطه‌ی x_{i+1} را از روی بسط تیلور تابع حول نقطه‌ی x_i محاسبه می‌کنیم. بسط را تا مرتبه‌ی اول مشتق نگه می‌داریم و الباقی را خطای برشی منظور می‌کنیم. رابطه‌ی (۲۱.۵) این بسط را نشان می‌دهد. سپس با توجه به اینکه مقادیر تابع در نقاط x_i و x_{i+1} مشخص است، پس مشتق تابع را برحسب البقی جملات محاسبه می‌کنیم. خطای بسط تیلور از مرتبه‌ی $\mathcal{O}(h^2)$ است. اما هنگامی که مشتق را محاسبه می‌کنیم با تقسیم کردن طرفین معادله به ضریب مشتق یعنی h مرتبه‌ی دقت مشتق نیز به $\mathcal{O}(h)$ کاهش می‌یابد. از آنجاییکه برای محاسبه‌ی مشتق در نقطه‌ی x_i از نقاط جلوتر یعنی x_{i+1} استفاده کردیم به معادله‌ی (۲۱.۵) رابطه‌ی **تفاضل محدود پیشرو (Forward Finite Difference)** برای مشتق مرتبه‌ی اول گفته می‌شود.

$$\begin{aligned} f(x_{i+1}) &= f(x_i) + hf'(x_i) + \mathcal{O}(h^2) \\ &\Rightarrow f'_i = \frac{f_{i+1} - f_i}{h} + \mathcal{O}(h) \end{aligned} \qquad (۲۱.۵)$$

با استفاده از نقطه‌ی پیشین نیز می‌توان مشتق را محاسبه نمود. معادله‌ی (۲۲.۵) نحوه‌ی محاسبه‌ی مشتق مرتبه‌ی اول با استفاده از بسط تیلورحول نقطه‌ی x_i برای نقطه‌ی پیشین x_{i-1} را نشان می‌دهد. به این معادله رابطه‌ی **تفاضل محدود پسرو (Backward Finite Difference)** گفته می‌شود. دقت این معادله نیز از مرتبه‌ی $\mathcal{O}(h)$ می‌باشد و مانند حالت پیشرو، برای محاسبه‌ی مشتق، فقط از مقدار تابع در ۲ نقطه استفاده شده است.

$$f_{i-1} = f_i - hf'_i + \mathcal{O}(h^2) \Rightarrow f'_i = \frac{f_i - f_{i-1}}{h} + \mathcal{O}(h) \qquad (۲۲.۵)$$

دقت مشتق مرتبه‌ی اول را می‌توان با افزایش تعداد نقاط درگیر در محاسبه‌ی آن افزایش داد. بطور مثال فرض کنید بسط تیلور حول نقطه‌ی x_i را برای نقاط x_{i+1} و x_{i-1} بنویسیم و بسط را تا مشتق مرتبه‌ی ۲ نگه داریم. رابطه‌ی (۲۳.۵) این معادلات را نشان می‌دهد. مقادیر تابع در تمام نقاط دانسته فرض می‌شود. پس تنها مجهول این دو معادله مشتق مرتبه‌ی اول و دوم تابع در نقطه‌ی x_i می‌باشد. با تفاضل دو معادله از هم مشتق مرتبه‌ی دوم حذف و مشتق مرتبه‌ی اول بدست می‌آید و خطای این مشتق از مرتبه‌ی $\mathcal{O}(h^2)$ است. دقت کنید که در محاسبه‌ی این مشتق از مقدار تابع در نقاط پسین و پیشین هر دو استفاده شده است. به این روش محاسبه‌ی مشتق که دقت بالاتری نسبت به پیشرو و پسرو دارد **تفاضل محدود مرکزی (Central Finite Difference)** گفته می‌شود.

$$\begin{cases} f_{i+1} &= f_i + hf'_i + \dfrac{h^2}{2}f''_i + \mathcal{O}(h^3) \\ f_{i-1} &= f_i - hf'_i + \dfrac{h^2}{2}f''_i + \mathcal{O}(h^3) \end{cases} \Rightarrow f'_i = \frac{f_{i+1} - f_{i-1}}{2h} + \mathcal{O}(h^2) \quad (۲۳.۵)$$

درصورت نیاز به دقت‌های بالاتر تفاضل محدود پیشرو و یا پسرو باید تعداد نقاط بیشتری را در محاسبه‌ی آن‌ها درگیر کنیم. بطور مثال روابط (۲۴.۵ و ۲۵.۵) با استفاده از مقدار تابع در ۲ نقطه‌ی جلوتر و یا ۲ نقطه‌ی عقب‌تر رابطه‌ی جدیدی برای مشتق مرتبه‌ی اول ارائه می‌دهند. همانطور که دیده می‌شود در هر دو رابطه دقت محاسبات از $\mathcal{O}(h)$ به $\mathcal{O}(h^2)$ ارتقاء پیدا کرده است. این دو رابطه تفاضلات محدود پیشرو و پسرو با دقت $\mathcal{O}(h^2)$ برای مشتق مرتبه‌ی اول نامیده می‌شوند.

$$\begin{cases} f_{i+1} &= f_i + hf'_i + \dfrac{h^2}{2}f''_i + \mathcal{O}(h^3) \\ f_{i+2} &= f_i + 2hf'_i + \dfrac{(2h)^2}{2}f''_i + \mathcal{O}(h^3) \end{cases} \Rightarrow f'_i = \frac{-f_{i+2} + 4f_{i+1} - 3f_i}{2h} + \mathcal{O}(h^2)$$

(۲۴.۵)

$$\begin{cases} f_{i-1} &= f_i - hf'_i + \dfrac{h^2}{2}f''_i + \mathcal{O}(h^3) \\ f_{i-2} &= f_i - 2hf'_i + \dfrac{(2h)^2}{2}f''_i + \mathcal{O}(h^3) \end{cases} \Rightarrow f'_i = \frac{f_{i-2} - 4f_{i-1} + 3f_i}{2h} + \mathcal{O}(h^2)$$

(۲۵.۵)

به عنوان آخرین مثال رابطه‌ی مشتق مرتبه‌ی دوم مرکزی با دقت $\mathcal{O}(h^2)$ را محاسبه می‌کنیم. در رابطه‌ی (۲۶.۵) برای محاسبه مشتق مرتبه ۲ بسط تیلور را حول نقطه‌ی x_i برای نقطه‌ی پسین و

پیشین آن می‌نویسیم. دو معادله تشکیل می‌شود و مجهول اصلی مشتق مرتبه‌ی دو در نقطه‌ی x_i است. مشتق دوم بر حسب مقدار تابع در ۳ نقطه‌ی x_{i-1}, x_i, x_{i+1} محاسبه می‌شود. خطا از مرتبه‌ی $\mathcal{O}(h^2)$ می‌باشد.

$$\begin{cases} f_{i+1} &= f_i + hf_i' + \frac{h^2}{2}f_i'' + \frac{h^3}{3!}f_i^{(3)} + \mathcal{O}(h^4) \\ f_{i-1} &= f_i - hf_i' + \frac{h^2}{2}f_i'' - \frac{h^3}{3!}f_i^{(3)} + \mathcal{O}(h^4) \end{cases} \Rightarrow f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \mathcal{O}(h^2) \tag{۲۶.۵}$$

در جدول (۱.۵) روابط تفاضلات محدود پیشرو، پسرو و مرکزی برای مشتقات مرتبه‌ی اول و دوم آورده شده است. برای هر مشتق، روابط برای ۲ دقت متفاوت محاسبه شده‌اند. همانطور که دیده می‌شود برای افزایش دقت نیاز به داشتن تابع در نقاط بیشتری هستیم.

Method	Formulas	Error	Formulas	Er
Forward	$f_i' = \frac{f_{i+1} - f_i}{h}$	$\mathcal{O}(h)$	$f_i'' = \frac{f_{i+2} - 2f_{i+1} + f_i}{h^2}$	$\mathcal{O}$
	$f_i' = \frac{-f_{i+2} + 4f_{i+1} - 3f_i}{2h}$	$\mathcal{O}(h^2)$	$f_i'' = \frac{-f_{i+3} + 4f_{i+2} - 5f_{i+1} + 2f_i}{h^2}$	$\mathcal{O}($
Backward	$f_i' = \frac{f_i - f_{i-1}}{h}$	$\mathcal{O}(h)$	$f_i'' = \frac{f_i - 2f_{i-1} + f_{i-2}}{h^2}$	$\mathcal{O}$
	$f_i' = \frac{3f_i - 4f_{i-1} + f_{i-2}}{2h}$	$\mathcal{O}(h^2)$	$f_i'' = \frac{2f_i - 5f_{i-1} + 4f_{i-2} - f_{i-3}}{h^2}$	$\mathcal{O}($
Centered	$f_i' = \frac{f_{i+1} - f_{i-1}}{2h}$	$\mathcal{O}(h^2)$	$f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$	$\mathcal{O}($
	$f_i' = \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12h}$	$\mathcal{O}(h^4)$	$f_i'' = \frac{-f_{i+2} + 16f_{i+1} - 30f_i + 16f_{i-1} - f_{i-2}}{12h^2}$	$\mathcal{O}($

جدول ۱.۵ : مشتقات مرتبه اول و دوم به روش‌های پیشرو، پسرو و مرکزی با دو دقت متفاوت.

مثال شماره: ۵

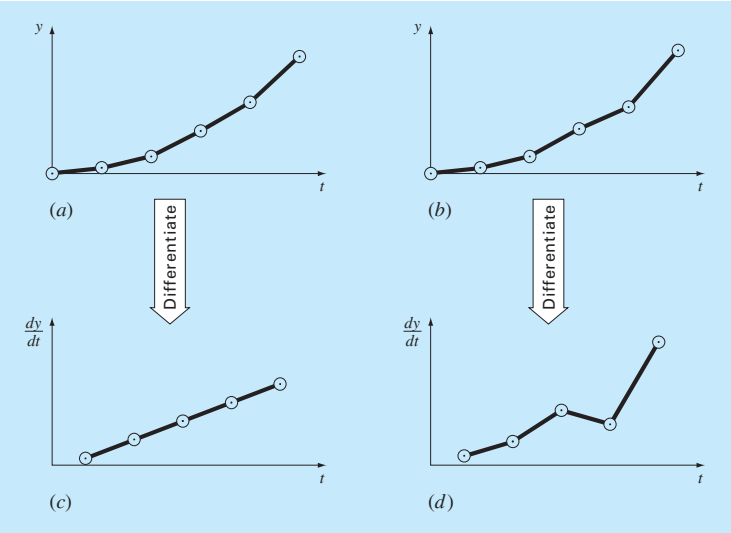
مختصات یک روبات برحسب زمان با جدول زیر داده شده است. سرعت این ربات را در $t = 1(\text{sec})$ با دقت $\mathcal{O}(h^2)$ محاسبه نمایید.

t	x	y
0.0	0.500	1.500
0.1	0.564	1.536
0.2	0.678	1.574
0.3	0.814	1.598
0.4	0.943	1.591
0.5	1.038	1.538
0.6	1.069	1.421
0.7	1.009	1.225
0.8	0.830	0.934
0.9	0.503	0.531
1.0	0.000	0.000

پاسخ: از آنجاییکه سرعت را در آخرین نقطه‌ی جدول باید محاسبه کنیم و داده‌ای بعد از آن نداریم و باید بکمک داده‌های قبلی مشتق (سرعت) را محاسبه کنیم، لذا باید از روش پسرو استفاده کنیم. با توجه به دقت خواسته شده نیز از رابطه‌ی زیر مشتق را برای هر دو راستا محاسبه می‌کنیم.

$$
\begin{aligned}
f_i' &= \frac{3f_i - 4f_{i-1} + f_{i-2}}{2h} \\
\dot{x} &= \frac{3(0.000) - 4(0.503) + (0.830)}{2(0.1)} = -5.91(\text{m/sec}) \\
\dot{y} &= \frac{3(0.000) - 4(0.531) + (0.934)}{2(0.1)} = -5.95(\text{m/sec})
\end{aligned}
$$

مشتق‌گیری عددی بسیار ناپایدار است و کوچکترین خطاها در داده منجر به خطای زیاد در مشتق می‌شود. شکل (۱۰.۵) نشان می‌دهد که چگونه یک خطای کم در تابع (b)، شکل مشتق آن را کاملا تغییر می‌دهد و از شکلی که انتظار داریم (c) تبدیل به شکل (d) می‌شود. لذا برای مشتق‌گیری از داده‌های غیردقیق، درونیابی توصیه نمی‌شود و بهتر است با استفاده از رگرسیون داده‌ها را برازش کنیم و سپس از تابع حاصل مشتق بگیریم.



شکل ۱۰.۵: مشتق‌گیری از داده‌هایی که دقیق نیستند ،(b) منجر به نتایج بسیار دور از انتظار (d) می‌شود.

تمرین شماره: ۱

درجه دقت روش سیمپسون را محاسبه نمایید.
راهنمایی: برای α های مختلف، انتگرال زیر را به روش تحلیلی و سیمپسون محاسبه نمایید. بازه‌ی انتگرال‌گیری را بصورت تصادفی انتخاب کنید.

$$I = \int_a^b x^\alpha \, dx$$

تمرین شماره: ۲

مسیر حرکت یک موشک با روابط زیر داده شده است. طول مسیر حرکت موشک را در ۱ ثانیه اول حرکت خود محاسبه نمایید. برای انتگرالگیری از روش سیمپسون مرکب استفاده نمایید. تعداد تقسیمات بازه را به نحوی انتخاب نمایید که خطای نسبی برشی کمتر از یک درصد باشد.

$$x(t) = 0.5 + 0.3t + 3.9t^2 - 4.7t^3$$
$$y(t) = 1.5 + 0.3t + 0.9t^2 - 2.7t^3$$

راهنمایی: برای محاسبه‌ی طول مسیر بین زمان‌های t_1 و t_2 از رابطه‌ی زیر استفاده کنید.

$$\mathcal{S} = \int_{t_1}^{t^2} \sqrt{x'(t)^2 + y'(t)^2} dt$$

تمرین شماره: ۳

انتگرال‌های زیر را به ترتیب اولی با روش ذوزنقه‌ی مرکب، دومی با روش سیمپسون مرکب و سومی را با روش لژاندرـ گاوس با حداکثر خطای نسبی یک درصد محاسبه نمایید.

- $I_1 = \int_0^{2\pi} \frac{\sin t}{t}\, dt$
- $I_2 = \int_{-\pi/2}^{\pi/2} t \cot(t)\, dt$
- $I_3 = \int_{-\infty}^{\infty} \mathrm{sech}^2(t)\, dt$

تمرین شماره: ۴

دبی حجمی سیال درون یک لوله به شعاع R از رابطه‌ی زیر بدست می‌آید:

$$Q = \int_0^R 2\pi r v\, dr$$

که در آن v سرعت سیال در لوله و r فاصله از مرکز لوله است. در یک آزمایش مقادیر سرعت سیال برحسب r در یک لوله‌ی به قطر $40\ (cm)$ در جدول زیر داده شده است. دبی حجمی عبور سیال را با روش‌های زیر محاسبه نمایید.

الف) یک چندجمله‌ای به داده‌های جدول برازش نمایید و با انتگرال تحلیلی مقدار دبی را محاسبه نمایید.

ب) با استفاده از روش سیمپسون از داده‌های جدول انتگرال بگیرید و دبی را محاسبه نمایید.

Radius: r (cm)	0.0	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0
Velocity: v (m/s)	0.914	0.890	0.847	0.795	0.719	0.543	0.427	0.204	0

تمرین شماره: ۵

الف) مختصات حرکت یک کوادکوپتر، برحسب زمان با جدول زیر داده شده است. پرنده در ارتفاع ثابت حرکت می‌کند، لذا مسیر فقط در صفحه‌ی $x-y$ داده شده است. مسیر حرکت این پرنده را در صفحه‌ی $x-y$ رسم نمایید.
ب) رابطه‌ی مشتق مرتبه‌ی اول مرکزی از دقت $\mathcal{O}(h^4)$ را بدست آورید.
ج) بیشینه‌ی سرعت و لحظه‌ی رسیدن به این سرعت توسط پرنده را با استفاده از رابطه‌ی بدست آمده در قسمت ب محاسبه نمایید. در لحظات اول و آخر مسیر می‌توانید محاسبات را با دقت $\mathcal{O}(h^2)$ انجام دهید.

t(sec)	x(m)	y(m)
0.000	0.000	4.500
0.165	0.036	4.725
0.331	0.274	5.319
0.496	0.863	6.060
0.661	1.854	6.664
0.827	3.186	6.866
0.992	4.694	6.504
1.157	6.144	5.555
1.323	7.288	4.127
1.488	7.918	2.406
1.653	7.918	0.594
1.819	7.288	-1.156
1.984	6.144	-2.764
2.150	4.694	-4.220
2.315	3.186	-5.549
2.480	1.854	-6.766
2.646	0.863	-7.846
2.811	0.274	-8.721
2.976	0.036	-9.298
3.142	0.000	-9.500

تمرین شماره: ۶

موقعیت یک هواپیما توسط رادار در مختصات قطبی در جدول زیر داده شده است. بردار سرعت و شتاب هواپیما را در $t = 206\ (\text{sec})$ با دقیق‌ترین روش ممکن محاسبه نمایید.

t (sec)	200	202	204	206	208	210
θ (rad)	0.75	0.72	0.70	0.68	0.67	0.66
r (m)	5120	5370	5560	5800	6030	6240

راهنمایی: در مختصات قطبی بردارهای سرعت و شتاب از روابط زیر محاسبه می‌شوند.

$$\vec{v} = \dot{r}\vec{e}_r + r\dot{\theta}\vec{e}_\theta \qquad \text{و} \qquad \vec{a} = (\ddot{r} - r\dot{\theta}^2)\vec{e}_r + (r\ddot{\theta} + 2\dot{r}\dot{\theta})\vec{e}_\theta$$

فصل ۶

معادلات دیفرانسیل معمولی

به معادلاتی که از یک تابع ناشناخته و مشتقات آن تشکیل شده اند، معادلات دیفرانسیل گفته می شود. معادلات دیفرانسیل نقشی اساسی در علوم و مهندسی بازی می‌کنند. در معادله‌ی دیفرانسیل، تابع ناشناخته، کمیت متمایزی است که متغیر وابسته نامیده می‌شود. کمیتی را که متغیر وابسته، نسبت به آن تغییر می‌کند، متغیر مستقل می‌نامند. وقتی تابع شامل یک متغیر مستقل باشد ، **معادله دیفرانسیل معمولی (Ordinary Differential Equation)** گفته می‌شود. در کنار این نوع معادله، **معادله دیفرانسیل جزئی (Partial Differential Equation)** وجود دارد که شامل دو یا چند متغیر مستقل است. معادلات دیفرانسیل از نظر مرتبه که بالاترین مشتق در معادله است، نیز تقسیم‌بندی می‌شوند. بطور مثال معادله‌ی زیر یک معادله‌ی دیفرانسیل معمولی مرتبه دوم است.

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0 \qquad (۱.۶)$$

که در آن x متغیر وابسته و t متغیر مستقل است. همچنین m، c و k پارامترهای معادله می‌باشند. این معادله‌ی دیفرانسیل مکان x یک جسم به جرم m که به یک فنر به سختی k متصل است و در محیطی با ثابت میرایی c حرکت می‌کند را بر حسب زمان t توصیف می‌کند. سرعت این جسم از رابطه‌ی زیر بدست می‌آید.

$$v = \frac{dx}{dt} \qquad (۲.۶)$$

با جایگذاری رابطه (۲.۶) در معادله دیفرانسیل (۱.۶) معادله دیفرانسیل (۳.۶) حاصل می‌شود.

$$\begin{aligned} m\frac{d}{dt}(\frac{dx}{dt}) + c(\frac{dx}{dt}) + kx = 0 \quad &\Rightarrow \quad m\frac{dv}{dt} + cv + kx = 0 \\ &\Rightarrow \quad \frac{dv}{dt} = -\frac{c}{m}v - \frac{c}{m}x \end{aligned} \qquad (۳.۶)$$

بطور خلاصه با این ترفند، معادله دیفرانسیل مرتبه‌ی دو (۱.۶) را به دو معادله‌ی دیفرانسیل مرتبه اول (۲.۶ و ۳.۶) تبدیل می‌کنیم. در این فصل روش‌های عددی را مطالعه می‌کنیم که معادلات دیفرانسیل مرتبه اول به شکل عمومی زیر را می‌توانند حل کنند.

$$\frac{dx}{dt} = f(x,t) \tag{۴.۶}$$

لذا برای حل معادلات دیفرانسیل مرتبه بالاتر، ابتدا باید معادله را بکمک تغییر متغیرهایی مانند (۲.۶) به دستگاه‌ای از معادلات دیفرانسیل مرتبه اول تبدیل کنیم.

۱.۶ روش بسط تیلور

یکی از روش‌های کارآمد برای حل معادلات دیفرانسیل معمولی استفاده از بسط تیلور متغیر وابسته است. برای شروع و بررسی این روش، معادله‌ی زیر با شرایط مشخص شده را در نظر بگیرید.

$$\begin{cases} \dfrac{dy}{dx} = f(x,y) \\ \\ y(0) = y_0 \qquad \text{شرایط اولیه} \end{cases} \tag{۵.۶}$$

این معادله مانند معادله (۴.۶) معادله دیفرانسیل مرتبه اول می‌باشد که در آن متغیر وابسته و مستقل به ترتیب y و x می‌باشند. به عنوان یک شرط مورد نیاز برای حل خصوصی معادله‌ی دیفرانسیل مرتبه اول، مقدار متغیر وابسته y را بازاء $x = 0$ داده شده است که به آن شرایط اولیه نیز گفته می‌شود. هدف از حل معادله (۵.۶) بدست آوردن متغیر وابسته y بازاء مقادیر دیگر x است. در این قسمت، در حالت کلی فرض می‌کنیم مقدار y در نقطه‌ی x_i یعنی $y_i = y(x_i)$ را داریم و بکمک بسط تیلور مقدار آن در نقطه‌ی x_{i+1} را محاسبه می‌کنیم.

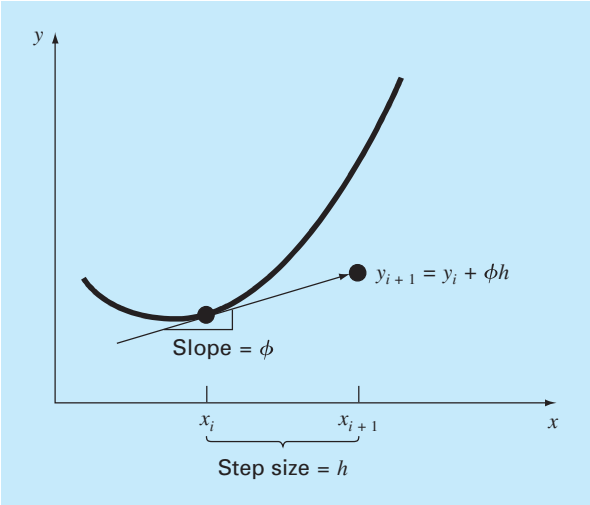
$$\begin{aligned} y_{i+1} \;\cong\; & y_i + y_i' h + \frac{y_i'' h^2}{2} + \frac{y_i^{(3)} h^3}{3!} + \ldots + \frac{y_i^{(n)} h^n}{n!} \\ & \text{where: } y_i' = \left.\frac{dy}{dx}\right|_{x=x_i} = f(x_i, y_i), \quad y_i'' = \left.\frac{df(x,y(x))}{dx}\right|_{x=x_i,\, y=y_i}, \ldots \\ & R_n = \frac{h^{n+1}}{(n+1)!} \left(\frac{d^n f(x,y)}{dx^n}\right)\Bigg|_{x=\xi,\, y=y(\xi)} \end{aligned} \tag{۶.۶}$$

که در آن h فاصله‌ی دو نقطه‌ی x_i و x_{i+1} است. در این روش، گام به گام، مشتقات مراتب بالاتر y از روی تابع $f(x,y)$ باید محاسبه شوند. خطای ارائه شده در رابطه‌ی (۶.۶) **خطای برشی موضعی**

است. اما باید توجه داشت که با پیش رفتن محاسبات به iهای بالاتر، خطای محاسبات نقاط قبلی به خطای این نقطه که در حال محاسبه هستیم انتشار می‌یابد. این خطای برشی را **خطای سراسری** می‌نامیم که یک مرتبه از خطای موضعی، تقریب بدتری است. قطعا استفاده از تعداد بیشتر جملات بسط برای تقریب، دقت محاسبات را افزایش و خطای برشی را کاهش می‌دهد. در اولین تقریب، از بسط تیلور برای مرتبه‌ی $n = 1$ استفاده می‌کنیم. این تقریب **روش اویلر (Euler's Method)** نامیده می‌شود.

$$y_{i+1} \cong y_i + f(x_i, y_i)h \tag{۷.۶}$$

$$R_1 = \frac{1}{2}h^2 f'(x_i, y_i) = \mathcal{O}(h^2)$$



شکل ۱.۶ : روش اویلر برای حل معادلات دیفرانسیل معمولی.

در روش اویلر اختلاف مقدار تابع در دو نقطه‌ی x_i و x_{i+1} از حاصل‌ضرب اندازه‌ی گام در شیب تابع در نقطه‌ی x_i که همان $f(x_i, y_i)$ است بدست می‌آید. در شکل (۱.۶) این جزییات نشان داده شده است. خطای برشی موضعی روش اویلر از مرتبه‌ی $\mathcal{O}(h^2)$ است. از آنجاییکه در هر گام از محاسبه به خطای برشی سراسری یک واحد خطای برشی موضعی افزوده می‌شود، برای محاسبه‌ی خطای برشی سراسری در انتهای بازه‌ی محاسباتی، باید خطای موضعی را در تعداد گام‌های برداشته شده ضرب کنیم. تعداد گام‌ها نیز از تقسیم بازه بر فاصله‌ی گام‌ها h بدست می‌آید. لذا خطای برشی سراسری در روش اویلر از مرتبه $\mathcal{O}(h)$ است. در مثال زیر با جزییات بیشتری از محاسبات در روش اویلر آشنا می‌شویم.

مثال شماره: ۱

معادله دیفرانسیل زیر را با شرایط اولیه داده شده و با استفاده از روش اویلر در بازه‌ی $x \in [0, 5]$ محاسبه نمایید. اندازه‌ی گام‌ها را $h = 0.5$ بگیرید.

$$\frac{dy}{dx} = 4\exp(0.8x) - 0.5y, \qquad y(0) = 2$$

پاسخ تحلیلی معادله دیفرانسیل قابل محاسبه است و در زیر آورده شده است. برای بررسی دقت محاسبات می‌توانید از این پاسخ استفاده کنید.

$$y_{\text{exact}} = \frac{4}{1.3}(\exp(0.8x) - \exp(-0.5x)) + 2\exp(-0.5x)$$

با استفاده از رابطه‌ی (۷.۶) برای هر گام رابطه‌ی بازگشتی زیر را بدست می‌آوریم

$$y_{i+1} = y_i + h(4\exp(0.8x_i) - 0.5y_i)$$
$$\text{(e.g.) } y(0.5) = y_1 = 2 + 0.5(4\exp(0) - 0.5(2)) = 3.5$$

با استفاده از پاسخ تحلیلی که در صورت مساله داده شده است، خطای نسبی دقیق را نیز محاسبه می‌کنیم. در جدول زیر به ترتیب از ستون چپ مقادیر x، پاسخ‌های اویلر، تحلیلی و خطای نسبی دقیق آورده شده است. خطای نسبی گام به گام در حال افزایش است.

x	y_{exact}	y_{euler}	ϵ_t
0.5	3.752	3.5	0.067
1.	6.195	5.609	0.095
1.5	9.707	8.658	0.108
2.	14.844	13.133	0.115
2.5	22.427	19.756	0.119
3.	33.677	29.595	0.121
3.5	50.412	44.243	0.122
4.	75.339	66.071	0.123
4.5	112.496	98.619	0.123
5.	167.906	147.16	0.124

برای بالا بردن دقت و کاهش مرتبه‌ی خطای برشی می‌توانیم از مرتبه‌های بالاتر بسط تیلور استفاده کنیم که به همان نسبت پیچیدگی محاسبات نیز افزایش می‌یابد. بطور مثال با بسط تیلور مرتبه‌ی $n = 2$

ادامه می‌دهیم.

$$
\begin{aligned}
y_{i+1} &\cong y_i + f(x_i, y_i)h + f'(x_i, y_i)\frac{h^2}{2} \\
f'(x_i, y_i) &= \frac{d}{dx} f(x, y)\Big|_{x_i, y_i} = \frac{\partial f}{\partial x} + \underbrace{(\frac{dy}{dx})}_{f(x,y)} \frac{\partial f}{\partial y}\Big|_{x_i, y_i} \\
y_{i+1} &\cong y_i + f(x_i, y_i)h + (\partial_x f + f\ \partial_y f)\ \frac{h^2}{2}\Big|_{x_i, y_i} \qquad (۸.۶)
\end{aligned}
$$

مثال شماره: ۲

مثال (۱) را با روش تیلور مرتبه‌ی $n = 2$ حل و پاسخ‌ها را مقایسه کنید.

برای بدست آوردن رابطه‌ی بازگشتی از رابطه‌ی (۸.۶) باید استفاده کنیم. برای شروع نیاز به مشتقات جزئی تابع $f(x, y)$ داریم.

$$
f(x, y) = 4\exp(0.8x) - 0.5y \quad \Rightarrow \begin{cases} \partial_x f(x, y) = 3.2\exp(0.8x) \\ \partial_y f(x, y) = -0.5 \end{cases}
$$

$$
\begin{aligned}
y_{i+1} &\cong y_i + f(x_i, y_i)h + (\partial_x f + f\ \partial_y f)\ \frac{h^2}{2}\Big|_{x_i,\, y_i} \\
y_{i+1} &\cong y_i + (4\exp(0.8x_i) - 0.5y_i)h \\
&\quad + \Big(3.2\exp(0.8x_i) + (4\exp(0.8x_i) - 0.5y_i)(-0.5)\Big)h^2/2 \\
y_{i+1} &\cong y_i + \Big(4\exp(0.8x_i) - 0.5y_i\Big)h + \Big(1.2\exp(0.8x_i) + 0.25y_i\Big)h^2/2
\end{aligned}
$$

در جدول زیر پاسخ دقیق (ستون ۲)، تیلور مرتبه اول یا اویلر(ستون ۳) و تیلور مرتبه دوم (ستون ۵) آورده شده است. خطای نسبی دقیق برای تیلور مرتبه اول و دوم در ستون‌های ۴ و ۶ مشاهده می‌شوند. مقایسه دو ستون خطاها تایید می‌کند که خطای تیلور مرتبه دوم $\mathcal{O}(h^2)$ و تیلور مرتبه اول $\mathcal{O}(h)$ است.

x	y_{exact}	y_{euler}	ϵ_t	$y_{taylor2}$	ϵ_{t2}
0.5	3.752	3.5	0.067	3.712	0.01
1.	6.195	5.609	0.095	6.108	0.014
1.5	9.707	8.658	0.108	9.557	0.015
2.	14.844	13.133	0.115	14.604	0.016
2.5	22.427	19.756	0.119	22.059	0.016
3.	33.677	29.595	0.121	33.12	0.017
3.5	50.412	44.243	0.122	49.575	0.017
4.	75.339	66.071	0.123	74.086	0.017
4.5	112.496	98.619	0.123	110.625	0.017
5.	167.906	147.16	0.124	165.112	0.017

۲.۶ روش رانگ‌ـ کوتا

در روش تیلور دیدیم که استفاده از مراتب بالاتر بسط، دقت محاسبات را افزایش می‌دهد، هرچند همانطور که در روابط (۸.۶) دیده می‌شود، پیچیدگی محاسبات بخصوص محاسبه مشتقات بالاتر از طریق تابع $f(x,y)$ افزایش می‌یابد. در روش رانگ‌ـ کوتا سعی می‌کنیم بدون درگیر شدن در مشتقات مراتب بالاتر، از مراتب بالاتر بسط تیلور به نوعی استفاده کنیم. در حالت کلی تصمیم داریم تا مقدار متغیر وابسته در نقطه‌ی بعدی را از طریق رابطه‌ی زیر بدست آوریم.

$$\begin{aligned} y_{i+1} &= y_i + \phi(x_i, y_i, h)\, h \\ \phi(x_i, y_i, h) &= a_1 k_1 + a_2 k_2 + \ldots + a_n k_n \end{aligned} \qquad (۹.۶)$$

که در آن تابع $\phi(x,y_i,h)$ را **تابع رشد (Increment Function)** می‌نامیم. این تابع از جمع n تابع k با ضرایب ثابت a بدست می‌آید. توابع k از جنس مشتق متغیر وابسته y و در نتیجه از جنس تابع $f(x,y,h)$ هستند. هر تابع k در واقع همان تابع f اما در نقطه‌ی خاصی است که باید محاسبه شود. تا حدود زیادی ترفند بکاررفته در روش رانگ‌ـ کوتا مانند روش انتگرال‌گیری گاوس است و اینجا نیز محل محاسبه‌ی تابع از مجهولات مساله است. توابع k را به شکل

$$
\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f(x_i + p_1 h, y_i + q_{11} k_1 h) \\
k_3 &= f(\underbrace{x_i + p_2 h}_{x}, \underbrace{y_i + q_{21} k_1 h + q_{22} k_2 h}_{y}) \\
&\vdots \\
k_n &= f(x_i + p_{n-1} h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \ldots + q_{n-1,n-1} k_{n-1} h)
\end{aligned} \tag{۱۰.۶}
$$

محاسبه می‌کنیم که در آن pها و qها مقادیر ثابتی هستند. بطور مثال تابع k_3 را در نظر بگیرید. همانطور که گفته شد kها از جنس مشتق متغیر وابسته و هم جنس f می‌باشند. در نتیجه عبارت $x_i + p_2 h$ در تابع k_3 در واقع مکانی روی محور x و از جنس متغیر مستقل x است که با توجه به مقادیر مشخص h و x_i، محل دقیق آن بستگی به ثابت p_2 دارد. از سوی دیگر حاصلضرب kh از جنس y می‌باشد و لذا عبارت $y_i + q_{21} k_1 h + q_{22} k_2 h$ در تابع k_3 معرف مکانی روی محور y در فاصله‌ی بین y_i و y_{i+1} است. محل دقیق آن نیز وابسطه به مقادیر ثابت q_{21} و q_{22} می‌باشد. پس در نهایت هدف محاسبه‌ی ثابت‌های p و q می‌باشد. دقت روش رانگ_کوتا با افزایش تعداد kها در تابع رشد افزایش می‌یابد. می‌توان اثبات کرد که خطای برشی سراسری برای رانگ_کوتا مرتبه‌ی n از مرتبه‌ی $\mathcal{O}(h^n)$ می‌باشد. برای شروع، ضرایب رانگ_کوتا مرتبه ۲ را محاسبه می‌کنیم.

$$
\begin{aligned}
y_{i+1} &= y_i + (a_1 k_1 + a_2 k_2) h \\
k_1 &= f(x_i, y_i) = f_i \\
k_2 &= f(x_i + p_1 h, y_i + q_{11} k_1 h) \\
&= f(x_i, y_i) + p_1 h \partial_x f + q_{11} k_1 h \partial_y f + \mathcal{O}(h^2) \\
&= f_i + p_1 h \partial_x f + q_{11} f_i h \partial_y f + \mathcal{O}(h^2)
\end{aligned}
$$

$$
y_{i+1} = y_i + \Big(a_1 f_i + a_2 (f_i + p_1 h \partial_x f + q_{11} f_i h \partial_y f) \Big) h
$$

$$
y_{i+1} = y_i + (a_1 + a_2) f_i h + a_2 (p_1 \partial_x f + q_{11} f_i \partial_y f) h^2 \tag{۱۱.۶}
$$

از آنجاییکه تابع f را در نقطه‌ی x_i و y_i داریم، در روابط (۱۱.۶) با وردش جزیی در راستاهای x و y تابع f را در نقطه‌ی جدید مورد نظر محاسبه کرده‌ایم. سپس جملات را برحسب توان‌های h مرتب نموده‌ایم تا با سری تیلور هم مرتبه خود، رابطه‌ی (۸.۶) ، قابل مقایسه باشد. و در نهایت با این مقایسه معادلات زیر را می‌توانیم بدست آوریم.

$$\begin{cases} (a_1+a_2)f_i = f_i \quad \Rightarrow a_1+a_2=1 \\ \\ a_2(p_1\partial_x f + q_{11}f_i\partial_y f) = \frac{1}{2}(\partial_x f + f_i\partial_y f) \quad \Rightarrow \begin{cases} a_2p_1 = 1/2 \\ a_2q_{11} = 1/2 \end{cases} \end{cases} \tag{۱۲.۶}$$

در روابط (۱۲.۶) تعداد معادلات بدست آمده برای ۴ مجهول a_1, a_2, p_1 و q_{11} تنها ۳ تا است. پس یک درجه‌ی آزادی در انتخاب ثابت‌ها داریم. به عبارتی تنها یک روش رانگ‌ـ کوتای مرتبه ۲ وجود ندارد و با هر انتخابی برای تک درجه آزادی پیش آمده یک روش رانگ‌ـ کوتای مرتبه‌ی ۲ حاصل می‌شود. اما معمول است مثلا $a_1 = 0$ انتخاب شود که منجر به مقادیر زیر برای ضرایب می‌شود.

$$a_1 = 0 \Rightarrow \begin{cases} a_2 = 1 \\ p_1 = q_{11} = 1/2 \end{cases} \tag{۱۳.۶}$$

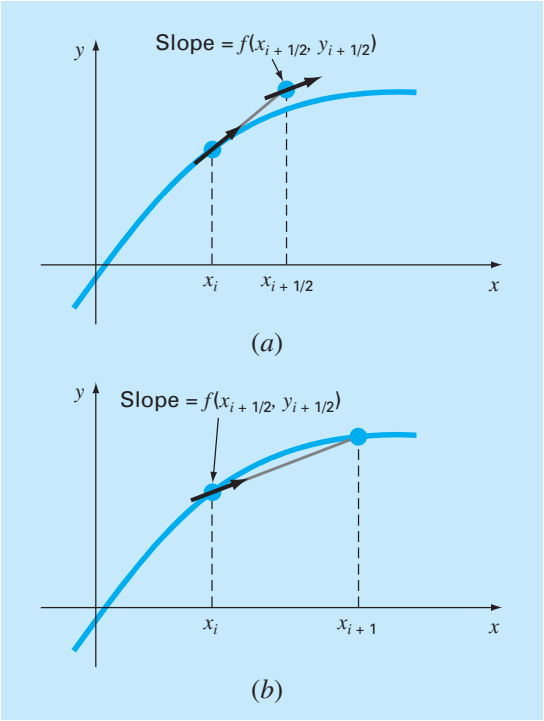
و در نهایت با این ضرایب روش رانگ‌ـ کوتای مرتبه ۲ به صورت روابط (۱۴.۶) خلاصه می‌شود. در روابط مشخص شده است که ابتدا k_1 و سپس k_2 را محاسبه می‌کنیم. با داشتن این دو تابع می‌توانیم مقدار متغیر وابسته‌ی y را در نقطه‌ی جدید x_{i+1} محاسبه کنیم. با این انتخاب این ضرایب، رانگ‌ـ کوتای مرتبه‌ی ۲ به روش **نقطه میانی (Midpoint Method)** نیز مشهور است. شکل (۲.۶) نحوه‌ی بدست آوردن نقطه‌ی y_{i+1} در روش رانگ‌ـ کوتای مرتبه‌ی ۲ (نقطه‌ی میانی) را نمایش می‌دهد.

$$\begin{cases} k_1 = f(x_i, y_i) \\ \\ k_2 = f(x_i + h/2, y_i + k_1 h/2) \end{cases} \quad \rightarrow \quad y_{i+1} = y_i + (k_2)\, h \tag{۱۴.۶}$$

مثال شماره: ۳

معادله دیفرانسیل زیر را با شرایط اولیه داده شده، با استفاده از روش رانگ‌ـ کوتای مرتبه‌ی ۲ محاسبه نمایید. مساله را برای بازه‌ی $x \in [0, 5]$ و با اندازه‌ی گام $h = 0.5$ محاسبه نمایید.

$$\begin{aligned} \ln(y') + y &= \sin(x) \\ y[0] &= 1 \end{aligned}$$

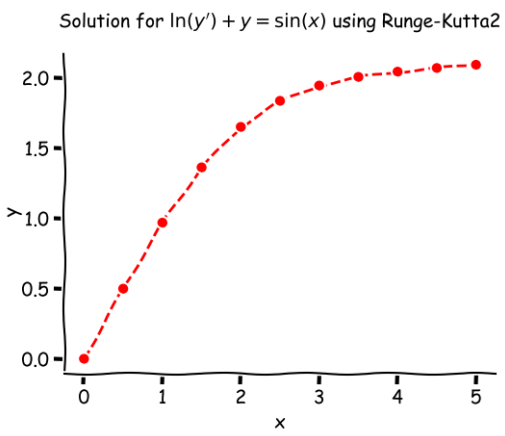


شکل ۲.۶: روش رانگ_کوتای مرتبه‌ی ۲ (نقطه‌ی میانی) برای حل معادلات دیفرانسیل معمولی.

```
import numpy as np
import matplotlib.pyplot as plt

def f(xi,yi):
    return np.exp(np.sin(xi)-yi)

def k1(xi,yi):
    return f(xi,yi)

def k2(xi,yi,h):
    x_ = xi + h/2
    y_ = yi + k1(xi,yi)*h/2
    return f(x_,y_)

def yi1(xi,yi,h):
    return yi + k2(xi,yi,h)*h

h = 0.5
```

```
x0 = 0
y0 = 0
x = [x0]
y = [y0]
for i in range(1,int(5/h)+1):
    x_ = x[-1]
    y_ = y[-1]
    x_new = x_ + h
    y_new = yi1(x_,y_,h)
    x.append(x_new)
    y.append(y_new)

fig = plt.figure()
ax = fig.add_axes((0.15, 0.15, 0.7, 0.7))
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
fig.text(0.5, 0.9, r'Solution for
    $\ln(y^{\prime}) + y = \sin(x)$ using Runge-Kutta2',
    ha='center')
plt.show()

for i in range(len(x)):
    print(f"{x[i]:0.1f} & {y[i]:0.3f} \\\\")
```

x	y
0.0	0.000
0.5	0.499
1.0	0.969
1.5	1.362
2.0	1.650
2.5	1.836
3.0	1.944
3.5	2.006
4.0	2.043
4.5	2.069
5.0	2.092



روش رانگ‌ـ کوتا مرتبه‌ی ۲ را می‌توان به مراتب بالاتر توسعه داد. یکی از روش‌هایی که استفاده از آن بسیار متداول است، روش رانگ‌ـ کوتای مرتبه‌ی ۴ است. این روش خطای برشی سراسری از مرتبه‌ی $\mathcal{O}(h^4)$ دارد. مانند روش رانگ‌ـ کوتای مرتبه‌ی ۲ باید ابتدا توابع k به ترتیب محاسبه شوند. زیرا برای محاسبه‌ی هر k به kی قبلی نیاز است. نحوه‌ی محاسبه‌ی ضرایب ثابت مساله مانند روش رانگ‌ـ کوتای مرتبه ۲ می‌باشد. البته قطعا محاسبات قدری طولانی‌تر است. بعد از محاسبه‌ی ضرایب ثابت، روابط رانگ‌ـ کوتای مرتبه‌ی ۴ برای حل معادلات دیفرانسیل معمولی به شکل روابط (۱۵.۶) خلاصه می‌شود.

$$\begin{cases} k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h) \\ k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h) \\ k_4 = f(x_i + h, y_i + k_3 h) \end{cases} \rightarrow \quad y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \quad (۱۵.۶)$$

۳.۶ دستگاه‌های معادلات دیفرانسیل

یک دستگاه n معادله دیفرانسیل معمولی مرتبه‌ی یک که در آن تعداد n متغیر $\{y_1, y_2, \ldots, y_n\}$ به هم جفت شده‌اند را در نظر بگیرید. این دستگاه در حالت کلی به شکل زیر نوشته می‌شود.

$$\begin{cases} \dfrac{dy_1}{dx} &= f_1(x, y_1, y_2, \ldots, y_n) \\ \dfrac{dy_2}{dx} &= f_2(x, y_1, y_2, \ldots, y_n) \\ \qquad \vdots & \\ \dfrac{dy_n}{dx} &= f_n(x, y_1, y_2, \ldots, y_n) \end{cases} \tag{۱۶.۶}$$

روش حل این دستگاه معادلات، چندان با حل تک معادله دیفرانسیل که در قسمت‌های قبل بررسی کردیم، متفاوت نیست. در هنگام حل هر یک از معادلات بالا مقادیر متغیر مستقل و تمام n متغیر وابسته در گام iام مشخص می‌باشند و با استفاده از آن‌ها مقادیر متغیرهای وابسته در گام $i+1$ام را محاسبه می‌کنیم. بطور مثال برای حل دو معادله دیفرانسیل جفت شده با روش اویلر روابط بازگشتی برای محاسبه‌ی y_1 و y_2 به شکل زیر خواهد شد. با قرار دادن روابط در شکل برداری نحوه بیان روش محاسباتی ساده‌تر و قابل درک‌تر می‌شود.

$$\begin{cases} y_{1,i+1} &= y_{1,i} + f_1(x_i, y_{1,i}, y_{2,i})h \\ y_{2,i+1} &= y_{2,i} + f_2(x_i, y_{1,i}, y_{2,i})h \end{cases} \Rightarrow \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{i+1} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_i + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}_i h$$

$$\Rightarrow \vec{y}_{i+1} = \vec{y}_i + \vec{f}(x_i, \vec{y}_i)h \tag{۱۷.۶}$$

این بیان برداری را می‌توان به تعداد بیشتر متغیر وابسته و معادله دیفرانسیل به شکل زیر تعمیم داد.

$$\begin{cases} \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \\ \vec{f}(x, \vec{y}) = \begin{bmatrix} f_1(x, \vec{y}) \\ f_2(x, \vec{y}) \\ \vdots \\ f_n(x, \vec{y}) \end{bmatrix} \end{cases} \Rightarrow \frac{d}{dx}\vec{y} = \vec{f}(x, \vec{y}) \tag{۱۸.۶}$$

با داشتن مقدار بردار $\vec{y}$ در یک نقطه‌ی x ، می‌توان از هر روشی (بسط تیلور، اویلر، رانگ‌ـ کوتا و ...) به شکل برداری، در نقاط بعدی آن را محاسبه نمود. به طور مثال شکل برداری روش رانگ‌ـ کوتای مرتبه‌ی ۲ با الهام از رابطه‌ی (۱۴.۶) به شکل رابطه‌ی (۱۹.۶) نوشته می‌شود.

$$\begin{cases} \vec{k}_1 = \vec{f}(x_i, \vec{y}_i) \\ \vec{k}_2 = \vec{f}(x_i + h/2, \vec{y}_i + \vec{k}_1 h/2) \end{cases} \rightarrow \vec{y}_{i+1} = \vec{y}_i + \vec{k}_2 h \qquad (۱۹.۶)$$

مثال شماره: ۴

در این مثال قصد داریم مدل دیفرانسیلی یک بیماری همه‌گیری مانند کوید‌ـ ۱۹ را بدست آوریم و سپس با روش رونگ‌ـ کوتای ۴ معادلات دیفرانسیلی حاکم را حل نماییم.
مدل بسیار ساده‌ای در بیماری‌های واگیردار با نام SIR وجود دارد. برای شروع و آشنایی این مدل، فرض کنید متغیر S معرف تعداد افراد مستعد به بیماری اما هنوز سالم، در یک جامعه باشند و متغیر I و R به ترتیب تعداد افراد مبتلا و تعداد افراد بهبودیافته باشند. این متغیرها همگی تابع زمان t هستند و با گذشت زمان تغییر می‌کنند. همچنین فرض کنید شانس بهبود یافتن هر بیمار در هر روز γ باشد. پس اگر I بیمار در جامعه وجود داشته باشد، در هر روز تعداد γI از جمعیت بیماران بهبود پیدا می‌کنند. می‌توانیم به زبان ریاضی اینگونه بنویسیم که

$$\frac{d}{dt}R(t) = \gamma I(t) \qquad (۲۰.۶)$$

اگر تعداد بیماران در جامعه‌ی N نفری I نفر باشد، یک فرد سالم که در جامعه حرکت می‌کند، در هر روز به احتمال I/N با هر فردی که مواجه می‌شود، آن فرد بیمار است. و اگر در هر تماس شانس اینکه فرد سالم مبتلا شود β باشد، پس در هر روز تعداد $\beta SI/N$ از افراد سالم کاسته می‌شود. بیان ریاضی این مطلب نیز به شکل زیر است.

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N} \qquad (۲۱.۶)$$

در نهایت هرچه از افراد سالم کم شود یعنی به بیماران اضافه شده است. و هر چه به افراد بهبود یافته اضافه شود یعنی از تعداد بیماران کاسته شده است.

$$\frac{d}{dt}I(t) = +\beta S(t)\frac{I(t)}{N} - \gamma I(t) \qquad (۲۲.۶)$$

سه معادله دیفرانسیل معمولی (۲۰.۶، ۲۱.۶، ۲۲.۶) تشکیل یک دستگاه معادلات دیفرانسیل با ۳ مجهول S, I و R می‌دهند. دقت کنید در این مدل ساده از مرگ و میر چه طبیعی و چه ناشی از این بیماری و نیز زاد و ولد در جامعه صرفنظر شده است که در گام‌های بعدی این موارد نیز باید لحاظ گردد.

طرح مساله: فرض کنید جمعیت کل جامعه مورد مطالعه، $N = 80$ میلیون نفر و ثابت است. همچنین فرض کنید 0.01% جامعه ناگهان به نحوی بیمار می‌شوند. همچنین نرخ سرایت و نرخ بهبود در این بیماری به ترتیب $\beta = 50\%$ و $\gamma = 10\%$ باشد. دستگاه معادلات مدل SIR را حل و منحنی تعداد افراد در هر سه دسته‌ی مستعد، مبتلا و بهبود یافته را بر حسب زمان بکشید.

پاسخ:

دقت کنید که جمعیت کل ثابت است. یعنی

$$I(t) + R(t) + S(t) = N = 80$$

با وجود این قید نیازی نیست هر ۳ معادله با هم حل شوند و فقط ۲ تا از معادلات مستقل هستند. لذا فقط برای تعداد مبتلا $I(t)$ و سالم $S(t)$ مساله را حل می‌کنیم، سپس از روی آن‌ها تعداد بهبود یافتگان $R(t)$ نیز بدست خواهد آمد. شکل برداری معادله دیفرانسیل را بدست می‌آوریم.

$$\vec{y} = \begin{bmatrix} S \\ I \end{bmatrix}, \qquad \vec{f} = \begin{bmatrix} -\beta SI/N \\ +\beta SI/N - \gamma I \end{bmatrix} \tag{۲۳.۶}$$

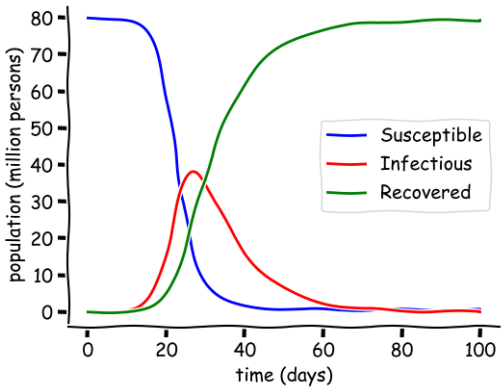
```
import numpy as np
import matplotlib.pyplot as plt

def f(xi,yi,**par):
    si,ii = yi[0],yi[1]
    beta = par["beta"]
    gamma = par["gamma"]
    N = par["N"]
    f_ = [
    - beta * si * ii / N,
    + beta * si * ii / N - gamma * ii
    ]
    return f_
```

```
def k1(xi,yi,**par):
    return f(xi,yi,**par)

def k2(xi,yi,**par):
    h = par["h"]
    x_ = xi + h/2
    y_ = [yi_ + k1_*h/2 for yi_,k1_ in zip(yi,k1(xi,yi,**par))]
    return f(x_,y_,**par)

def k3(xi,yi,**par):
    h = par["h"]
    x_ = xi + h/2
    y_ = [yi_ + k2_*h/2 for yi_,k2_ in zip(yi,k2(xi,yi,**par))]
    return f(x_,y_,**par)

def k4(xi,yi,**par):
    h = par["h"]
    x_ = xi + h
    y_ = [yi_ + k3_*h for yi_,k3_ in zip(yi,k3(xi,yi,**par))]
    return f(x_,y_,**par)

def yi1(xi,yi,**par):
    h = par["h"]
    yi1_ = [yi_ + (h/6) * (k1_+2*k2_+2*k3_+k4_)
        for yi_,k1_,k2_,k3_,k4_ in zip(yi,k1(xi,yi,**par),k2(xi,yi,**par
        k3(xi,yi,**par),k4(xi,yi,**par))
        ]
    return yi1_

N = 80
i0 = (0.01/100.0)*N
s0 = N-i0
gamma = 10.0/100.0
beta = 50.0/100.0

h = 0.1
```

```
T = 100
Nsteps = int(T/h)

yy = [[s0,i0]]
tt = [0]


for i in range(1,Nsteps):
        y_ = yy[-1]
        t_ = tt[-1]

        t_new = t_ + h
        y_new = yi1(t_,y_,h=h,gamma=gamma,beta=beta,N=N)

        tt.append(t_new)
        yy.append(y_new)

S = [i[0] for i in yy]
I = [i[1] for i in yy]
R = [N - i[0] - i[1] for i in yy]
t = tt

fig = plt.figure()
ax = fig.add_axes((0.15, 0.15, 0.7, 0.7))
plt.plot(t,S,"b-",label="Susceptible")
plt.plot(t,I,"r-",label="Infectious")
plt.plot(t,R,"g-",label="Recovered")
plt.xlabel('time (days)')
plt.ylabel('population (million persons)')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
plt.legend()

plt.show()
```

در کد اندازه‌ی گام‌ها $h = 0.1$ در نظر گرفته شده است. تابع f و همچنین توابع k_1 تا k_4 نیز بصورت یک لیست دو تایی تعریف شده‌اند. به نحوه‌ی انتقال پارامترها در توابع دقت کنید.

در پاسخ همانطور که دیده می‌شود بعد از شروع همه‌گیری بعد از ۲۵ روز پیک بیماری مشاهده می‌شود و سپس بعد از ۲ ماه تقریبا همه به این بیماری مبتلا و بهبود یافته‌اند. اما در زمان پیک نزدیک به ۵۰ درصد افراد جامعه در چند روز بیمار خواهند بود و این فشار بسیار زیادی به سیستم درمان و بیمارستان‌ها وارد می‌کند.



۴.۶ معادلات دیفرانسیل از مراتب بالاتر

در طول این فصل با روش‌های مختلفی برای حل عددی معادلات دیفرانسیل مرتبه اول آشنا شدیم. در این قسمت معادلات دیفرانسیل مراتب بالاتر را بررسی می‌کنیم. یک معادله دیفرانسیل از مرتبه‌ی n به شکل زیر است.

$$\frac{d^n y}{dx^n} = f\left(x, y, y', y^{(2)}, \ldots, y^{(n-2)}, y^{(n-1)}\right) \qquad (۲۴.۶)$$

برای حل ابتدا یک معادله دیفرانسیل مرتبه‌ی n را باید به یک دستگاه n معادله دیفرانسیل مرتبه ۱ تبدیل کنیم. برای این منظور از تغییر متغیر زیر استفاده می‌کنیم.

$$\begin{cases} y_1 &= y \\ y_2 &= y' = \dfrac{dy}{dx} \\ &\vdots \\ y_n &= y^{(n-1)} = \dfrac{d^{n-1}y}{dx^{n-1}} \end{cases} \Rightarrow \begin{cases} \dfrac{dy_1}{dx} &= y_2 \\ \dfrac{dy_2}{dx} &= y_3 \\ &\vdots \\ \dfrac{dy_{n-1}}{dx} &= y_n \\ \dfrac{dy_n}{dx} &= f(x, y_1, y_2, \ldots, y_n) \end{cases} \tag{۲۵.۶}$$

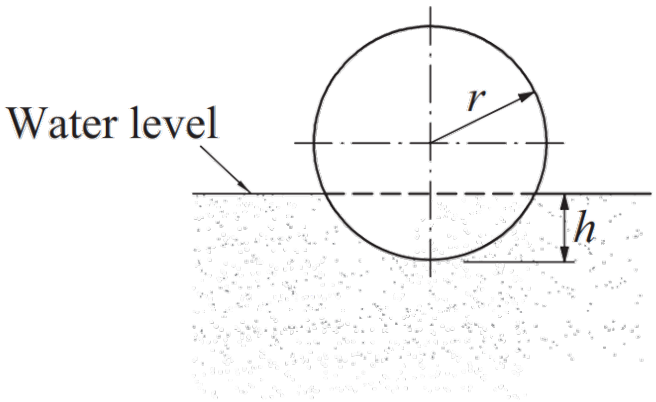
با استفاده از نمایش برداری معادله‌ی دیفرانسیل مرتبه‌ی n به شکل معادله‌ی دیفرانسیل برداری مرتبه اول زیر ساده می‌شود.

$$\begin{cases} \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \\ \vec{f}(x, \vec{y}) = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ f(x, \vec{y}) \end{bmatrix} \end{cases} \Rightarrow \frac{d}{dx}\vec{y} = \vec{f}(x, \vec{y}) \tag{۲۶.۶}$$

مثال شماره: ۵

موقعیت تعادل استوانه‌ی شناور نشان داده شده در شکل، $h = r$ است. اگر استوانه به موقعیت $h = 1.5r$ جابجا شده و آزاد شود، معادله دیفرانسیل توصیف حرکت آن می‌شود:

$$\ddot{y} = \frac{2}{\pi}\left(\tan^{-1}\frac{1-y}{\sqrt{2y-y^2}} + (1-y)\sqrt{2y-y^2}\right) \qquad (۲۷.۶)$$



که در آن $y = h/r$ است. معادله دیفرانسیل را برای y با روش رانگ‌ـ کوتای ۴ محاسبه کنید. منحنی y برحسب زمان را رسم نمایید و با استفاده از منحنی دور تناوب نوسانات استوانه را بدست آورید.

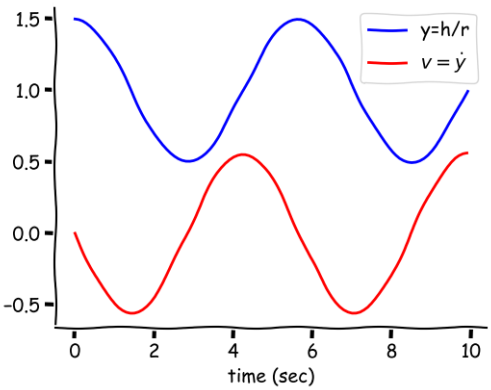
پاسخ:
معادله نوسانات استوانه معادله دیفرانسیل مرتبه دو می‌باشد و باید با روش توضیح داده شده در روابط (۲۵.۶) به دو معادله مرتبه یک تبدیل شود.

$$\begin{cases} \dot{y} = v \\ \dot{v} = \dfrac{2}{\pi}\left(\tan^{-1}\dfrac{1-y}{\sqrt{2y-y^2}} + (1-y)\sqrt{2y-y^2}\right) \end{cases}$$

```
import numpy as np
import matplotlib.pyplot as plt

def f(xi,yi,**par):
    y,v = yi[0],yi[1]
    f_ = [
        v,
        2/np.pi * (np.arctan((1-y)/(np.sqrt(2*y-y**2)))
        +(1-y)*np.sqrt(2*y-y**2))
```

```
        ]
    return f_

def k1(xi,yi,**par):
    return f(xi,yi,**par)

def k2(xi,yi,**par):
    h = par["h"]
    x_ = xi + h/2
    y_ = [yi_ + k1_*h/2 for yi_,k1_ in zip(yi,k1(xi,yi,**par))]
    return f(x_,y_,**par)

def k3(xi,yi,**par):
    h = par["h"]
    x_ = xi + h/2
    y_ = [yi_ + k2_*h/2 for yi_,k2_ in zip(yi,k2(xi,yi,**par))]
    return f(x_,y_,**par)

def k4(xi,yi,**par):
    h = par["h"]
    x_ = xi + h
    y_ = [yi_ + k3_*h for yi_,k3_ in zip(yi,k3(xi,yi,**par))]
    return f(x_,y_,**par)

def yi1(xi,yi,**par):
    h = par["h"]
    yi1_ = [yi_ + (h/6) * (k1_+2*k2_+2*k3_+k4_)
        for yi_,k1_,k2_,k3_,k4_ in zip(yi,k1(xi,yi,**par)
        ,k2(xi,yi,**par),
        k3(xi,yi,**par),k4(xi,yi,**par))
        ]
    return yi1_

# %%

h = 0.1
T = 10
```

```
Nsteps = int(T/h)

y0,v0 = 1.5,0
yy = [[y0,v0]]
tt = [0]

for i in range(1,Nsteps):
    y_ = yy[-1]
    t_ = tt[-1]

    t_new = t_ + h
    y_new = yi1(t_,y_,h=h)

    tt.append(t_new)
    yy.append(y_new)

y = [i[0] for i in yy]
v = [i[1] for i in yy]
t = tt

fig = plt.figure()
ax = fig.add_axes((0.15, 0.15, 0.7, 0.7))
plt.plot(t,y,"b-",label="y=h/r")
plt.plot(t,v,"r-",label=r"$v=\dot{y}$")
plt.xlabel('time (sec)')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
plt.legend()

plt.show()
fig.savefig('C606_rungekutta6.png')
```



با توجه به شکل بدست آماده، دور تناوب $T = 5.5(sec)$ می‌باشد.

تمرین شماره: ۱

الف - روابط بازگشتی تیلور مرتبه $n = 3$ را بدست آورید.
ب - سپس مثال (۱) را با روش تیلور مرتبه $n = 3$ مجددا حل نمایید. خطای نسبی دقیق محاسبات خود را بدست آورید و نتایج را در جدولی مانند مثال (۱) گزارش نمایید.

تمرین شماره: ۲

معادله دیفرانسیل زیر را با روش رانگ‌ـ کوتای مرتبه ۴ و اندازه گام $h = 0.5$ حل نمایید و مقدار تابع را در $x = 2$ بدست آورید.

$$y' = \sin(y), \qquad y(0) = 1$$

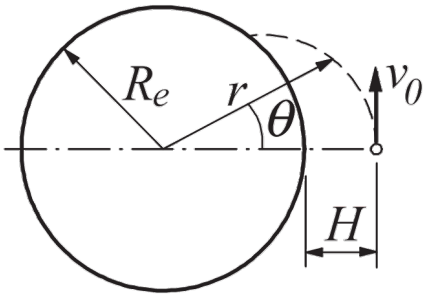
تمرین شماره: ۳

یک فضاپیما در ارتفاع $H = 772\ (km)$ از سطح دریا با سرعت $v_0 = 6700\ (m/s)$ در جهت نشان داده شده پرتاب می‌شود. معادلات دیفرانسیل که حرکت فضاپیما را توصیف می‌کنند عبارتند از:

$$\ddot{r} = r\dot{\theta}^2 - \frac{GM_e}{r^2}, \qquad \ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r} \tag{۲۸.۶}$$

که در آن r و θ مختصات قطبی فضاپیما می‌باشد. با استفاده از روش رانگ‌ـ کوتا مرتبه ۴، مسیر حرکت فضاپیما را از لحظه‌ی شروع حرکت تا لحظه‌ی برخورد با سطح زمین بدست آورید. مشخص کنید فضاپیما چه مدت طول می‌کشد و با چه زاویه‌ای با زمین برخورد می‌کند. مقادیر ثابت مورد نیاز نیز عبارتند:

G	$= 6.672 \times 10^{-11}\ (m^3 kg^{-1} s^{-2})$	universal gravitational constant
M_e	$= 5.9742 \times 10^{24}\ (kg)$	mass of the earth
R_e	$= 6378.14\ (km)$	radius of the earth at sea level



تمرین شماره: ۴

معادله دیفرانسیل زیر و شرایط اولیه داده شده را در نظر بگیرید. این معادله دارای پاسخ تحلیلی است و در ادامه معادله پاسخ تحلیلی آن نیز آورده شده است. با داشتن جواب تحلیلی مساله، می‌توانید خطاهای دقیق را محاسبه نمایید.

$$\begin{cases} y' - y &= 2(1-x) \\ y(0) &= 3 \end{cases}$$

$$y_{exact} = 2x + 3\exp(x)$$

این معادله را با روش‌های اویلر، رانگ‌ـ کوتای ۲ و رانگ‌ـ کوتای ۴ در فاصله‌ی $x = 0$ تا $x = 1$ حل نمایید. هر ۳ روش را با ۸ اندازه‌ی گام زیر تکرار کنید و خطای دقیق پاسخ خود را در نقطه‌ی $x = 1$ محاسبه نمایید. حال منحنی خطای دقیق در نقطه‌ی $x = 1$ را برحسب h را رسم کنید. مقیاس محورهای افقی و عمودی را لگاریتمی بگیرید. با توجه به منحنی که رسم کرده‌اید، نرخ همگرایی هر سه روش را بدست آورید.

$$h = \begin{Bmatrix} 0.20000 \\ 0.10000 \\ 0.05000 \\ 0.02500 \\ 0.01250 \\ 0.00625 \\ 0.00312 \\ 0.00156 \end{Bmatrix}$$

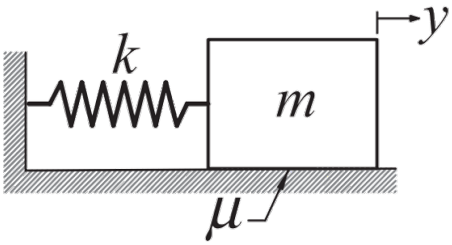
تمرین شماره: ۵

سیستم جرم و فنر را در نظر بگیرید که اصطکاک خشک بین بلوک و سطح افقی وجود دارد. نیروی اصطکاک دارای مقدار ثابت $\mu\, mg$ (μ ضریب اصطکاک است) می‌باشد و همیشه با جهت حرکت مخالف است. معادله دیفرانسیل برای حرکت بلوک را می‌توان به صورت زیر بیان کرد:

$$\ddot{y} \;=\; -\frac{k}{m}y - \mu g \frac{\dot{y}}{|\dot{y}|}$$

که در آن y از موقعیتی که فنر کشیده نشده است اندازه‌گیری می‌شود. اگر بلوک در $y = y_0$ رها شود ، با حل معادله‌ی دیفرانسیل به روش رانگ‌ـ کوتای مرتبه ۲ نشان دهید که مقدار پیک مثبت بعدی $y = y_0 - 4\mu mg/k$ است (این رابطه را می توان از روش‌های

تحلیلی بدست آورد). برای ثابت‌ها و پارامترهای مساله از مقادیر $k = 3000\,N/m$ ، $m = 6\,kg$ ، $\mu = 0.5$ ، $g = 9.80665\,m/s^2$ و $y_0 = 0.1\,m$ استفاده کنید.

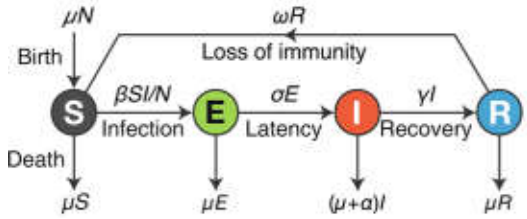


پروژه

برای اینکه تعداد معادلات دیفرانسیل بیشتری را در یک دستگاه تمرین نماییم، در این تمرین، تصمیم داریم مثال ۴ را کمی واقعی‌تر نموده و اثرات مرگ و میر، زاد و ولد، کاهش ایمنی و دوران نهفتگی بیماری را نیز در نظر بگیریم. به این منظور جامعه مورد مطالعه را به ۴ گروه زیر تقسیم می‌کنیم.

- مستعد بیماری (S)
- افراد در معرض بیماری (E)
- بیماران با عوارض مشهود (I)
- بیماران بهبود یافته (R)

که تعداد جمعیت هر گروه نوشته شده است. نمودار زیر تغییرات جمعیت هر گروه را نشان می‌دهد.



در این نمودار فرض می‌کنیم، جمعیت جامعه برابر N است. با نرخ μ از افراد تمام گروه‌های S, E, I, R در این جامعه کاسته (مرگ) و با همین نرخ به افراد مستعد (S) افزوده می‌شود (تولد). بطور مثال اگر امید به زندگی در جامعه 76 سال باشد، آنگاه $1/\mu = 76(\text{years})$ می‌شود.
فرض کنید متوسط تعداد تماس ها برای هر فرد در جامعه، در هر لحظه ضربدر احتمال انتقال بیماری در تماس بین یک فرد مستعد و یک فرد آلوده و عفونی β باشد. احتمال آنکه در هر تماس یکی از دو نفر مستعد و دیگری آلوده باشد $(S/N) * (I/N)$ است. پس بطور متوسط احتمال آنکه یک فرد سالم و مستعد بیماری، در هر لحظه در جامعه آلوده شود $\beta SI/N^2$ است و با ضرب این مقدار متوسط در تعداد کل اعضای جامعه، در هر لحظه میزان کاهش افراد سالم و افزایش افراد در فاز نهفتگی بیماری بدست می‌آید. بطور مثال در جامعه‌ی مد نظر فرض کنید $\beta = 0.50/\text{day}$ است.

در بیماری مورد نظر، دوره‌ی نهفتگی بیماری در افراد 7 روز است که هنوز بیماری عوارض خود را نشان نداده است. هر روز به تعداد σE از افراد در شُرُف بیماری کاسته و در این افراد عوارض بیماری، خود را نمایان می‌کند. با توجه به دوره‌ی نهفتگی این بیماری، $1/\sigma = 7(\text{days})$ است. فرض کنید دوره‌ی درمان این بیماری 14 روز است و از تعداد بیماران به اندازه‌ی γI کاسته و به تعداد افراد بهبود یافته اضافه می‌شود که در آن $1/\gamma = 14(\text{days})$ است. اگر نرخ فوت ناشی از بیماری را در نظر بگیریم و فرض کنیم که هر بیمار با احتمال δ بدلیل بیماری فوت می‌کند، مقدار γ باید بشکل $\gamma = (1-\delta)/14(\text{day})$ اصلاح شود و تعداد αI از بیماران بدلیل فوت ناشی از بیماری کاسته می‌شود که در آن $\alpha = \delta/14(\text{day})$ است.

در نهایت افرادی که بهبود می‌یابند بعد از 1 سال ایمنی خود را از دست می‌دهند و سال آینده مجدد تبدیل به یک فرد سالم اما مستعد به بیمار شدن می‌شوند. به عبارتی از جمعیت افراد بهبود یافته علاوه بر کاهش بدلیل فوت طبیعی μR تعداد ωR نیز بدلیل رفع مصونیت کاسته می‌شود و به افراد مستعد اضافه می‌شود. که با توجه به دوره‌ی مصونیت 1 ساله در آن $1/\omega = 365(\text{days})$ است.

اگر در این جامعه واکسیناسیون را نیز در نظر بگیریم و فرض کنیم در هر لحظه ν برابر افراد مستعد به بیماری واکسینه می‌شوند و بدون آنکه بیمار شوند مصونیت پیدا می‌کنند، در هر لحظه تعداد νS فرد از افراد مستعد کاسته و به تعداد افرادی که مصونیت دارند یعنی R اضافه می‌شود. با تجمیع همه موارد برای نرخ تغییرات جمعیت گروه‌های مختلف، به زبان ریاضی می‌توانیم دستگاه معادلات دیفرانسیل زیر را بنویسیم و از حل این دستگاه جمعیت هر دسته را در هر لحظه بدست آوریم.

$$\begin{aligned}
\frac{dS}{dt} &= \underbrace{\mu N}_{\text{تولد}} - \underbrace{\beta IS/N}_{\text{عفونت}} + \underbrace{\omega R}_{\text{رفع مصونیت}} - \underbrace{\mu S}_{\text{مرگ}} - \underbrace{\nu S}_{\text{واکسیناسیون}} \\
\frac{dE}{dt} &= -\underbrace{\sigma E}_{\text{نهفتگی}} - \underbrace{\mu E}_{\text{مرگ}} + \underbrace{\beta IS/N}_{\text{عفونت}} \\
\frac{dI}{dt} &= \underbrace{\sigma E}_{\text{نهفتگی}} - \underbrace{\gamma I}_{\text{بهبود}} - \underbrace{(\mu+\alpha)I}_{\text{مرگ}} \\
\frac{dR}{dt} &= \underbrace{\gamma I}_{\text{بهبود}} - \underbrace{\omega R}_{\text{رفع مصونیت}} - \underbrace{\mu R}_{\text{مرگ}} + \underbrace{\nu S}_{\text{واکسیناسیون}}
\end{aligned} \tag{۲۹.۶}$$

طرح مساله:

فرض کنید در ابتدا فقط 0.1 درصد از جامعه در معرض بیماری قرار بگیرند. یعنی $E(0) = 0.001\,N$. تعداد افراد در جامعه را نیز $N = 100$ در نظر بگیرید تا تمام پاسخ‌ها برحسب درصد محاسبه شوند. حال موارد زیر را محاسبه نمایید. برای محاسبه‌ی موارد، از هر روش تدریس شده در درس محاسبات عددی می‌توانید استفاده کنید و با هر پارامتر محاسباتی دلخواه می‌توانید محاسبات خود را انجام دهید تا نتایج را با دقت ۲ رقم معنی‌دار بدست آورید.

الف - نرخ واکسیناسیون و نرخ فوت ناشی از بیماری را صفر گرفته و مجموع جمعیت چهار دسته‌ی S, E, I, R را بعد از یک‌سال محاسبه نمایید. بدیهی است که پاسخ شما باید عدد ۱۰۰ شود. اما اگر نرخ فوت ناشی از بیماری صفر نباشد و شانس درمان یک بیمار را ۹۸ درصد بگیرید، در نتیجه $\delta = 0.02$ می‌شود. حال مجدد مجموع جمعیت چهار دسته‌ی S, E, I, R را بعد از گذشت یک‌سال محاسبه نمایید و محاسبه کنید چند درصد جامعه بدلیل بیماری فوت کرده است.

ب - اینک فرض کنید طبق برنامه‌ای یکنواخت و منظم تا آخر سال (۳۶۵ روز) همه افراد مستعد جامعه واکسینه شوند. یعنی $\nu = 1/365(\text{day})$ باشد. مجدد مجموع جمعیت چهار دسته‌ی S, E, I, R را بعد از گذشت یک‌سال مجدد محاسبه نمایید و نشان دهید با فرض واکسیناسیون چند درصد جامعه بدلیل بیماری فوت کرده‌اند.

ج - برای اجرای طرح واکسیناسیون مورد نظر به چند جفت دوز واکسن نیاز می‌باشد.

د - احتمال آنکه بعد از یک سال، یک فرد از جامعه بی آنکه واکسن بزند، بیمار نشود، چقدر است؟[a]

[a] برای طرح این سوال از مقاله زیر استفاده شده است. برای مطالعه‌ی بیشتر و در صورت علاقه می‌توانید به آن مراجعه نمایید.
Bjørnstad et al. The seirs model for infectious disease dynamics. Nature Methods 17, 555–558(2020)

پروژه

در این تمرین ما مسئله سه جسم را در نظر می گیریم که برای آن دو جرم m_1 و m_2 بسیار بزرگتر از m_3 هستند به طوری که $m_1, m_2 >> m_3$ علاوه بر این، فرض می شود که ما در یک چارچوب مرجع قرار داریم به طوری که دو جرم m_1 و m_2 به ترتیب در نقاط $(0,0)$ و $(-1,0)$ قرار دارند. بنابراین ما حرکت m_3 را تحت کشش گرانشی m_1 و m_2 در نظر می گیریم. این به عنوان مثال می‌تواند یک سیستم موشک، زمین و ماه را مدل کند. معادلات حاکم به شکل زیر خواهند شد.

$$
\begin{aligned}
\dot{x} &= u \\
\dot{u} &= 2v + x - \frac{\mu(1+x+\mu)}{\left(y^2+z^2+(-1+x+\mu)^2\right)^{3/2}} - \frac{(1-\mu)(x+\mu)}{\left(y^2+z^2+(x+\mu)^2\right)^{3/2}} \\
\dot{y} &= v \\
\dot{v} &= -2u + y - \frac{\mu y}{\left(y^2+z^2+(-1+x+\mu)^2\right)^{3/2}} - \frac{(1-\mu)y}{\left(y^2+z^2+(x+\mu)^2\right)^{3/2}} \\
\dot{z} &= w \\
\dot{w} &= -\frac{\mu z}{\left(y^2+z^2+(-1+x+\mu)^2\right)^{3/2}} - \frac{(1-\mu)z}{\left(y^2+z^2+(x+\mu)^2\right)^{3/2}}
\end{aligned}
$$

که در آن موقعیت جرم m_3 با بردار $\vec{r} = (x, y, z)$ و سرعت آن با بردار $\vec{v} = (u, v, w)$ داده می‌شود. پارامتر $0 < \mu < 1$ نسبت جرم ۲ به جرم ۱ است، یعنی $\mu = m_2/m_1$. برای سادگی می‌توانید مساله را محدود و در صفحه $x - y$ حل کنید. در این حالت دینامیک z و w را از مساله حذف کنید و تنها چهارمعادله‌ی اول را برای $z = 0$ حل نمایید. برای این پروژه در نظر بگیرید که $\mu = 0.1$ باشد.

الف - 5 نقطه‌ی بحرانی (لاگرانژ) این سیستم را پیدا کنید. تعریف نقاط لاگرانژ را می‌توانید در لینک زیر مطالعه کنید. از کاربردهای این نقاط محل قرار گرفتن تلسکوپ‌های فضایی، از جمله تلسکوپ جیمز وب می‌باشد.

https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/

ب - با نشان دادن حساسیت به شرایط اولیه نشان دهید که سیستم آشوبناک (Chaotic) است، یعنی جدایی دو شرایط اولیه تقریباً یکسان را در زمان اندازه گیری کنید.

پیوست آ

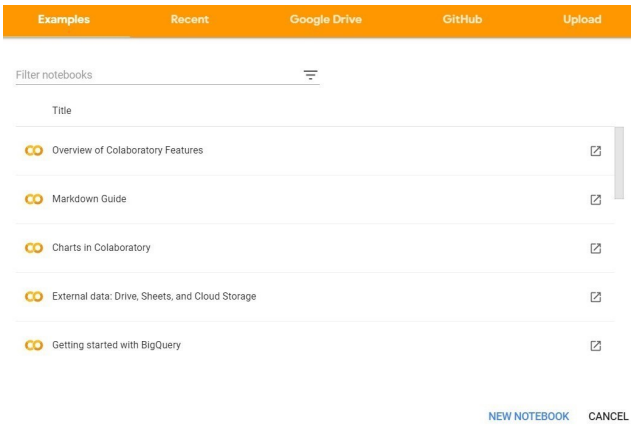
آشنایی با پایتون

برای درس روش‌های محاسبات عددی، این بسیار مورد نیاز است که بعد از یادگیری هر روش، آن روش در یک زبان برنامه نویسی تمرین شود و روش مورد تحلیل قرار گیرد. زبان‌های برنامه نویسی معمولا چنان دارای جزییات و نکات ریز و درشت می‌باشند که موجب می‌شوند دانشجو بیشتر از آنکه درگیر استفاده‌ی از آنها باشد، درگیر یادگیری آن‌ها باشد و از محور اصلی که همانا آموزش روش‌های محاسباتی است غافل گردد. به این منظور معمولا استفاده از زبان‌هایی مانند پایتون و نرم‌افزارهایی مانند متلب توصیه می‌شوند که امکانات بسیار زیادی برای عملیات محاسباتی در اختیار کاربر قرار می‌دهند. از آنجاییکه نرم‌افزار متلب بسیار حجیم می‌باشد و استفاده از آن نیازمند سخت‌افزار نسبتا قوی می‌باشد که در مرحله‌ی آموزش به آن چندان نیاز نیست و امکان استفاده از آن در اختیار همه‌ی دانشجویان نیست، لذا ما در این درس از زبان پایتون استفاده می‌کنیم که بسیار سبک، ساده و کاملا در دسترس برای همه دانشجویان می‌باشد استفاده می‌کنیم. نکته جالب‌تری که در استفاده از زبان پایتون وجود، ویرایشگرها ، مفسرهای (Interpreters) آنلاین بسیاری برای این زبان موجود است. و این دانشجو را از داشتن حتی یک کامپیوتر در صورت وجود محدودیت بی‌نیاز می‌کند و دانشجو قادر خواهد بود با اتصال به اینترنت، در داخل گوشی موبایل خود اقدام به نوشتن کدهای ساده‌ی پایتون نماید. یکی از محیط‌های بسیار خوب که تنها با داشتن یک حساب گوگل در دسترس قرار دارد سامانه‌ی (colab.research.google.com) Google Colab است. در ادامه با اجرای یک مثال چند خطی ساده با امکانات این سامانه آشنا می‌شوید.

آ.۱ شروع کار با گوگل کولب

بعد از اطمینان از داشتن حساب کاربری گوگل، وب.بروزر خود که بهتر است گوگل کروم (Google Chrome) باشد، را باز و به آدرس colab.research.google.com بروید. در صفحه‌ی مورد نظر روی NEW NOTEBOOK کلیک نموده و یک نوت‌بوک ایجاد نمایید.

در نوت‌بوک ایجاد شده، روی دکمه‌ی Connect کلیک نمایید تا برای شما RAM و CPU اختصاص یابد. سپس وارد سلول اول شده (که در شکل نشان داده شده است.) و اولین دستور پایتون



شکل آ.۱: صفحه‌ی شروع در گوگل کولب که حاوی لینک نوتبوک جدید می‌باشد.

شکل آ.۲: محیط کولب نوتبوک که در آن RAM و DISK اختصاص پیدا کرده است و در Cell شماره ۱ دستور پرینت نوشته و اجرا شده است.

خود که دستور زیر باشد را بنویسید.

مثال شماره: ۱

```
print("Hello, World!")
```

برای اجرا می‌توانید از دکمه‌ی سمت چپ هر سلول (▶) استفاده نمایید. بعد از اجرای هر سلول، نتیجه در پایین همان سلول ظاهر خواهد شد.

آ.۲ متغیرها

در زبان پایتون به مانند دیگر زبان‌های برنامه نویسی داده‌ها درون متغیرها نگهداری می‌شوند و عملگرها روی داده‌ها و یا متغیرهایی که در آن‌ها داده وجود دارد عمل می‌کنند. این عملیات عموما مانند دیگر

زبان‌ها می‌باشند. اما شاید در چند مورد متفاوت باشند که در مثال زیر با مواردی از آنها آشنا می‌شوید.

مثال شماره: ۲

```
# comment ....

a,b = 35,10

c = "Exponent: b**3"
print(f"c={b**3}")

c="Modulus (remainder): a%b"
print(f"c={a%b}")

c="Floor Division (quotient): a//b"
print(f"c={a//b}")
```

در مثال بالا دقت نمایید که متغیر c یک متغیر رشته‌ای یا string است و یک دنباله از کرکترها را در خود جا می‌دهد. رشته‌ها عبارت‌هایی هستند که در بین دو علامت (" ") قرار می‌گیرند. نوعی از رشته‌ها که به fstring شهرت دارند و با قرار داد حرف f در ابتدای اولین (") مشخص می‌شوند، می‌توانند درون خود متغیرهای دیگر را بین یک جفت علامت ({ }) بپذیرند. همانطور که دیده می‌شود، در مثال بالا توابع print رشته‌های fstring را فراخوانده‌اند. همچنین دقت بفرمایید که مفسر پایتون عبارت comment که با علامت (#) در ابتدای خط مشخص شده است را تفسیر و اجرا نمی‌کند و آن را توضیحی در برنامه در نظر می‌گیرد.

آ.۳ ساختار داده‌ها

در پایتون داده‌ها دارای ساختارهای مختلفی هستند و برنامه‌نویس نیز می‌تواند در قالب اشیاء یا Ob-jects، ساختارهای جدید و دلخواه خود را ایجاد نماید. در پیش‌فرض علاوه بر نرده‌ها که تک عدد (number) و یا تک رشته (string) می‌باشند، ۳ ساختار مهم دیگر در پایتون وجود دارد.

- لیست‌ها (Lists)
- چندتایی‌ها (Tuples)

— واژه‌نامه‌ها (Dictionaries)

لیست‌ها دنباله‌ای از داده‌ها هستند که با نظمی مشخص در کنار هم قرار گرفته‌اند و از طریق اندیس و یا درون یک حلقه می‌توان به تک تک آن داده‌ها دسترسی داشت. در مثال زیر با این ساختار آشنا می‌شوید:

مثال شماره: ۳

```
bikes = ['trek', 'redline', 'giant']
first_bike = bikes[0]
last_bike = bikes[-1]
# Looping through a list
for bike in bikes:
    print(bike)
# Adding items to a list
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
# Making numerical lists
squares = []
for x in range(1, 11):
    squares.append(x**2)
# List comprehensions
squares = [x**2 for x in range(1, 11)]
# Slicing a list
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
# Copying a list
copy_of_bikes = bikes[:]
```

چندتایی‌ها یا tuples نوعی لیست هستند که داده‌های آنها بعد از تعریف تغییر نمی‌کند. در مثال زیر نحوه‌ی تعریف یک چندتایی را مشاهده می‌کنید.

مثال شماره: ۴

```
# Making a tuple
dimensions = (1920,1080)
print(dimensions[0])
```

واژه‌نامه‌ها یا Dictionaries نوع دیگری از ساختارهای داده‌ها هستند که در آنها داده‌ها بصورت جفت کلید و مقدار (key-value pair) نگهداری می‌شوند. که با استفاده از کلید مربوطه، مقدار مد نظر فراخوانده می‌شود. برای آشنایی با ایجاد و استفاده از این ساختار به مثال زیر توجه کنید:

مثال شماره: ۵

```
# A simple dictionary
alien = {'color': 'green', 'points': 5}
# Accessing a value
print(f"The alien's color is {alien['color']}")
# Adding a new key-value pair
alien['x_position'] = 0
# Looping through all key-value pairs
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(f"{name} loves {number}")
# Looping through all keys
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(f"{name} loves a number")
#Looping through all the values
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(f"{number} is a favorite")
```

آ.۴ عبارت‌های شرطی

همانطور که از نام آن برمی‌آید، عبارت‌های شرطی در مواردی استفاده می‌شوند که وجود یک شرط و شرایط را بررسی کنند. به کمک این عبارتها می‌توان مسیرهای مختلفی از اجرای کد، برای شرایط مختلف را ایجاد نمود. در مثال زیر با این عبارتها و نحوه‌ی استفاده از آن‌ها آشنا می‌شوید.

مثال شماره: ۶

```
# Conditional tests
# equals x == 42
# not equal x != 42
# greater than x > 42
# or equal to x >= 42
# less than x < 42
# or equal to x <= 42
# Conditional test with lists
'trek' in bikes
'surly' not in bikes
# Assigning boolean values
game_active = True
can_edit = False
# A simple if test
if age >= 18:
    print("This line is the first line of block.")
    print("You can vote!")
    print("This line is the last line of block.")
# If-elif-else statements
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

در مثال بالا به این نکته توجه کنید که عبارتهای شرطی بعد از if و elif و else به یک علامت

: ختم می‌شوند. فرمانهایی که بعد از علامت : در خطوط بعدی با یک تورفتگی همراه هستند، در صورت ارضاء شرط مذکور، اجرا خواهند شد. به این فرمان‌هایی که با هم تورفتگی دارند یک بلوک (block) گفته می‌شوند.

آ.۵ حلقه‌ها

حلقه‌ها هم مانند عبارتهای شرطی دارای بلوک هستند. و کدهای درون این بلوک‌ها پیوسته تکرار می‌شوند. در پایتون، دو نوع حلقه‌ی while و for معمولا مورد استفاده قرار می‌گیرد. در حلقه‌ی while شرطی در ابتدا آورده می‌شود و کدهای درون بلوک تا زمانی تکرار می‌شوند که این شرط ارضاء می‌شود. و در حلقه‌ی for یک لیست در ابتدا معرفی می‌شود کدهای درون بلوک بازاء تمام اعضاء لیست تکرار می‌گردد. در مثال زیر با مفهوم حلقه‌های مختلف بیشتر آشنا می‌شوید.

مثال شماره: ۷

```
# A simple while loop
current_value = 1
while current_value <= 5:
    print(current_value)
    current_value += 1
# Letting the user choose when to quit
msg = ' '
while msg != "quit":
    msg = input("What's your message? ")
    print(msg)

# A simple for loop
finishers = ['sam', 'bob', 'ada', 'bea']
for x in finishers:
    print(x)
# A for loop using range
for x in range(1, 11):
    print(x)
```

در مثال بالا تابع input مقدار متغیر msg را از کاربر می‌گیرد. در مثال زیر مشاهده می‌کنید که

از حلقه‌ها و عبارت‌های شرطی چگونه می‌توان لیست‌های جدید درست کرد.

مثال شماره: ۸

```
# using loop and condition in index of list
L1 = list(range(1,10))
L2 = [2,30,8]
L3 = [x for x in L1 if x not in L2]
print(L3)
```

آ.۶ توابع

توابع یا Functions بلوکی از کدها هستند که طراحی می‌شوند تا عملیات خاصی را انجام دهند. توابع معمولا اطلاعاتی را به عنوان ورودی یا argument دریافت می‌کنند و از این اطلاعات جهت انجام عملیات استفاده می‌کنند. با چند نمونه از تعریف توابع و استفاده از آنها در مثال زیر آشنا می‌شوید.

مثال شماره: ۹

```
# A simple function
def greet_user():
    """Display a simple greeting."""
    print("Hello!")
greet_user()

# Passing an argument
def greet_user(username):
    """Display a personalized greeting."""
    print(f"Hello, {username}!")
greet_user('jesse')

# Default values for parameters
def make_pizza(topping='bacon'):
    """Make a single-topping pizza."""
```

```
    print(f"Have a {topping} pizza!")
make_pizza()
make_pizza('pepperoni')

# Returning a value
def add_numbers(x, y):
    """Add two numbers and return the sum."""
    return x + y
sum = add_numbers(3, 5)
print(sum)
```

توابع در پایتون دارای خاصیت جالبی هستند و قادرند در داخل بدنه‌ی خود، خود را صدا کرده و به این طریق در واقع نوعی حلقه ایجاد کنند. این نوع توابع را توابع بازگشتی (Recursive Functions) می‌نامند. برای آشنایی با این توابع به مثال [۱] زیر که در آن فاکتوریل یک عدد را محاسبه می‌کنیم، و در مثال بعدی آن که مکان‌ها درون یک لیست پرش می‌شود، توجه فرمایید.

مثال شماره: ۱۰

```
def factorial_recursive(n):
    # Base case: 1! = 1
    if n == 1:
        return 1

    # Recursive case: n! = n * (n-1)!
    else:
        return n * factorial_recursive(n-1)

# Example
factorial_recursive(5)

#----------------------------------------
# Output
# 120
```

[۱] https://realpython.com/python-thinking-recursively

مثال شماره: ۱۱

```
houses = ["Eric's house", "Kenny's house", "Kyle's house", "Stan's hous

# Each function call represents an elf doing his work
def deliver_presents_recursively(houses):
    # Worker elf doing his work
    if len(houses) == 1:
        house = houses[0]
        print("Delivering presents to", house)

    # Manager elf doing his work
    else:
        mid = len(houses) // 2
        first_half = houses[:mid]
        second_half = houses[mid:]

        # Divides his work among two elves
        deliver_presents_recursively(first_half)
        deliver_presents_recursively(second_half)

# Example:
deliver_presents_recursively(houses)

#---------------------------------------
# Output:
# Delivering presents to Eric's house
# Delivering presents to Kenny's house
# Delivering presents to Kyle's house
# Delivering presents to Stan's house
```

مثال شماره: ۱۲

```
# Storing a function in a module
# File: pizza.py
def make_pizza(size, *toppings):
    """Make a pizza."""
    print(f"\nMaking a {size} pizza.")
    print("Toppings:")
    for topping in toppings:
        print(f"- {topping}")
```

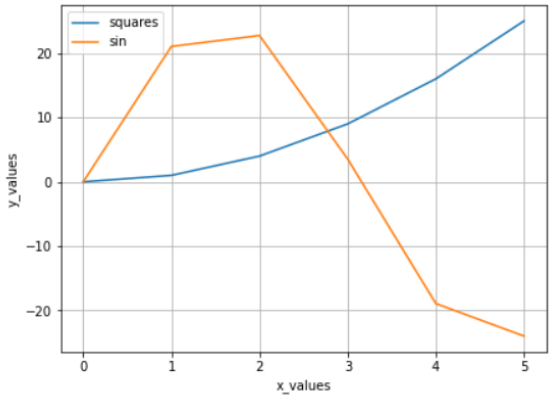
مثال شماره: ۱۳

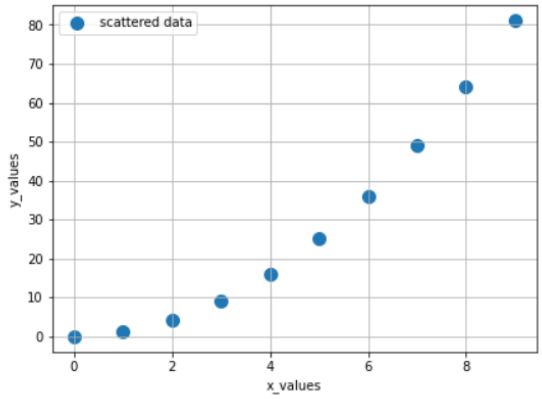
```
# Importing an entire module
# File: making_pizzas.py
# Every function in the module is available
# in the program file.
import pizza
pizza.make_pizza('medium', 'pepperoni')
```

```
pizza.make_pizza('small', 'bacon', 'pineapple')

# Importing a specific function
# Only the imported functions are available in the program file.
from pizza import make_pizza
make_pizza('medium', 'pepperoni')
make_pizza('small', 'bacon', 'pineapple')

# Giving a module an alias
import pizza as p
p.make_pizza('medium', 'pepperoni')
p.make_pizza('small', 'bacon', 'pineapple')

# Giving a function an alias
from pizza import make_pizza as mp
mp('medium', 'pepperoni')
mp('small', 'bacon', 'pineapple')

# Importing all functions from a module
# Don't do this, but recognize it when you see it in others' code.
# It can result in naming conflicts, which can cause errors.
from pizza import *
make_pizza('medium', 'pepperoni')
make_pizza('small', 'bacon', 'pineapple')
```

مثال شماره: ۱۴

```
# Making a line graph
# fig represents the entire figure, or collection of plots; ax
# represents a single plot in the figure.
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure(figsize=(15, 5))

x_values = [0, 1, 2, 3, 4, 5]
squares = [0, 1, 4, 9, 16, 25]
y_values = 25*np.sin(x_values)
ax = plt.subplot(121)
ax.plot(x_values, squares, label='squares')
ax.plot(x_values,y_values, label= 'sin')
ax.set_xlabel("x_values")
ax.set_ylabel("y_values")
ax.grid(True)
ax.legend()

# Making a scatter plot
# scatter() takes a list of x and y values; the s=10 argument
# controls the size of each point.
x_values = list(range(10))
squares = [x**2 for x in x_values]
ax = plt.subplot(122)
ax.scatter(x_values, squares, s=100,label='scattered data')
ax.set_xlabel("x_values")
ax.set_ylabel("y_values")
ax.grid(True)
ax.legend()

plt.show()
```





تمرین شماره: ۱

الف: برنامه‌ای بنویسید و در آن منحنی‌های زیر را در بازه‌ی $x \in [0, 2\pi]$ ترسیم و با هم مقایسه کنید. (در برنامه خود از حلقه و کتابخانه‌ی numpy حتما استفاده کنید.)

$$y_1 = \sin(x),$$
$$y_2 = \sum_{n=0}^{5} (-1)^{(n)} \frac{x^{(2n+1)}}{(2n+1)!}$$

ب: برنامه‌ای بنویسید و در آن اختلاف منحنی‌های قسمت الف را در همان بازه قبلی ترسیم کنید.

تمرین شماره: ۲

برنامه‌ای بنویسید که در آن تابعی نوشته شود و سری فیبوناچی از جمله‌ی امn تا جمله‌ی امm محاسبه و برگرداند (return). در ادامه، تابع مورد نظر فراخوانده شود و این سری را از جمله ۱۰۰ام تا ۱۱۰ام را پرینت نماید.

$$F_0 = 0, \quad F_1 = 1,$$
$$F_n = F_{n-1} + F_{n-2}$$

تمرین شماره: ۳

روش غربال اراتوستن (Sieve of Eratosthenes) برای یافتن اعداد اول

در این روش لیستی از اعداد کوچکتر یا مساوی عدد داده شده n ایجاد می‌کنیم و به ترتیب تمام اعدادی که عدد اول نیستند و مضربی از یک عدد کوچکتر می‌باشند را از لیست حذف می‌کنیم در الگوریتم (1) جزییات این مراحل آمده است. [a]

برنامه‌ای بنویسید که عدد صحیح n را به عنوان ورودی بگیرد و ۲ لیست به برنامه برگرداند.

۱ - لیستی از اعداد اول کوچکتر یا مساوی n.

۲ - لیستی از اعدادی که عدد اول نبوده‌اند و از لیست اولیه حذف شده‌اند. این لیست شامل شماره ترتیب حذف شدن نیز باشد.

۳ - خروجی برنامه خود را بطور مثال برای $n = 15$ پرینت نمایید.

List 1	List 2	
Prime Numbers	Order	Removed Value
2	1	4
3	2	6
5	3	8
7	4	10
11	5	12
13	6	14
	7	9
	8	15

[a] https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

Algorithm 1: SIEVE OF ERATOSTHENES

Input: An integer n
Output: A list p, consisting prime numbers less than or equal to the given integer n

```
1 p ← [2, ..., n]
2 for i ← 2 to n do
3     if i not in p then
4         continue
5     j_max ← n/i
6     for j ← 2 to j_max do
7         # Remove j from p
8         p.remove(j)
9 return p
```