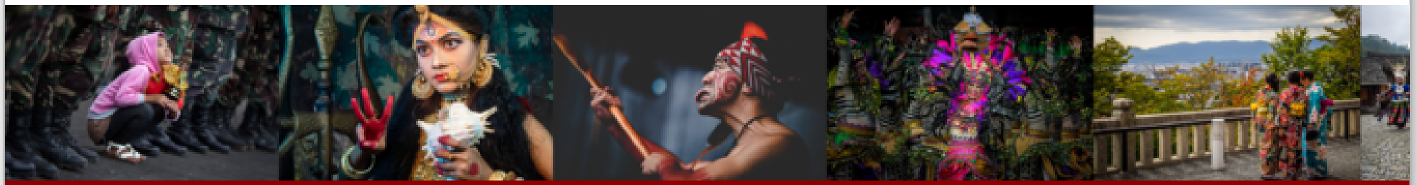
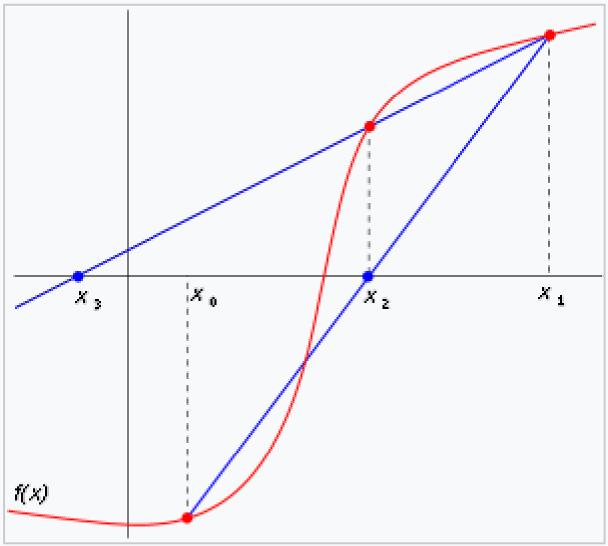


Secant method

24 languages

In numerical analysis, the **secant method** is a root-finding algorithm that uses a succession of roots of secant lines to better approximate a root of a function *f*. The secant method can be thought of as a finite-difference approximation of Newton's method. However, the secant method predates Newton's method by over 3000 years.[1]



The first two iterations of the secant method. The red curve shows the function *f*, and the blue lines are the secants. For this particular case, the secant method will not converge to the visible root.

The method [edit]

For finding a zero of a function f, the secant method is defined by the recurrence relation.

$$x_n = x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} = \frac{x_{n-2}f(x_{n-1}) - x_{n-1}f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}.$$

As can be seen from this formula, two initial values x_0 and x_1 are required. Ideally, they should be chosen close to the desired zero.

Derivation of the method [edit]

Starting with initial values x_0 and x_1, we construct a line through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$, as shown in the picture above. In slope–intercept form, the equation of this line is

$$y = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) + f(x_0).$$

The root of this linear function, that is the value of x such that $y = 0$ is

$$x = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}.$$

We then use this new value of x as x_2 and repeat the process, using x_1 and x_2 instead of x_0 and x_1. We continue this process, solving for x_3, x_4, etc., until we reach a sufficiently high level of precision (a sufficiently small difference between x_n and x_{n-1}):

$$\begin{aligned}
x_2 &= x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)},\\
x_3 &= x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)},\\
&\ \ \vdots\\
x_n &= x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}.
\end{aligned}$$

Convergence [edit]

The iterates x_n of the secant method converge to a root of f if the initial values x_0 and x_1 are sufficiently close to the root. The order of convergence is φ, where

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$$

is the golden ratio. In particular, the convergence is super linear, but not quite quadratic.

This result only holds under some technical conditions, namely that f be twice continuously differentiable and the root in question be simple (i.e., with multiplicity 1).

If the initial values are not close enough to the root, then there is no guarantee that the secant method converges. There is no general definition of "close enough", but the criterion has to do with how "wiggly" the function is on the interval $[x_0, x_1]$. For example, if f is differentiable on that interval and there is a point where $f' = 0$ on the interval, then the algorithm may not converge.

Comparison with other root-finding methods [edit]

The secant method does not require that the root remain bracketed, like the bisection method does, and hence it does not always converge. The false position method (or *regula falsi*) uses the same formula as the secant method. However, it does not apply the formula on x_{n-1} and x_{n-2}, like the secant method, but on x_{n-1} and on the last iterate x_k such that $f(x_k)$ and $f(x_{n-1})$ have a different sign. This means that the false position method always converges; however, only with a linear order of convergence. Bracketing with a super-linear order of convergence as the secant method can be attained with improvements to the false position method (see Regula falsi § Improvements in *regula falsi*) such as the ITP method or Illinois method.

The recurrence formula of the secant method can be derived from the formula for Newton's method

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

by using the finite-difference approximation, for a small ϵ:

$$f'(x_{n-1}) \approx \frac{f(x_{n-1}) - f(x_{n-2})}{x_{n-1} - x_{n-2}} \approx \frac{f(x_{n-1} + \frac{\epsilon}{2}) - f(x_{n-1} - \frac{\epsilon}{2})}{\epsilon}$$

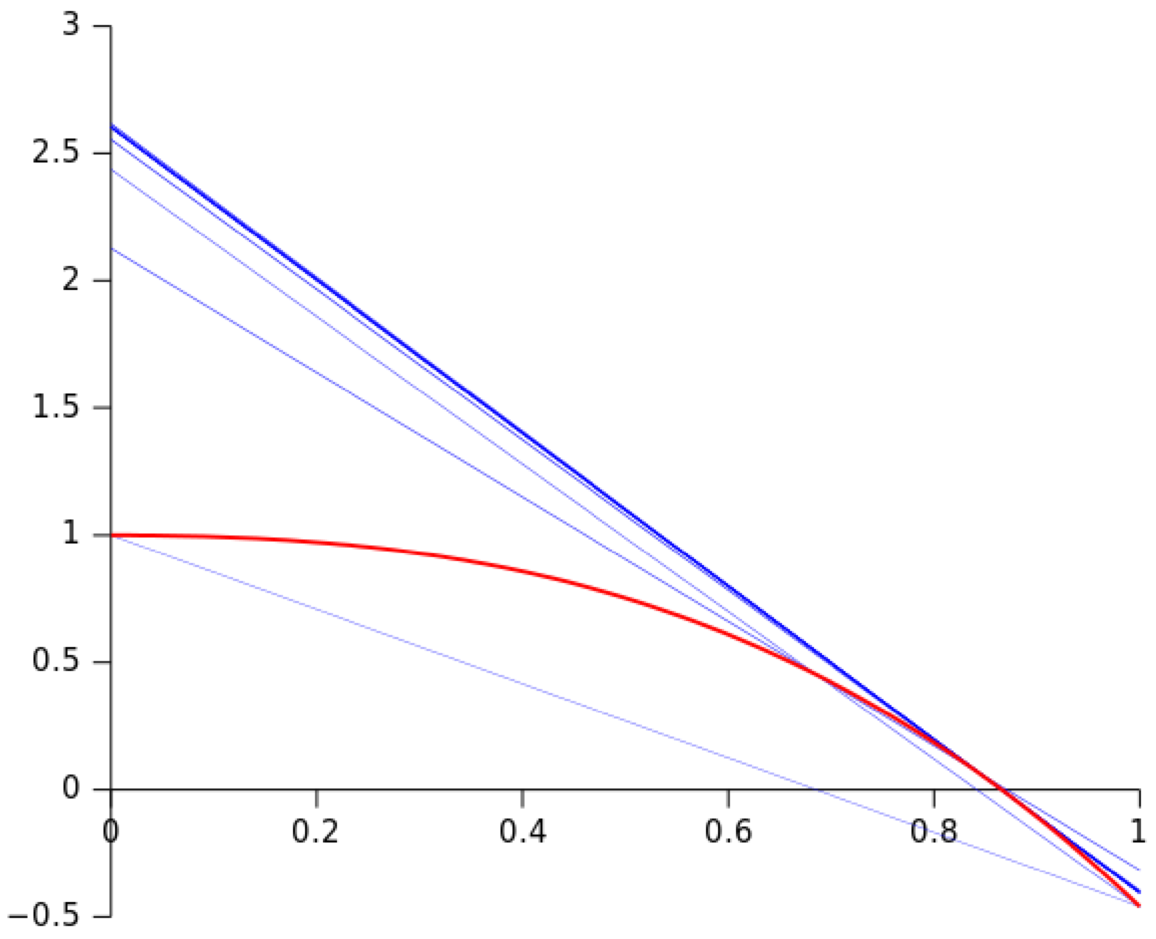
The secant method can be interpreted as a method in which the derivative is replaced by an approximation and is thus a quasi-Newton method.

If we compare Newton's method with the secant method, we see that Newton's method converges faster (order 2 against $\varphi \approx 1.6$). However, Newton's method requires the evaluation of both f and its derivative f' at every step, while the secant method only requires the evaluation of f. Therefore, the secant method may occasionally be faster in practice. For instance, if we assume that evaluating f takes as much time as evaluating its derivative and we neglect all other costs, we can do two steps of the secant method (decreasing the logarithm of the error by a factor $\varphi^2 \approx 2.6$) for the same cost as one step of Newton's method (decreasing the logarithm of the error by a factor 2), so the secant method is faster. If, however, we consider parallel processing for the evaluation of the derivative, Newton's method proves its worth, being faster in time, though still spending more steps.

Generalization [edit]

Broyden's method is a generalization of the secant method to more than one dimension.

The following graph shows the function *f* in red and the last secant line in bold blue. In the graph, the *x* intercept of the secant line seems to be a good approximation of the root of *f*.



Computational example [edit]

Below, the secant method is implemented in the Python programming language.

It is then applied to find a root of the function $f(x) = x^2 - 612$ with initial points $x_0 = 10$ and $x_1 = 30$

```python
def secant_method(f, x0, x1, iterations):
    """Return the root calculated using the secant method."""
    for i in range(iterations):
        x2 = x1 - f(x1) * (x1 - x0) / float(f(x1) - f(x0))
        x0, x1 = x1, x2
        # Apply a stopping criterion here (see below)
    return x2

def f_example(x):
    return x ** 2 - 612

root = secant_method(f_example, 10, 30, 5)

print(f"Root: {root}")  # Root: 24.738633748750722
```

It is very important to have a good stopping criterion above, otherwise, due to limited numerical precision of floating point numbers, the algorithm can return inaccurate results if running for too many iterations. For example, the loop above can stop when one of these is reached first: abs(x0 - x1) < tol, or abs(x0/x1-1) < tol, or abs(f(x1)) < tol. [2]

Notes [edit]

1. ^ Papakonstantinou, Joanna; Tapia, Richard (2013). "Origin and evolution of the secant method in one dimension". *American Mathematical Monthly*. **120** (6): 500–518. doi:10.4169/amer.math.monthly.120.06.500. JSTOR 10.4169/amer.math.monthly.120.06.500. S2CID 17645996 – via JSTOR.
2. ^ "MATLAB TUTORIAL for the First Course. Part 1.3: Secant Methods".

See also [edit]

- False position method

References [edit]

- Avriel, Mordecai (1976). *Nonlinear Programming: Analysis and Methods*. Prentice Hall. pp. 220–221. ISBN 0-13-623603-0.
- Allen, Myron B.; Isaacson, Eli L. (1998). *Numerical analysis for applied science*. John Wiley & Sons. pp. 188–195. ISBN 978-0-471-55266-6.

External links [edit]

- Secant Method Notes, PPT, Mathcad, Maple, Mathematica, Matlab at Holistic Numerical Methods Institute
- Weisstein, Eric W. "Secant Method". *MathWorld*.

Category: Quasi-Newton methods