فصل دوم

حل معادلات f(x) = 0

۱-۲ مقدمه

در اغلب کارهای مهندسی با حل معادله ای به شکل f(x) = 0 مواجه می شویم که در آن f یک تابع مفروض است. منظور از حل معادله f(x) = 0 یافتن مقادیری از متغیر x است که به ازای آنها مقدار تابع صفر شود. هرگاه f(α) = 0 آنگاه α را یک ریشه معادله می نامیم و یا α یک صفر تابع است.

معادلاتی مانند معادلات زیر قابل حل با روشهای تحلیلی نبوده و برای حل آنها بایستی روشهای تقریبی استفاده کرد:

$$e^{-x} - \cos x = 0 \qquad x + \cos x = 0 \qquad x^2 - (1-x)^5 = 0$$

معمولا برای تعیین ریشه ای از یک معادله با دقت مورد نظر، لازم است تقریبی از آن ریشه یا فاصله کوچکی را که حاوی آن ریشه باشد، معلوم کرد. به این منظور محدودیتهای زیر را در نظر می گیریم.

محدودیت الف: فاصله ای موجود باشد که شامل ریشه باشد.
محدودیت ب: بایستی ریشه در فاصله مورد نظر یکتا باشد.
از نظر ریاضی محدودیت الف را به صورت زیر بیان می کنیم:
۱- تابع $y = f(x)$ در فاصله $[a, b]$ پیوسته است.
۲- $f(a)$ و $f(b)$ دارای علامتهای متفاوتند، یعنی $f(a)f(b) < 0$.
در نتیجه بنا به قضیه مقدار میانی، عددی مانند α در فاصله $[a, b]$ وجود دارد، به طوری که: $f(\alpha) = 0$
با داشتن شرایط ۱ و ۲، محدودیت ب از نظر ریاضی، به صورت زیر بیان می شود:
۳- برای هر $x \in [a, b]$: $f'(x) \neq 0$

۲-۲ تعیین ریشه ها با دقت مورد نظر

با مشخص بودن فاصله ای که شامل یک ریشه معادله $f(x) = 0$ است، برای تعیین تقریبی از ریشه مورد نظر با دقت مطلوب، دنباله ای از اعداد مانند x_n می سازیم به طوری که با افزایش n، مقدار x_n به α نزدیک شود، یعنی

$$\lim_{n \to \infty} x_n = \alpha$$

بنابراین با توجه به تعریف حد، عددی مانند N وجود دارد که $x_N \approx \alpha$
تعیین Nی که تقریب بالا را بدهد، معیار توقف محاسبه x_n ها نامیده می شود. در اینجا سه معیار برای توقف ارائه می شود. در یک مساله پیچیده باید شانس استفاده از هر سه معیار را به طور مناسب در الگوریتم قرار داد.

الف) هرگاه ε عددی معلوم و مفروض باشد (مثلا $\varepsilon = 10^{-6}$)، x_n ها را تا جایی محاسبه می کنیم که $|f(x_N)| < \varepsilon$ ، یعنی به محض اینکه $|f(x_N)| < \varepsilon$ ، عملیات محاسبه x_n را متوقف می کنیم و x_n را به عنوان تقریب α می پذیریم.

ب) هرگاه $|x_N - x_{N-1}| < \varepsilon$ عملیات را متوقف نموده و x_n را به عنوان تقریب α می پذیریم.

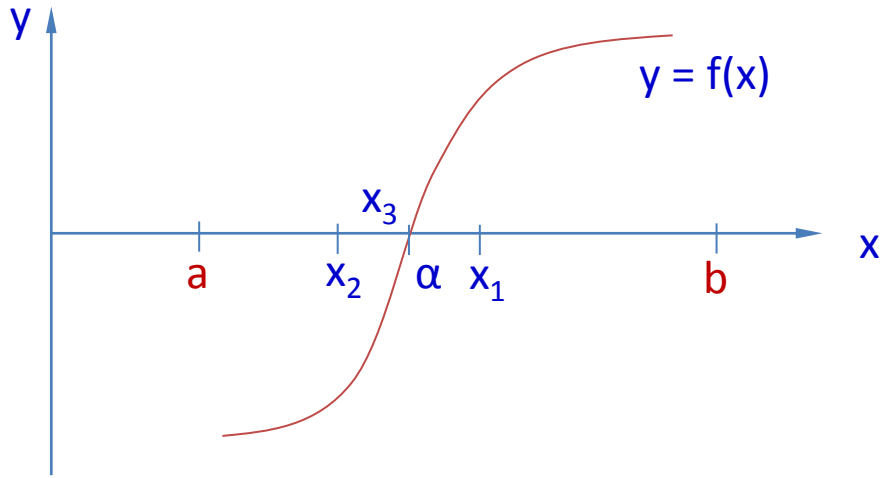
پ) هرگاه تعداد تکرار، n از یک عدد مشخص بیشتر شود عملیات را متوقف نموده و x_n را به عنوان تقریب α می پذیریم.

در ادامه چند روش عددی را برای تعیین ریشه معادله $f(x) = 0$ معرفی می کنیم. در تمام حالتها فرض می کنیم محدودیتهای الف و ب برقرارند.

۲-۳ روش های عددی حل معادله f(x) = 0

۱-۳-۲ روش دو بخشی یا روش تنصیف (Bisection)

هرگاه شرایط ۱ تا ۳ برقرار بوده و ریشه معادله f(x) = 0 باشد، در این صورت نقطه (α ,۰) بر روی نمودار تابع y = f(x) قرار دارد:



شکل ۱. روش دو بخشی

در روش دو بخشی وسط فاصله [a, b] را بعنوان اولین تقریب α در نظر می گیریم

$$x_1 = \frac{a+b}{2}$$

برای x_1 سه حالت وجود دارد:

۱- $f(x_1) = 0$، که در این صورت x_1 ریشه معادله بوده $\alpha = x_1$.

۲- $f(a)f(x_1) < 0$ یعنی ریشه بین a و x_1 است، که در این صورت در تکرار بعدی فاصله $[a, x_1]$ را برای تعیین ریشه معادله در نظر می گیریم.

۳- $f(a)f(x_1) > 0$ یعنی ریشه بین a و x_1 نیست، که در این صورت در تکرار بعدی فاصله $[x_1, b]$ را برای تعیین ریشه در نظر می گیریم.

با مقدمه بالا، الگوریتم روش دو بخشی یا تنصیف را بصورت زیر خواهیم داشت:

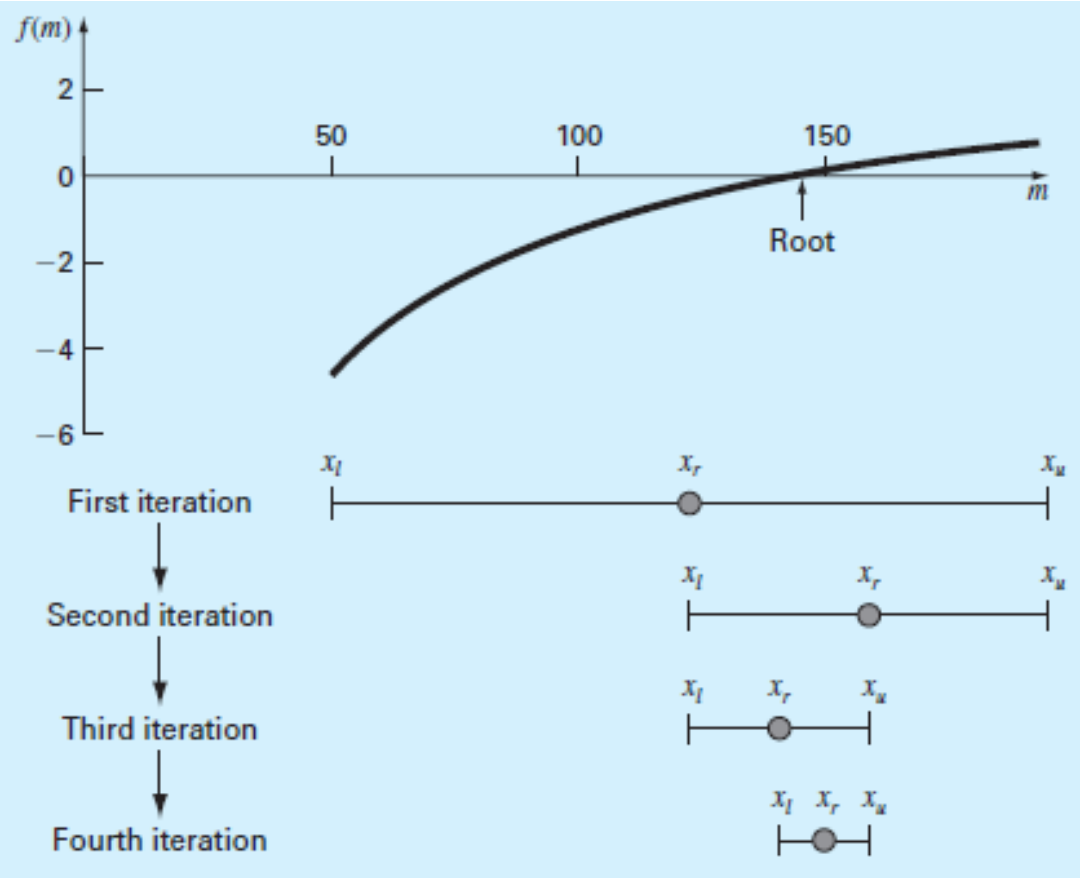
الگوریتم روش دو بخشی (Bisection)

قدم ۱. قرار دهید $x = (a + b) / 2$

قدم ۲. اگر $f(a)f(x) < 0$، آنگاه ریشه در فاصله (a, x) است، قرار دهید $b = x$ و دوباره قدم ۱ را در فاصله جدید $[a, b]$ تکرار کنید.

قدم ۳. اگر $f(a)f(x) > 0$، آنگاه ریشه در فاصله (x, b) است، قرار دهید $a = x$ و دوباره قدم ۱ را در فاصله جدید $[a, b]$ تکرار کنید.

قدم ۴. اگر $f(a)f(x) = 0$، آنگاه x ریشه است و عملیات خاتمه می یابد.



شکل ۲. توصیف نموداری روش دو بخشی

نکته ۱. از آنجا که لزومی ندارد که قدم ۴ اتفاق بیافتد، بنابراین جهت کنترل خطا و پایان الگوریتم باید معیارهای توقف یاد شده را در الگوریتم به طور مناسب به کار گرفت.

نکته ۲. روش دو بخشی همگرایی تضمین شده دارد، یعنی همواره با دقت مورد نظر به جواب خواهیم رسید.

مثال ۱. تقریبی از ریشه معادله $f(x)=3x-e^{-x}=0$ را که در فاصله $(0.25, 0.27)$ قرار دارد، با سه رقم اعشار به دست آورید، به طوری که داشته باشیم $|f(x_n)|<0.001$ که x_n تقریب ریشه در تکرار nام است.

حل: با توجه به الگوریتم یاد شده، جدول زیر را خواهیم داشت:

n	a	b	$x_n = (a+b)/2$	f(a)	f(b)	$f(x_n)$	$f(a)f(x_n)$ علامت
1	0.25	0.27	0.26	-0.0288	0.0466	0.0089	-
2	0.25	0.26	0.255	-0.0288	0.0089	-0.0099	+
3	0.255	0.26	0.2575	-0.0099	0.0089	-0.0005	+

چون $|f(x_3)| = 0.0005 < 0.001$ بنابراین x_2 را به عنوان تقریب ریشه معادله در نظر می گیریم. از اینرو هرگاه α ریشه مورد نظر باشد، با سه رقم اعشار داریم:

$$\alpha \approx 0.285$$

تذکر ۱. با توجه به نکته بیان شده در فصل اول، چون در مثال ۱ تقریب را با سه رقم اعشار خواستیم، محاسبات میانی را با چهار رقم اعشار انجام داده و در نهایت x_3 را تا سه رقم اعشار گرد نموده به عنوان تقریب α قرار داده ایم. بدیهی است این تعداد رقم اعشار برای محاسبات میانی لزوما در هر مساله مفید نخواهد بود و باید بسته به نوع مساله، خود استفاده کننده از الگوریتم، در این مورد تصمیم بگیرد.

تذکر ۲. دقت کنید که شرایط ۱ تا ۳ برای معادله مثال ۱ برقرارند، زیرا داریم:
$f(a) = f(0.25) = -0.0288$ و $f(b) = f(0.27) = +0.0466$ پس f در فاصله $(0.25, 0.27)$ دارای ریشه می باشد، همچنین داریم:

$$f'(x) = 3 + e^{-x} > 0 \qquad \text{همواره}$$

تکلیف ۱. محاسبات میانی مثال ۱ را با سه رقم اعشار انجام داده و نتیجه را با مثال ۱ مقایسه نمایید.

HW #2. Use bisection to determine the drag coefficient needed so that an 80-kg bungee jumper has a velocity of 36 m/s after 4 s of free fall. Note: The acceleration of gravity is 9.81 m/s^2. Start with initial guesses of $x_l = 0.1$ and $x_u = 0.2$ and iterate until the approximate relative error falls below 2%.

Hint: You know from your previous studies that the following analytical solution can be used to predict fall velocity as a function of time:

$$v(t) = \sqrt{\frac{g\,m}{c_d}}\tanh\left(\sqrt{\frac{g\,c_d}{m}}\,t\right) \qquad \text{(H.1)}$$

Try as you might, you cannot manipulate this equation to explicitly solve for c_d - that is, you cannot isolate the mass on the left side of the equation. An alternative way of looking at the problem involves subtracting v(t) from both sides to give a new function:

$$f(c_d) = \sqrt{\frac{g\,m}{c_d}}\tanh\left(\sqrt{\frac{g\,c_d}{m}}\,t\right) - v(t) \qquad \text{(H.2)}$$

Although Eq. (H.1) provides a mathematical representation of the interrelationship among the model variables and parameters, it cannot be solved explicitly for drag coefficient. In such cases, c_d is said to be implicit.

We require an error estimate that is not contingent on foreknowledge of the root. One way to do this is by estimating an approximate percent relative error as

$$|\varepsilon_a| = \left| \frac{x_r^{new} - x_r^{old}}{x_r^{new}} \right| \times 100\%$$

Provide the following information.

Iteration	x_l	x_u	x_r	$f(x_l)$	$f(x_u)$	$f(x_r)$	sign fl*fu	$\lvert\varepsilon_a\rvert$
1								
2								
⋮								

Plot the approximate error versus the number of iterations.

Although bisection is generally slower than other methods, the neatness of its error analysis is a positive feature that makes it attractive for certain engineering and scientific applications. Another benefit of the bisection method is that the number of iterations required to attain an absolute error can be computed *a priori* - that is, before starting the computation.

This can be seen by recognizing that before starting the technique, the absolute error is

$$E_a^0 = x_u^0 - x_l^0 = \Delta x^0$$

where the superscript designates the iteration.

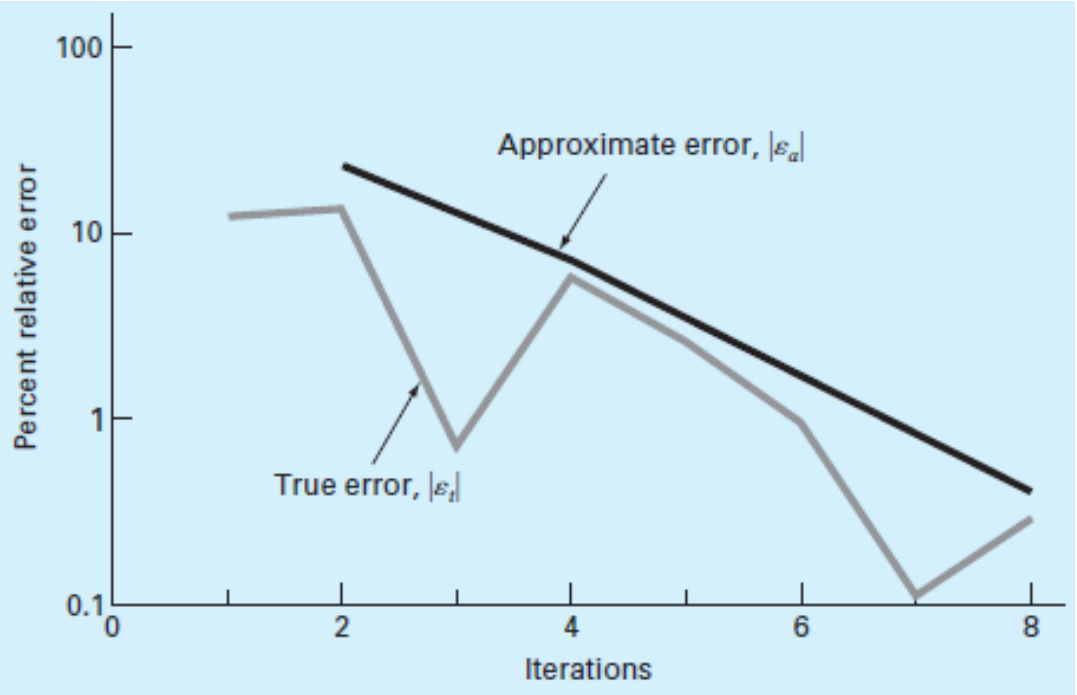
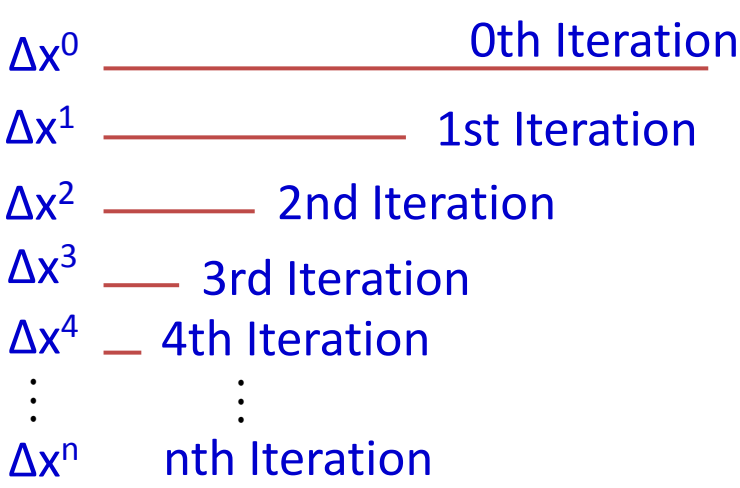


Fig. 3 Sample errors in bisection method

Hence, before starting the method we are at the "zero iteration." After the first iteration, the absolute error becomes

$$E_a^1 = \frac{\Delta x^0}{2}$$



Because each succeeding iteration halves the error, a general formula relating the error and the number of iterations *n* is

$$E_a^n = \frac{\Delta x^0}{2^n}$$

If $E_{a,d}$ is the desired error, this equation can be solved for

$$n = \frac{\log\left(\Delta x^0 / E_{a,d}\right)}{\log 2} = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right)$$

From Example 1, the absolute error can be drawn with respect to the number of iteration, as shown in Fig. 4.

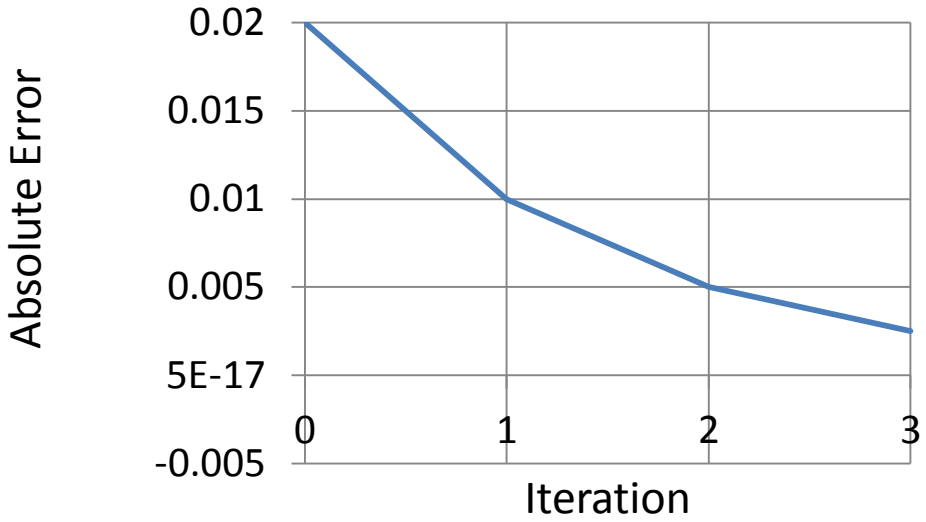


Fig. 4 Absolute error versus iteration of Example 1

MATLAB M-file: bisect

An M-file to implement bisection is displayed below. It is passed the function (func) along with lower (xl) and upper (xu) guesses. In addition, an optional stopping criterion (es) and maximum iterations (maxit) can be entered. The function first checks whether there are sufficient arguments and if the initial guesses bracket a sign change. If not, an error message is displayed and the function is terminated. It also assigns default values if maxit and es are not supplied. Then a while...break loop is employed to implement the bisection algorithm until the approximate error falls below es or the iterations exceed maxit.

```
function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
% bisect: root location zeroes
% [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):
% uses bisection method to find the root of func
```

```
% input:
% func = name of function
% xl, xu = lower and upper guesses
% es = desired relative error (default = 0.0001%)
% maxit = maximum allowable iterations (default = 50)
% p1,p2,... = additional parameters used by func
% output:
% root = real root
% fx = function value at root
% ea = approximate relative error (%)
% iter = number of iterations
if nargin<3,error('at least 3 input arguments required'),end
test = func(xl,varargin{:})*func(xu,varargin{:});
if test>0,error('no sign change'),end
if nargin<4|isempty(es), es=0.0001;end
if nargin<5|isempty(maxit), maxit=50;end
```

```
iter = 0; xr = xl; ea = 100;
while (1)
xrold = xr;
xr = (xl + xu)/2;
iter = iter + 1;
if xr ~= 0,ea = abs((xr - xrold)/xr) * 100;end
test = func(xl,varargin{:})*func(xr,varargin{:});
if test < 0
xu = xr;
elseif test > 0
xl = xr;
else
ea = 0;
end
if ea <= es | iter >= maxit,break,end
end
root = xr; fx = func(xr, varargin{:});
```

Example 2. We can employ this MATLAB function to solve the following problem. You need to determine the mass at which a bungee jumper's free-fall velocity exceeds 36 m/s after 4 s of free fall given a drag coefficient of 0.25 kg/m. The mass is assumed to be between 140 and 150 kg. Thus, you have to find the root of

(H.2): $$f(m)=\sqrt{\frac{9.81m}{0.25}}\tanh\left(\sqrt{\frac{9.81(0.25)}{m}}4\right)-36$$

The bisect function can be used to determine the root as

```
>> fm=@(m) sqrt(9.81*m/0.25)*tanh(sqrt(9.81*0.25/m)*4)-36;
>> [mass fx ea iter]=bisect(fm,40,200)
mass =
142.74
fx =
4.6089e-007
```

```
function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)
```

ea =
5.345e-005
iter =
21

Thus, a result of m = 142.74 kg is obtained after 21 iterations with an approximate relative error of ε_a = 0.00005345%, and a function value close to zero.

HW #3. Repeat Example 2 step by step and plot the approximate error versus the number of iterations.

A function stored in an M-file can be plotted with a single command, as shown below.

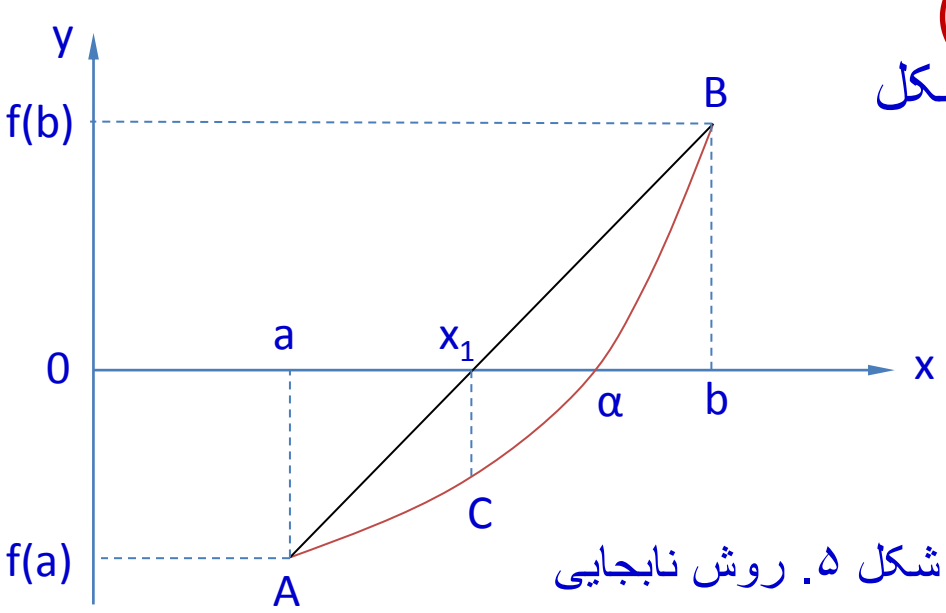
```
>> vel=@(t) sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);

>> fplot(vel,[0 12])      % It plots vel from t = 0 to 12
```

۲-۳-۲ روش نابجایی (False Position)

فرض کنید نمودار $y = f(x)$ به صورت شکل روبرو باشد.

دو نقطه A و B واقع بر منحنی را با یک خط مستقیم به هم وصل می کنیم، محل تلاقی این خط با محور x ها را به عنوان اولین تقریب α یعنی x_1 در نظر می گیریم.



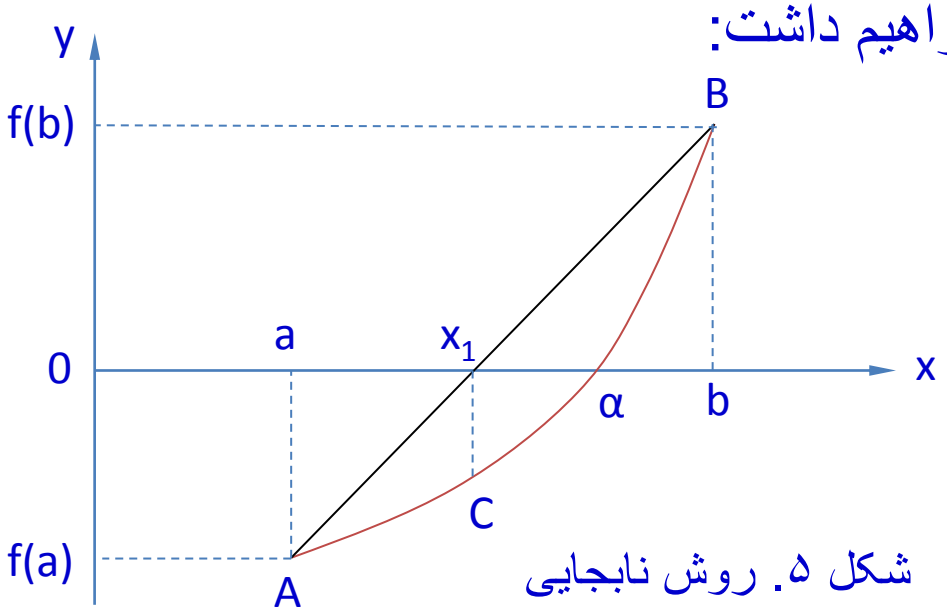
شکل ۵. روش نابجایی

حال چون (طبق شکل بالا) ریشه بین x_1 و b است دوباره با یک خط مستقیم دو نقطه B و C بر روی منحنی را به هم وصل می کنیم و محل تلاقی این خط با محور xها را x_2 یعنی دومین تقریب ریشه در نظر می گیریم. این کار را تا جایی ادامه می دهیم که به اندازه کافی به ریشه α نزدیک شویم.

برای تعیین x_1 ابتدا معادله خط AB را می نویسیم، این معادله به صورت زیر است:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}$$

زیرا مختصات نقاط A و B به ترتیب عبارتند از (a, f(a)) و (b, f(b)). چون $(x_1, 0)$ بر روی خط AB واقع است، از اینرو خواهیم داشت:



شکل ۵. روش نابجایی

$$\frac{0-f(a)}{f(b)-f(a)}=\frac{x_1-a}{b-a}$$

و از آن

$$x_1=\frac{af(b)-bf(a)}{f(b)-f(a)}$$

حال با توجه به اینکه ریشه در فاصله $[a, x_1]$ و یا در فاصله $[x_1, b]$ قرار گرفته باشد، عمل بالا را در یکی از فاصله های یاد شده تکرار می کنیم بنابراین با توجه به فرضهای روش دو بخشی، در روش نابجایی قرار می دهیم:

$$x=\frac{af(b)-bf(a)}{f(b)-f(a)} \qquad (۱)$$

و حالتهای زیر را بررسی می کنیم:

اگر $f(a)f(x) < 0$ آنگاه ریشه در (a, x) است، از اینرو قرار می دهیم $b = x$ و جدید را از رابطه (۱) حساب می کنیم.

اگر $f(a)f(x) = 0$ آنگاه ریشه برابر x بوده و کار تمام است.

نکته: روش نابجایی نیز مانند روش دو بخشی همگرایی تضمین شده دارد.

مثال ۳. تقریبی از ریشه معادله $f(x) = 3x - e^{-x} = 0$ را به روش نابجایی با سه رقم اعشار بدست آورید. این ریشه در فاصله $(0.25, 0.27)$ قرار دارد. محاسبات را تا جایی ادامه دهید که $|f(x_n)| \le 2\times10^{-4}$

حل: با توجه به رابطه (۱) و اینکه $a = 0.25$ و $b = 0.27$ داریم:

$$x_1 = \frac{0.25\times0.0466 - 0.27\times(-0.0288)}{0.0466 - (-0.0288)} = 0.2576 \qquad f(x_1) = -0.0001$$

از آنجا که $|f(x_1)| = 0.0001 < 2\times10^{-4}$ بنابراین x_1 تقریب ریشه بوده و این تقریب با سه رقم اعشار عبارت است از: $\alpha \approx 0.258$

تکلیف ۴. نتیجه مثال ۳ را با مثال ۱ مقایسه کنید.

Example 4. A Case Where Bisection Is Preferable to False Position

Use bisection and false position to locate the root of $f(x) = x^{10} - 1$ between x = 0 and 1.3.

Solution. Using bisection, the results can be summarized as

Iteration	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	0	1.3	0.65	100.0	35
2	0.65	1.3	0.975	33.3	2.5
3	0.975	1.3	1.1375	14.3	13.8
4	0.975	1.1375	1.05625	7.7	5.6
5	0.975	1.05625	1.015625	4.0	1.6

$\varepsilon_a = (x_r^{new} - x_r^{old}) / x_r^{new}\ 100\%$ $\varepsilon_t = (x_t - x_r) / x_t\ 100\%$

Thus, after five iterations, the true error is reduced to less than 2%.

For false position, a very different outcome is obtained:

Iteration	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	0	1.3	0.09430		90.6
2	0.09430	1.3	0.18176	48.1	81.8
3	0.18176	1.3	0.26287	30.9	73.7
4	0.26287	1.3	0.33811	22.3	66.2
5	0.33811	1.3	0.40788	17.1	59.2

ε_a and ε_t can be calculated in Excel as shown in Fig. 6. In column A, x_r is written then the formula of ε_a is written as demonstrated in column B. ε_t is calculated in column C.

$$\varepsilon_a = (x_r^{new} - x_r^{old}) / x_r^{new}\ 100\% \qquad \varepsilon_t = (x_t - x_r) / x_t\ 100\%$$

B2 f_x =ABS((A2-A1)/A2)*100

	A x_r	B ε_a	C ε_t	D	E	F
1	0.65	100	35			
2	0.975	33.33333	2.5			
3	1.1375	14.28571	13.75			
4	1.05625	7.692308	5.625			
5	1.015625	4	1.5625			
6						

Fig. 6 Error calculation in Excel

After five iterations, the true error has only been reduced to about 59%. Insight into these results can be gained by examining a plot of the function. As in Figure 7, the curve violates the premise on which false position was based - that is, if $f(x_l)$ is much closer to zero than $f(x_u)$, then the root should be much closer to x_l than to x_u (recall Figure 5). Because of the shape of the present function, the opposite is true.

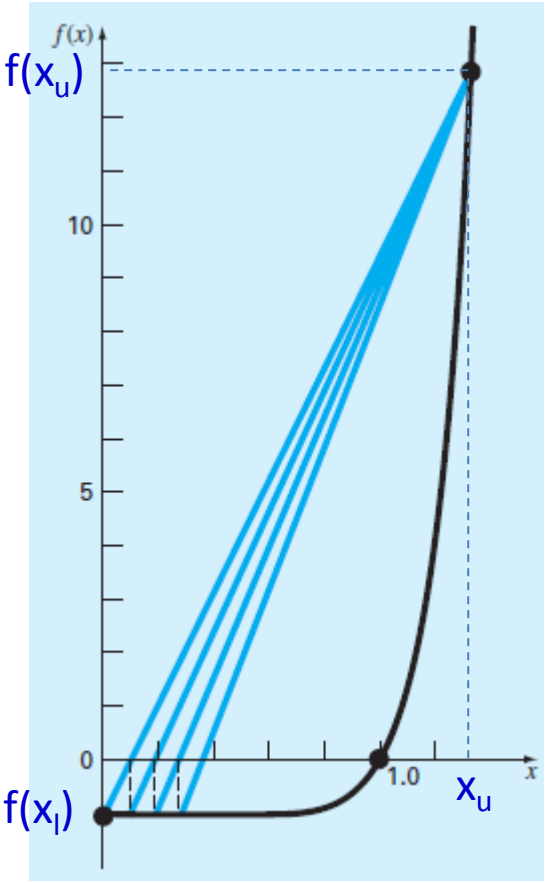
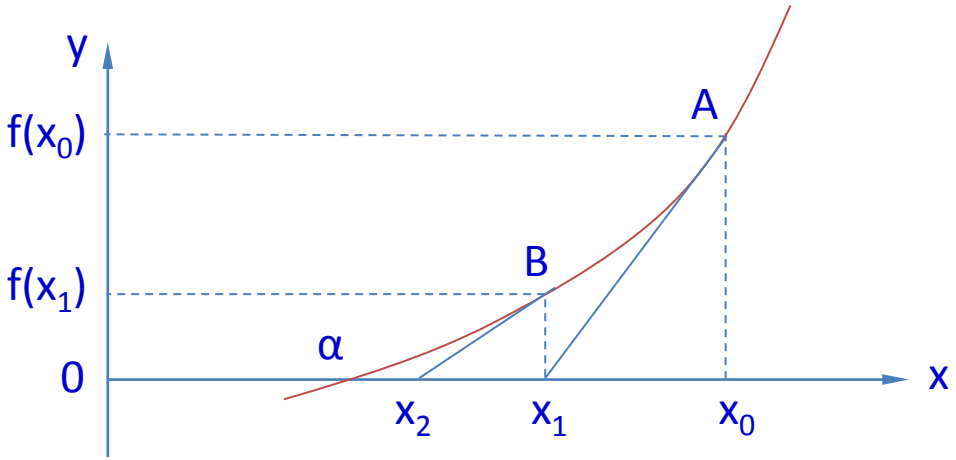


Fig. 7 Plot of $f(x) = x^{10} - 1$, illustrating slow convergence of the false-position method.

۳-۳-۲ روش نیوتن – رفسون (Newton-Raphson)

برای توضیح این روش، فرض کنید نمودار $y = f(x)$ به صورت زیر باشد:

همانگونه که شکل ۸ نشان می دهد α ریشه مورد نظر است. هرگاه x_0 تقریبی از ریشه باشد، از نقطه $A(x_0, f(x_0))$ واقع بر منحنی $y = f(x)$ مماس بر منحنی را رسم می کنیم. محل تلاقی این مماس را با محور طولها x_1 می نامیم. سپس



شکل ۸. روش نیوتن - رفسون

از نقطه $B(x_1, f(x_1))$ واقع بر منحنی مماس را رسم می کنیم و محل تلاقی این مماس جدید را با محور طولها x_2 می نامیم. این عمل را تا جایی که به تقریب مطلوب برسیم، یعنی x_nها به اندازه کافی به ریشه α نزدیک شوند، ادامه می دهیم.

با داشتن x_0 برای تعیین x_1، بایستی معادله خط مماس بر منحنی $y = f(x)$ را در نقطه $A(x_0, f(x_0))$ بنویسیم و محل تلاقی آن را با محور xها تعیین کنیم. ضریب زاویه این خط مماس $m = f'(x_0)$ است، بنابراین معادله خط مماس عبارتست از:

$$y - f(x_0) = f'(x_0)(x-x_0)$$

محل تلاقی این خط با محور طولها را $(x_1, 0)$ می گیریم، از اینرو

$$0 - f(x_0) = f'(x_0)(x_1-x_0)$$

که اگر $f'(x_0) \neq 0$ خواهیم داشت:

$$x_1 = x_0 - f(x_0) / f'(x_0)$$

بنابراین در حالت کلی با در دست داشتن x_n خواهیم داشت:

$$x_{n+1} = x_n - f(x_n) / f'(x_n) \qquad (۲)$$

رابطه (۲) فرمول تکرار روش نیوتن – رفسون یا به اختصار روش تکرار نیوتن نامیده می شود.

نکته: روش نیوتن تضمین همگرایی ندارد، اما به محض قرار گرفتن در مسیر همگرایی می توان نشان داد x_nها سریع به جواب مورد نظر میل می کند.

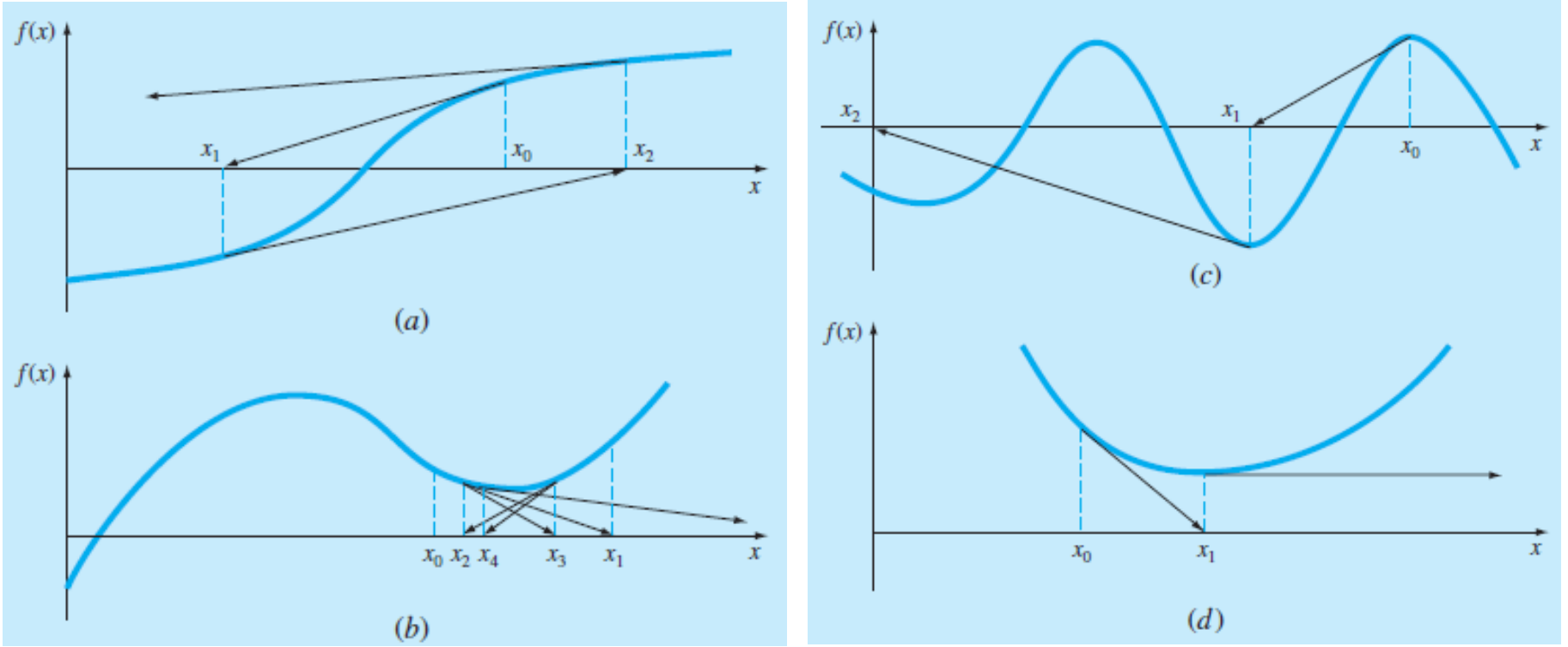


Fig. 9 Four cases where the Newton-Raphson method exhibits poor convergence.

مثال ۵. ریشه معادله $f(x) = x - \cos x = 0$ را که در فاصله $[0, 1]$ قرار دارد به روش نیوتن با چهار رقم اعشار بدست آورید بطوریکه $|x_n - x_{n-1}| < 10^{-4}$ که x_n تقریب ریشه مورد نظر در تکرار nام است. قرار دهید $x_0 = 0.5$.

حل: داریم $f(x) = x - \cos x$ و $f'(x) = 1 + \sin x$، از رابطه (۲) خواهیم داشت:

$$x_{n+1} = x_n - \frac{x_n - \cos x_n}{1 + \sin x_n}$$

با قرار دادن $x_0 = 0.5$ داریم: $x_1 = 0.75522$، $x_2 = 0.73914$ و $x_3 = 0.73909$ چون $|x_3 - x_2| = 5 \times 10^{-5} < 10^{-4}$ از اینرو x_3 تقریب ریشه مورد نظر است و با 4D این تقریب عبارتست از: $\alpha \approx 0.7391$.

Perhaps the most widely used of all root-locating formulas is the Newton-Raphson equation. It is the best-known method of finding roots for a good reason: it is simple and fast. The only drawback of the method is that it uses the derivative f(x) of the function as well as the function f(x) itself. Therefore, Newton–Raphson method is usable only in problems where f'(x) can be readily computed.

The Newton–Raphson formula can be derived from the Taylor series expansion of f(x) about x:

$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + f''(x_i)(x_{i+1} - x_i)^2/2!$ (a)

If x_{i+1} is a root of f(x) = 0, Eq. (a) becomes

$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) + f''(x_i)(x_{i+1} - x_i)^2/2!$ (b)

Assuming that x_i is close to x_{i+1}, we can drop the last term in Eq. (b) and solve for x_{i+1}. The result is the Newton–Raphson formula

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

Letting x denote the true value of the root, the error in x_i is $E_i = x - x_i$. It can be shown that if x_{i+1} is computed from Eq. (c), the corresponding error is

$$E_{i+1} = -\frac{f''(x_i)}{2f'(x_i)} E_i^2$$

indicating that Newton–Raphson method converges quadratically (the error is the square of the error in the previous step). As a consequence, the number of significant figures is roughly doubled in every iteration, provided that x_i is close to the root.

MATLAB M-file: newtraph

An algorithm for the Newton-Raphson method can be easily developed. Note that the program must have access to the function (func) and its first derivative (dfunc). These can be simply accomplished by the inclusion of user-defined functions to compute these quantities. Alternatively, as in the algorithm, they can be passed to the function as arguments.

```
function [root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,varargin)
% newtraph: Newton-Raphson root location zeroes
% [root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,p1,p2,...):
% uses Newton-Raphson method to find the root of func
```

```
% input:
% func = name of function
% dfunc = name of derivative of function
% xr = initial guess
% es = desired relative error (default = 0.0001%)
% maxit = maximum allowable iterations (default = 50)
% p1,p2,... = additional parameters used by function
% output:
% root = real root
% ea = approximate relative error (%)
% iter = number of iterations

if nargin<3,error('at least 3 input arguments required'),end
if nargin<4|isempty(es),es=0.0001;end
if nargin<5|isempty(maxit),maxit=50;end
```

```
iter = 0;
while (1)
xrold = xr;
xr = xr - func(xr)/dfunc(xr);
iter = iter + 1;
if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
if ea <= es | iter >= maxit, break, end
end
root = xr;
```

$$x_{n+1} = x_n - f(x_n) / f'(x_n) \qquad (2)$$

After the M-file is entered and saved, it can be invoked to solve for root. For example, for the simple function $x^2 - 9$, the root can be determined as in

```
>> newtraph(@(x) x^2-9,@(x) 2*x,5)
ans =
3
```

[root,ea,iter]=newtraph(func,dfunc,xr,es,maxit,varargin)

EXAMPLE 6 Newton-Raphson Bungee Jumper Problem

Problem Statement. Use the M-file function to determine the mass of the bungee jumper with a drag coefficient of 0.25 kg/m to have a velocity of 36 m/s after 4 s of free fall. The acceleration of gravity is 9.81 m/s^2.

Solution. The function to be evaluated is

$$f(m)=\sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)-v(t)$$

To apply the Newton-Raphson method, the derivative of this function must be evaluated with respect to the unknown, m:

$$\frac{df(m)}{dm}=\frac{1}{2}\sqrt{\frac{g}{mc_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)-\frac{g}{2m}t\,\operatorname{sech}^2\left(\sqrt{\frac{gc_d}{m}}t\right)$$

We should mention that although this derivative is not difficult to evaluate in principle, it involves a bit of concentration and effort to arrive at the final result.
The two formulas can now be used in conjunction with the function newtraph to evaluate the root:

```
>> y = @m sqrt(9.81*m/0.25)*tanh(sqrt(9.81*0.25/m)*4)-36;
>> dy = @m 1/2*sqrt(9.81/(m*0.25))*tanh((9.81*0.25/m) ...
          ^(1/2)*4)-9.81/(2*m)*sech(sqrt(9.81*0.25/m)*4)^2;
>> newtraph(y,dy,140,0.00001)
ans =
142.7376
```

۲-۳-۴ روش وتری (Secant)

A potential problem in implementing the Newton-Raphson method is the evaluation of the derivative. Although this is not inconvenient for polynomials and many other functions, there are certain functions whose derivatives may be difficult or inconvenient to evaluate. For these cases, the derivative can be approximated by a backward finite divided difference:

$$f'(x_i) \cong \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

$$\lim_{x \to x_n} \frac{f(x_n) - f(x)}{x_n - x} = f'(x_n)$$ می دانیم

هرگاه x مقداری نزدیک x_n باشد، مثلا x_{n-1}، در این صورت

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \cong f'(x_n) \qquad (۳)$$

بنابراین هرگاه در فرمول نیوتن، یعنی رابطه (۲)، به جای $f'(x_n)$ مقدار تقریبی آنرا از رابطه (۳) قرار دهیم بدست می آوریم:

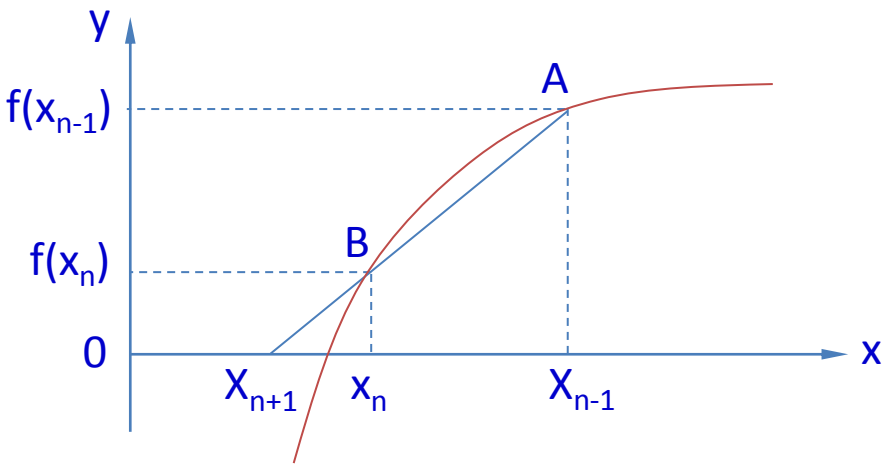
$$x_{n+1} = x_n - f(x_n) / f'(x_n) \qquad (2)$$

$$\frac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}} \cong f'(x_n) \qquad (3)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{\dfrac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}}$$

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n)-f(x_{n-1})} \qquad (۴)$$

رابطه (۴) فرمول روش وتری برای بدست آوردن ریشه معادله $f(x) = 0$ نامیده می شود. برای محاسبه x_n ها از فرمول وتری به دو مقدار اولیه x_0 و x_1 نیاز داریم. علت اینکه این روش را وتری می نامند آن است که در مرحله nام x_{n+1} از محل برخورد خط (وتر) وصل نقاط $A(x_{n-1}, f(x_{n-1}))$ و $B(x_n, f(x_n))$ با محور xها به دست می آید.



شکل ۱۰. روش وتری

Starting with the two initial approximations p_0 and p_1, the approximation p_2 is the x-intercept of the line joining $(p_0, f(p_0))$ and $(p_1, f(p_1))$. The approximation p_3 is the x-intercept of the line joining $(p_1, f(p_1))$ and $(p_2, f(p_2))$, and so on.

Note that only one function evaluation is needed per step for the Secant method after p_2 has been determined.

In contrast, each step of Newton's method requires an evaluation of both the function and its derivative.

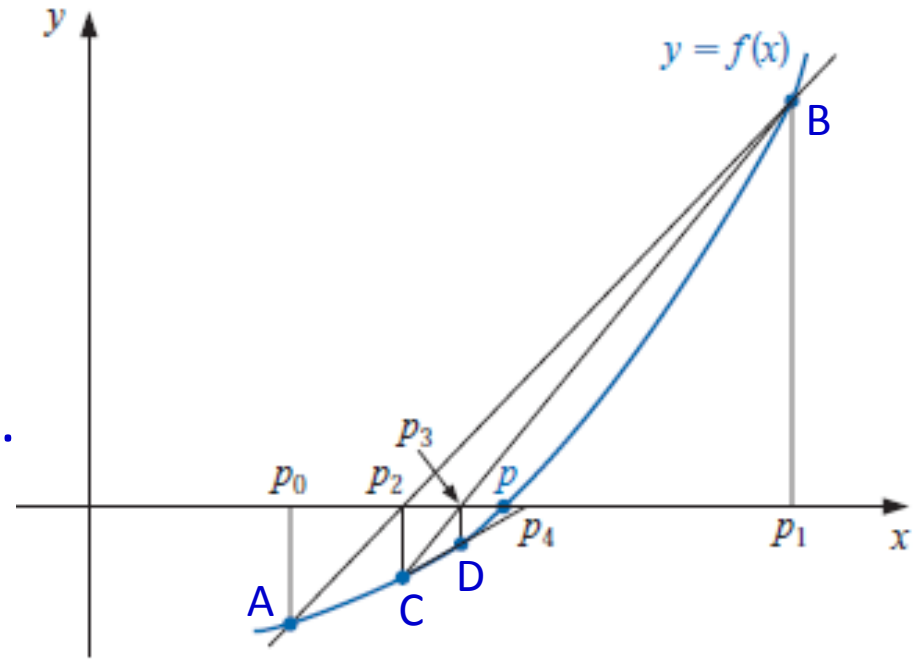


Fig. 11 Secant Method

نکته: تفاوت عمده روش وتری با روش نابجایی در این است که در روش نابجایی در هر مرحله بررسی و فاصله ای در نظر گرفته می شود که تابع در آن تغییر علامت می دهد. در حالیکه در روش وتری صرفا زیر فاصله آخر که تنها نقاط انتهایی آن در تکرار اخیر بدست آمده اند، جهت تکرار روش به کاربرده می شود و به همین دلیل روش وتری تضمین همگرایی ندارد. اما می توان نشان داد که سرعت همگرایی روش وتری (در صورت همگرایی) بیشتر از روش نابجایی است.

The Difference Between the Secant and False-Position Methods

Note the similarity between the secant method and the false-position method. For example, Eqs. (4) and (1) are identical on a term-by-term basis. Both use two initial estimates to compute an approximation of the slope of the function that is used to project to the x axis for a new estimate of the root. However, a critical difference between the methods is how one of the initial values is replaced by the new estimate. Recall that in the false-position method the latest estimate of the root replaces whichever of the original values yielded a function value with the same sign as $f(x_r)$.

Consequently, the two estimates always bracket the root. Therefore, for all practical purposes, the method always converges because the root is kept within the bracket. In contrast, the secant method replaces the values in strict sequence, with the new value x_{i+1} replacing x_i and x_i replacing x_{i-1}. As a result, the two values can sometimes lie on the same side of the root. For certain cases, this can lead to divergence.

Example 7 The Secant Method

Problem Statement. Use the secant method to estimate the root of $f(x) = e^{-x} - x$. Start with initial estimates of $x_{-1} = 0$ and $x_0 = 1.0$.

Solution. Recall that the true root is 0.56714329. . . .

First iteration: $x_{-1} = 0 \quad f(x_{-1}) = 1.00000$

$x_0 = 1 \quad f(x_0) = -0.63212$

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \qquad (4)$$

(4): $$x_1 = 1 - \frac{-0.63212(0 - 1)}{1 - (-0.63212)} = 0.61270 \qquad \varepsilon_t = 8.0\%$$

ε_t = (true value – approximation) / true value × 100%

Second iteration: $x_0 = 1$ $f(x_0) = -0.63212$

$x_1 = 0.61270$ $f(x_1) = -0.07081$

(Note that both estimates are now on the same side of the root.)

$$x_2 = 0.61270 - \frac{-0.07081(1 - 0.61270)}{-0.63212 - (-0.07081)} = 0.56384 \qquad \varepsilon_t = 0.58\%$$

Third iteration:

$x_1 = 0.61270$ $f(x_1) = -0.07081$

$x_2 = 0.56384$ $f(x_2) = 0.00518$

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \qquad (4)$$

$$x_3 = 0.56384 - \frac{0.00518(0.61270 - 0.56384)}{-0.07081 - (-0.00518)} = 0.56717 \qquad \varepsilon_t = 0.0048\%$$

Example 8 Comparison of Convergence of the Secant and False-Position Techniques

Problem Statement. Use the false-position and secant methods to estimate the root of f(x) = ln x. Start the computation with values of $a = x_l = x_{i-1} = 0.5$ and $b = x_u = x_i = 5.0$.

Solution. For the false-position method, the use of Eq. 1 and the bracketing criterion for replacing estimates results in the following iterations:

$$x = \frac{af(b) - bf(a)}{f(b) - f(a)} \qquad (1)$$

Iteration	x_l	x_u	x_r
1	0.5	5.0	1.8546
2	0.5	1.8546	1.2163
3	0.5	1.2163	1.0585

As can be seen (Fig. 12a and c), the estimates are converging on the true root which is equal to 1.

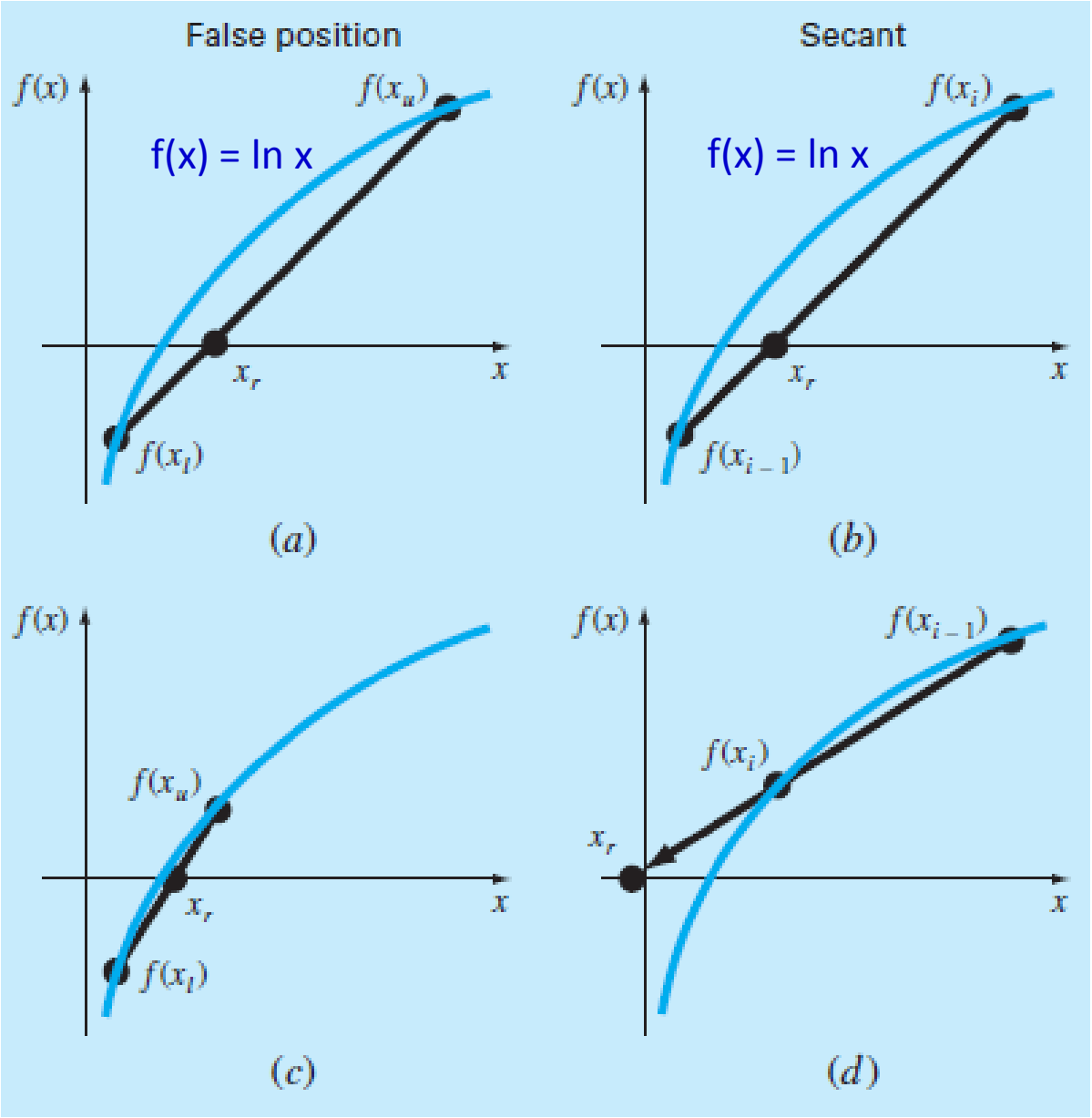
For the secant method, using Eq. 4 and the sequential criterion for replacing estimates results in

$$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \qquad (4)$$

Iteration	x_{i-1}	x_i	x_{i+1}
1	0.5	5.0	1.8546
2	5.0	1.8546	−0.10438

As in Fig. 12d, the approach is divergent.

Fig. 12
Comparison of the false-position and the secant methods. The first iterations (a) and (b) for both techniques are identical. However, for the second iterations (c) and (d), the points used differ. As a consequence, the secant method can diverge, as indicated in (d).



Although the secant method may be divergent, when it converges it usually does so at a quicker rate than the false-position method. For instance, Fig. 13 demonstrates the superiority of the secant method in this regard. The inferiority of the false-position method is due to one end staying fixed to maintain the bracketing of the root. This property, which is an advantage in that it prevents divergence, is a shortcoming with regard to the rate of convergence; it makes the finite-difference estimate a less-accurate approximation of the derivative.

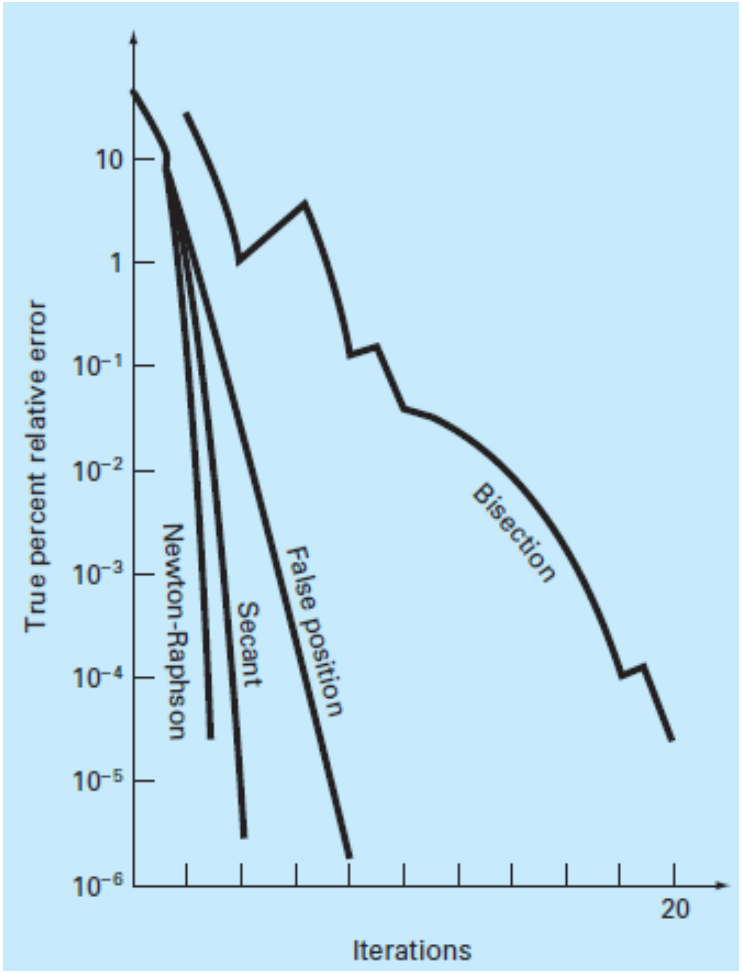


Fig. 13 Comparison of the true percent relative errors ε_t for the methods to determine the roots of $f(x) = e^{-x} - x$.