

プログラミングII

<http://bit.ly/Prog3i>

数値計算 (1)

前期 第13週

2019/7/16

本日は・・・

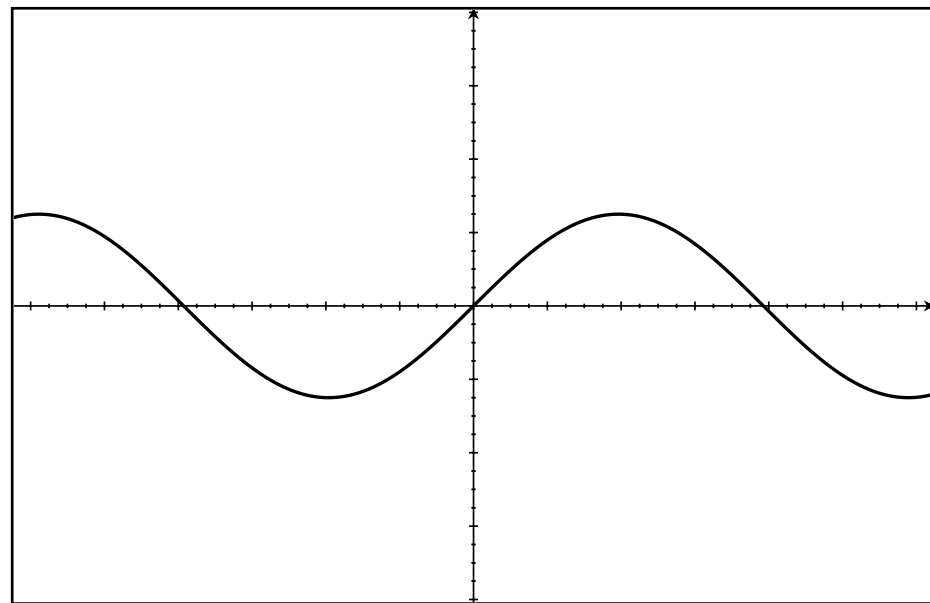
近似値を求めるアルゴリズムについて学びます

例：平方根を求めるアルゴリズム

連続的と離散的

現実や数学の世界では

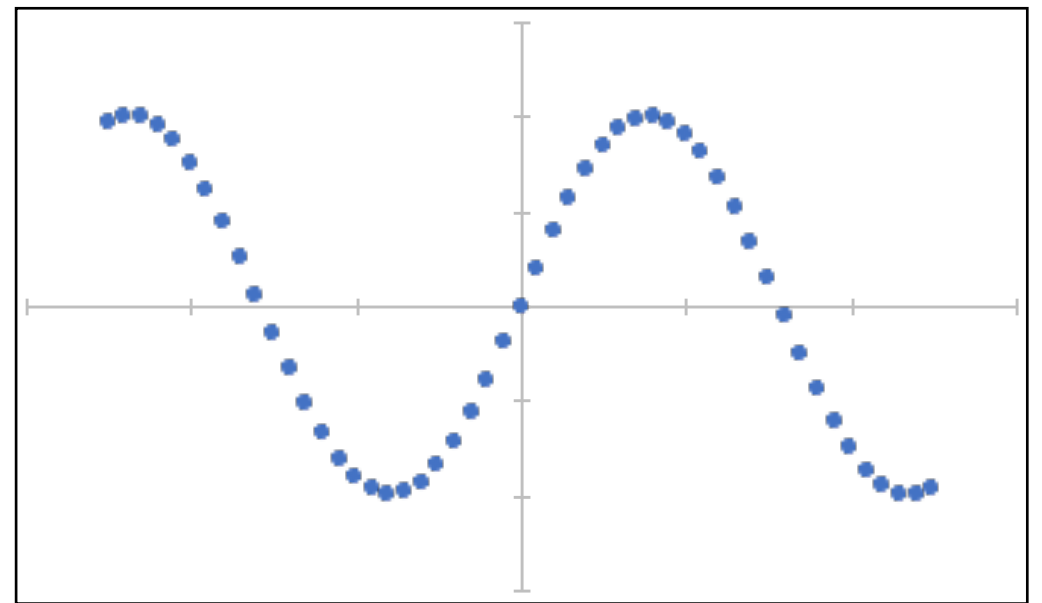
連続的に表現するデータでも...



連続的に表現された $\sin(x)$

コンピュータで表現される

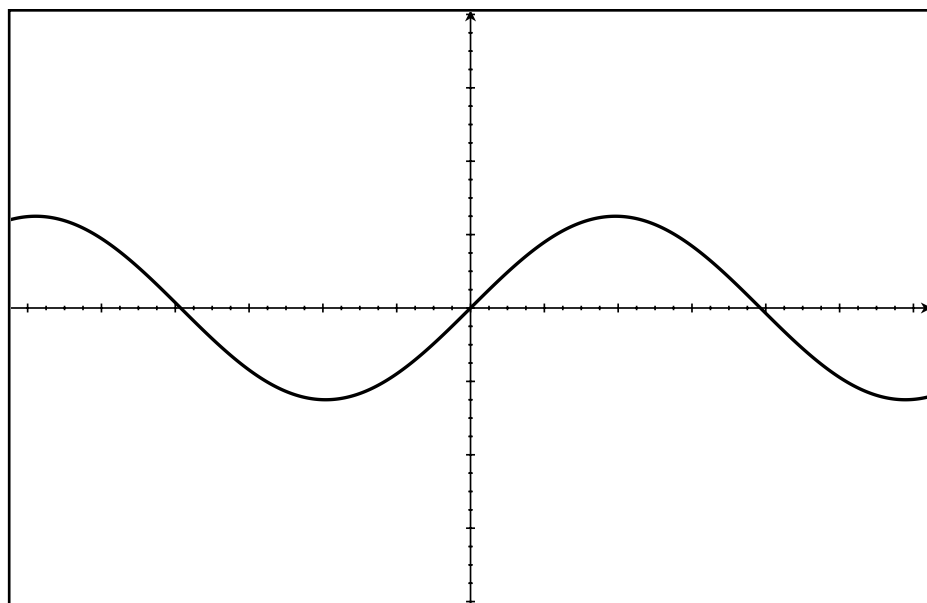
データは**離散的**になる



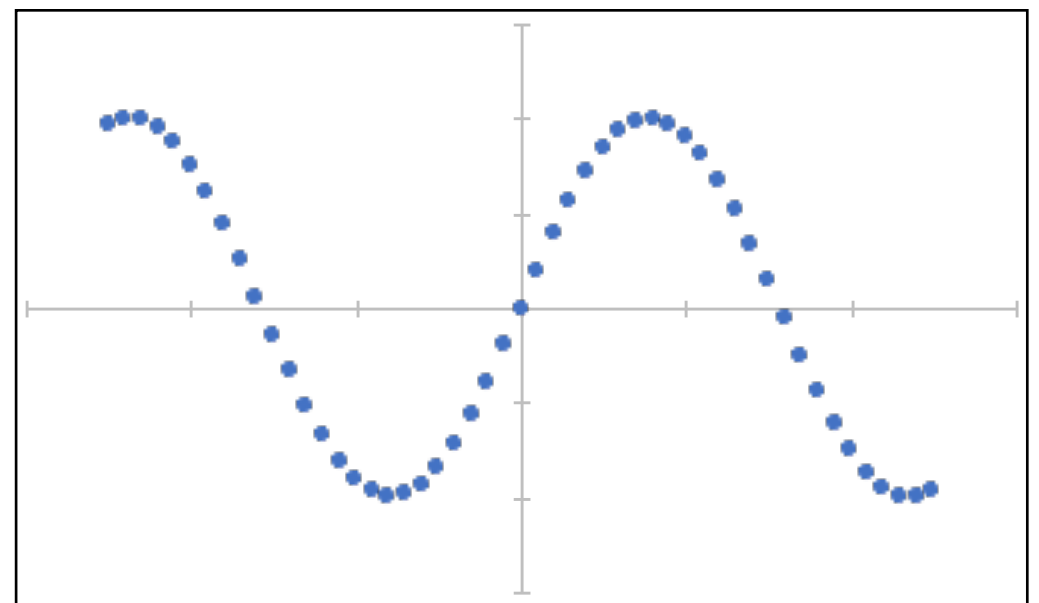
離散的に表現された $\sin(x)$

連続的と離散的

連続的に表現されたデータは、
離散的に表現されたデータに**近似**される



近似
→



連続的に表現された $\sin(x)$

離散的に表現された $\sin(x)$

近似のアルゴリズム例

今回は近似のアルゴリズムの例として、

平方根の近似値を求めるアルゴリズムを考えます

$$\sqrt{2}$$

近似



1.414213...

2の平方根を求めるには

次のようなxを求めたいので式を変形

$$x = \sqrt{2}$$

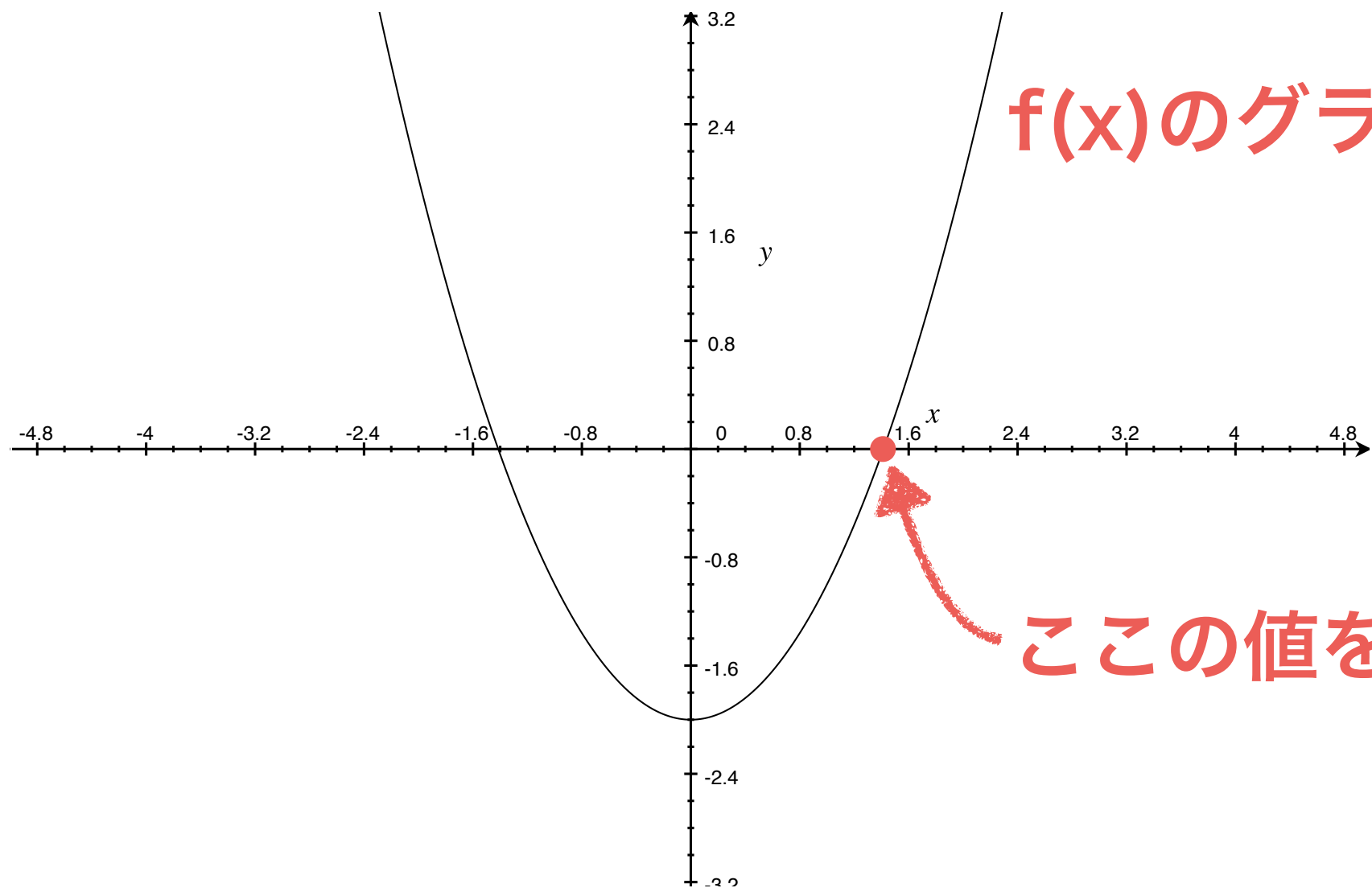
$$x^2 = 2$$

$$x^2 - 2 = 0$$

2の平方根を求めるには

$$f(x) = x^2 - 2 \quad \text{とすると}$$

$f(x) = 0$ となる箇所が2の平方根



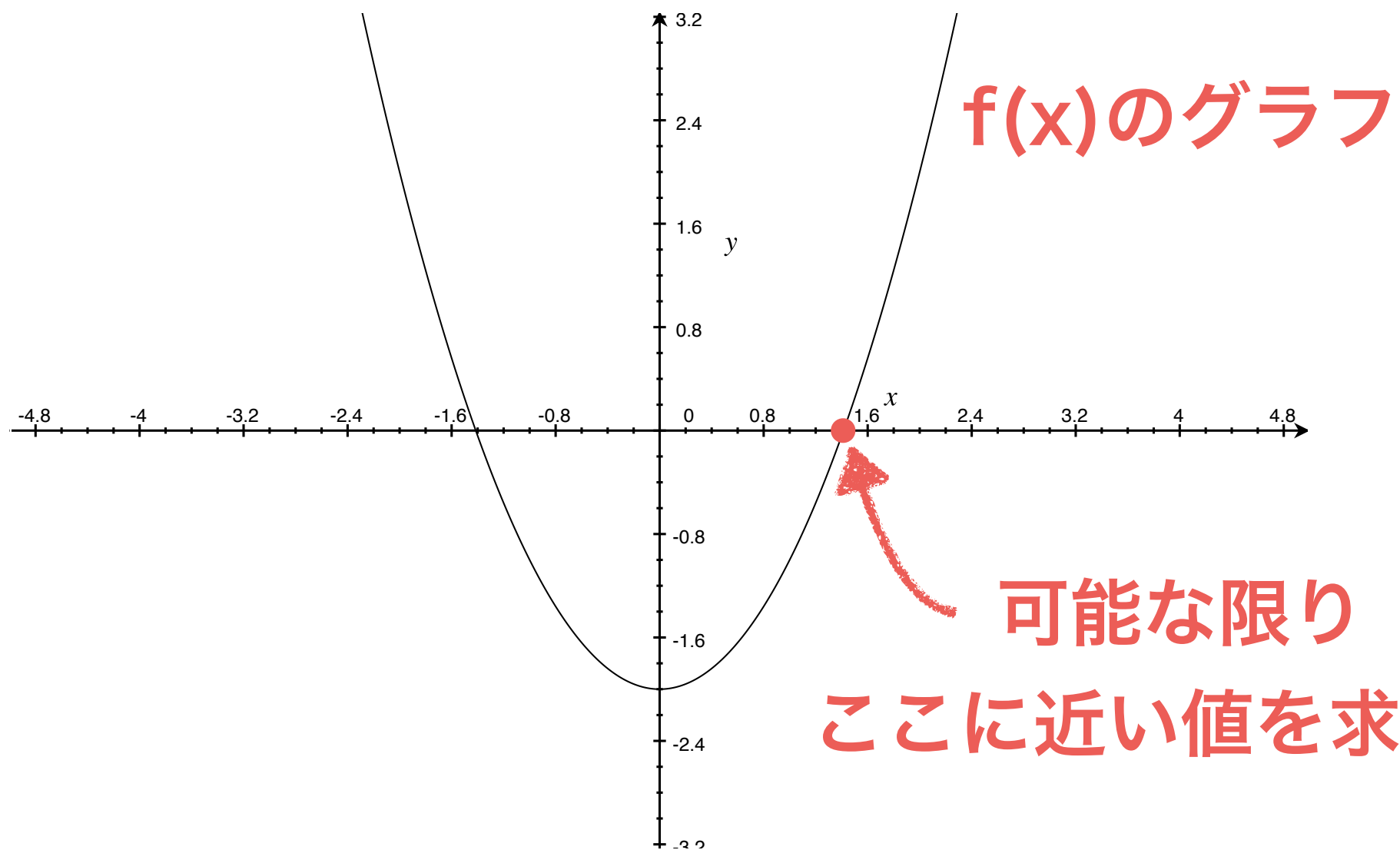
$f(x)$ のグラフ

この値を求めたい

ニュートン法

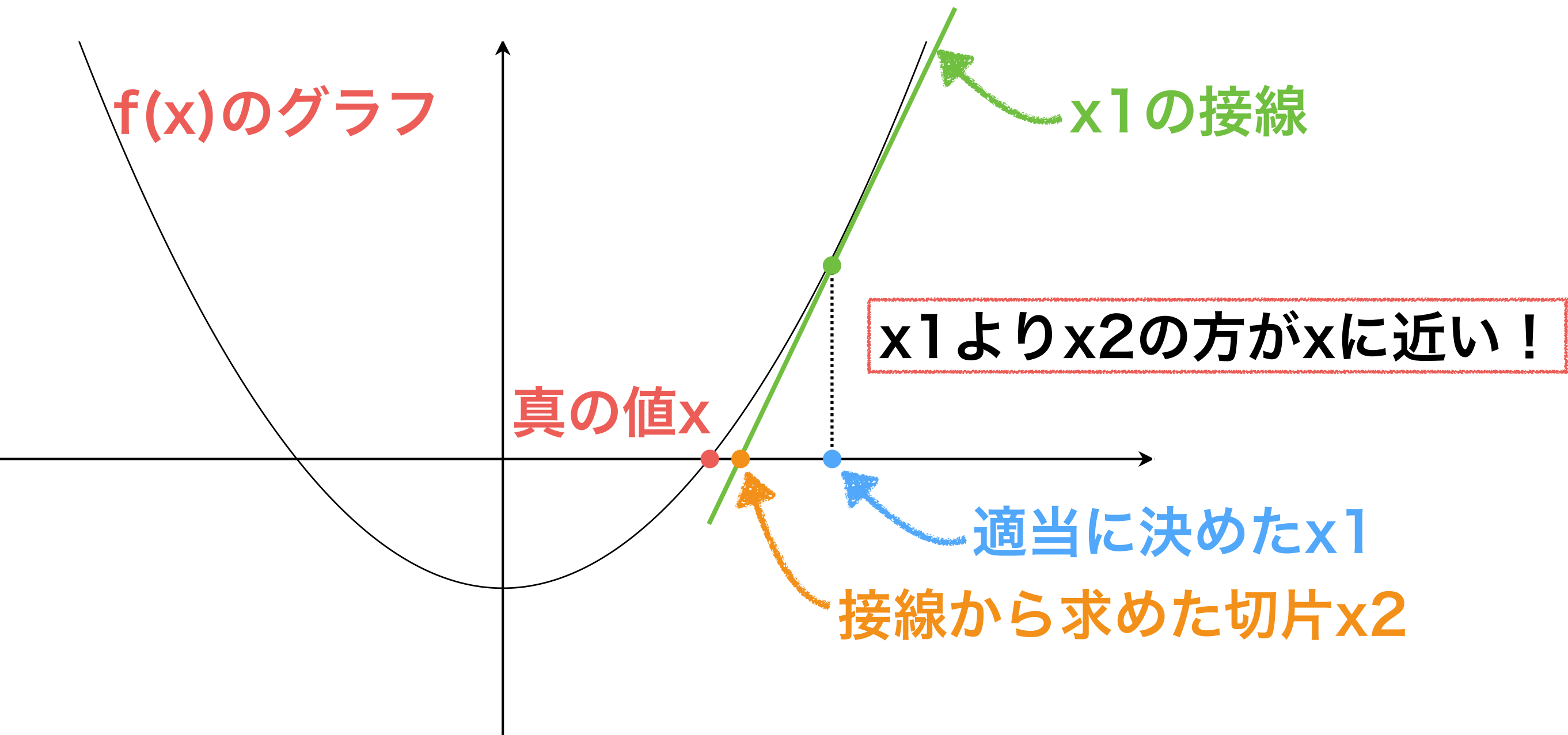
$f(x) = 0$ となるような

x の近似値を求める アルゴリズム



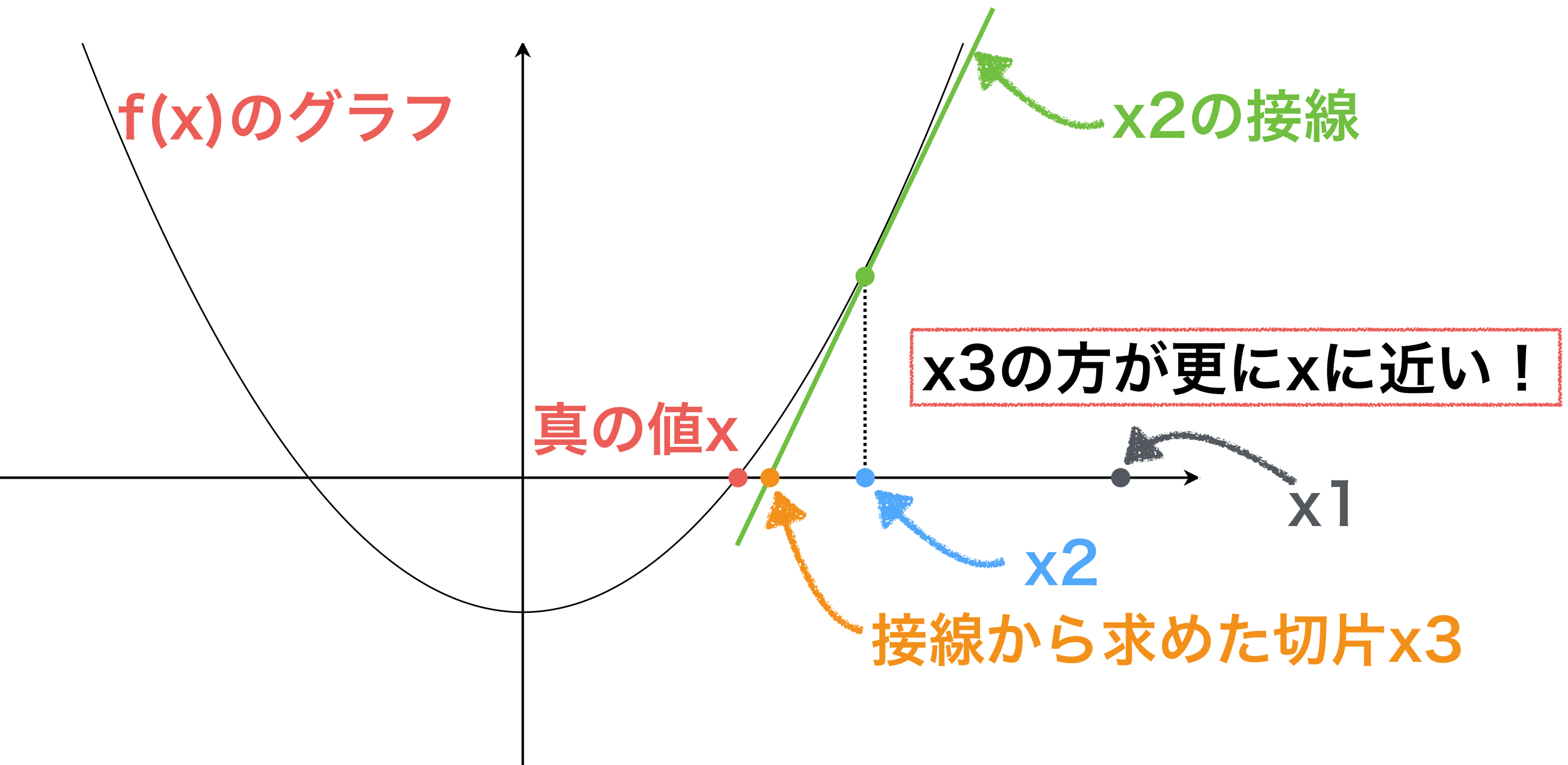
ニュートン法

$f(x)=0$ となるような x を探す場合、
ある値 x_1 における接線の切片 x_2 は、
 x_1 より真の値 x に近くなる



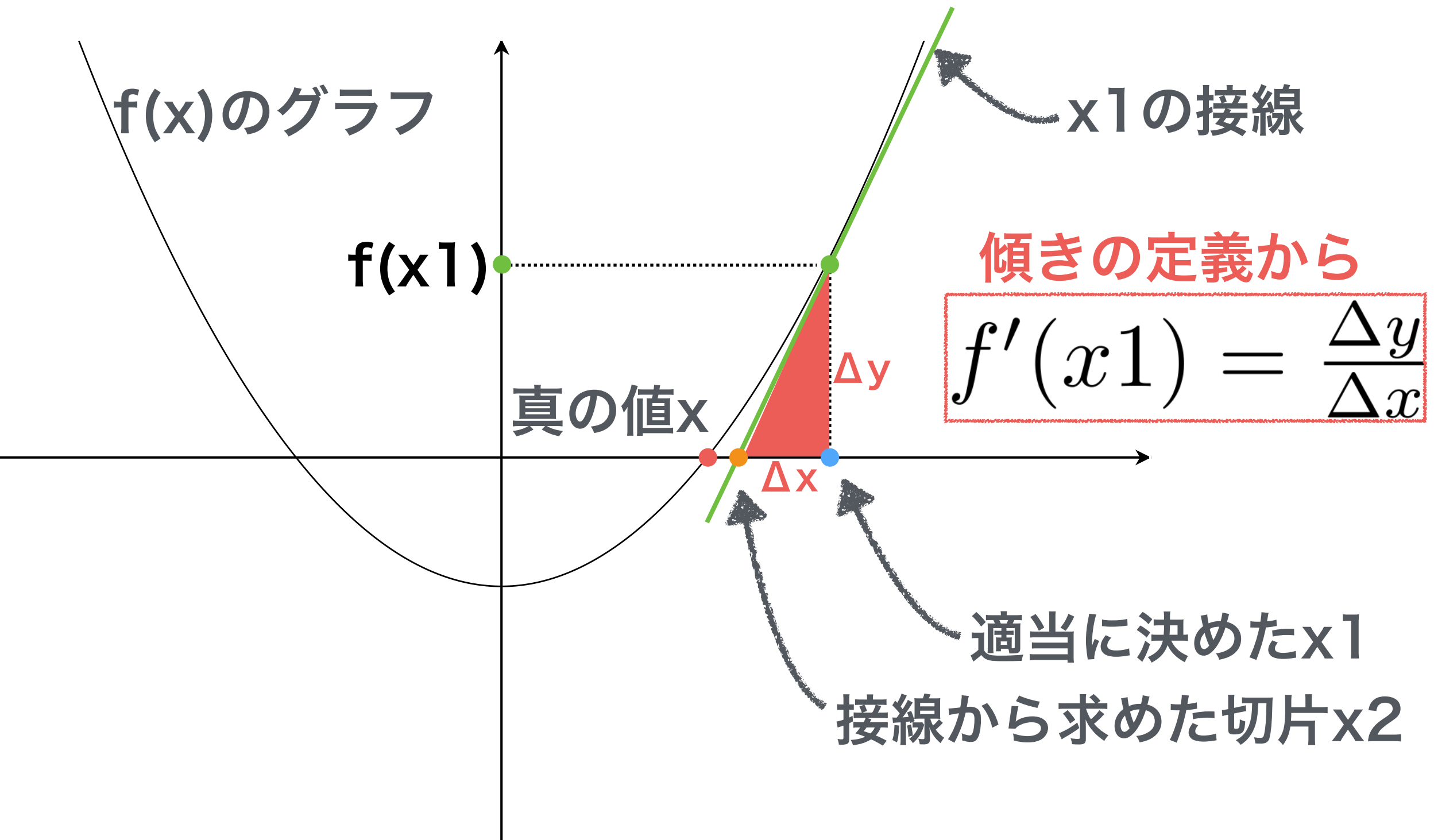
ニュートン法

x_2 に対して、同様に接線から x_3 を求めれば、
さらに真の値 x に近くなるので繰り返せば、
近似値の精度が上がっていく



x1からx2を求める

Δx と Δy からなる直角三角形に注目



x1 から x2 を求める

Δx と Δy からなる **直角三角形** に注目

$$f'(x1) = \frac{\Delta y}{\Delta x}$$

$$f'(x1) = \frac{f(x1)}{x1 - x2}$$

グラフから求まる

$$x1 - x2 = \frac{f(x1)}{f'(x1)}$$

$$x2 = x1 - \frac{f(x1)}{f'(x1)}$$

x1 から x2 を求める

2の平方根に当てはめると…

$$x2 = x1 - \frac{x1^2 - 2}{2 * x1}$$

cの平方根を求めるには…

$$x2 = x1 - \frac{x1^2 - c}{2 * x1}$$

この式の計算を繰り返してx2, x3, x4…と求めていけばよい

【課題の準備】

演習室で作業する前に、以下のコマンドを
入れるだけで準備が完了する

```
$ mygitclone 「自分のGitHubユーザ名」  
$ cd prog3i-(ユーザ名)  
$ ./myconf
```

※本体をシャットダウンするまでは、
上記「mygitclone」と「myconf」の設定は有効です

【課題の準備】

以下の流れで、課題のプログラムを作るためのフォルダを準備しましょう。

1. 端末を起動して、以下のコマンドを実行して前期第13週のフォルダを作る
\$ cd prog3i-(ユーザ名) (←既に移動しているなら不要)
\$ mkdir week113
\$ cd week113

【課題13-1】

ニュートン法を使って引数cの平方根を求める関数 `mysqrt()` を作成して下さい。

[この関数のプロトタイプ宣言]

```
double mysqrt(double c);
```

```
/* ニュートン法を使って、x1からx2を求める処理を繰り返す */
```

- ▶ 変数x1, x2はdouble型で宣言する
- ▶ x1の初期値は十分大きい値にしておく（今回は100）
- ▶ x1からx2を求める計算を20回繰り返す
- ▶ 繰り返し処理で、x2を求めたら、次のx1になるように代入する

【課題13-1】

[mainの処理]

```
printf("%lf\n", mysqrt(2));  
printf("%lf\n", mysqrt(3));  
printf("%lf\n", mysqrt(4));  
printf("%lf\n", mysqrt(5));
```

[実行結果]

```
1.414214  
1.732051  
2.000000  
2.236068
```

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題13-1提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題13-2】

課題13-1のmysqrt()に対して、繰り返し処理でx2が変化する様子が見えるように出力を追加してください。

[この関数のプロトタイプ宣言]

```
double mysqrt(double c);
```

```
/* 課題13-1で作った繰り返し処理の中でx2の出力を追加する */
```

繰り返す度に、x2が真の値xに近づいていくのを見ることができるようになります

【課題13-2】

[実行結果] (mainは課題13-1と同じ)

0:	50.010000	(←2の平方根を求める繰り返し処理の0回目の出力)
1:	25.024996	(←2の平方根を求める繰り返し処理の1回目の出力)
2:	12.552458	(←2の平方根を求める繰り返し処理の2回目の出力)
3:	6.355895	(←2の平方根を求める繰り返し処理の3回目の出力)
4:	3.335282	
5:	1.967466	
6:	1.492001	
7:	1.416241	
8:	1.414215	
9:	1.414214	(←2の平方根の真の値に近似していく様子が見える)
(途中省略)		
19:	1.414214	
	1.414214	

(3以降の平方根の出力も同様に続く)

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題13-2提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題13-3】

課題13-2の関数mysqrt()に対して、繰り返し処理を
「**x1とx2の差分が0.0001未満になったら終了する**」
ように改良した関数mysqrt2を作成してください。

[この関数のプロトタイプ宣言]

```
double mysqrt2(double c);
```

```
/* 繰り返す回数を20回ではなく、差分の値で終了するようにする */
```

- ▶ 繰り返し処理の条件は無条件（無限ループ）となるように変更する
- ▶ 繰り返し処理で、x1とx2の差が0.0001未満なら抜け出すようにする

【課題13-3】

[mainの処理]

```
printf("%lf\n", mysqrt2(2));  
printf("%lf\n", mysqrt2(3));  
printf("%lf\n", mysqrt2(4));  
printf("%lf\n", mysqrt2(5));
```

[実行結果]

0: 50.010000

1: 25.024996

2: 12.552458

3: 6.355895

(途中省略)

8: 1.414215

9: 1.414214

(←この場合は9回目で繰り返し処理を終了した)

1.414214

(3以降の平方根の出力も同様に続く)

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題13-3提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

まだ余裕のある人は…【課題13-4】

前回作成した配列処理のプログラムに「配列のi番目の値の平方根を出力する」コマンドsqrtを追加して下さい。

- ▶ array.lexに、コマンド「sqrt」と識別子SQRTの字句定義を追記する
- ▶ array.yaccに、このコマンドに関する構文を追加する
コマンドの後に、NUMBERが1個続く
- ▶ myproc.hにこの演算で呼び出す関数のプロトタイプ宣言を追加する
`int mysqrt3(int i);`
- ▶ myproc.cに関数mysqrt3の定義を追加する
配列aのi番目に、前の課題で作成したmysqrt2を呼び出して、
求めた平方根を出力する（つまり、mysqrt2の定義も必要）
最後に0を返す

※動作確認をするためには、コマンドclear, set, exitも少なくとも必要になります。

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題13-4提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))