Newton’s Method I

Roots and Stationary Points of Univariate Scalar Functions

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering
RWTH Aachen University
Outline

Objective and Learning Outcomes

Recall
  - Linearization
  - Intermediate Value Theorem

Newton’s Method
  - Roots of Nonlinear Equations
  - Stationary Points and Local Optima

Summary and Next Steps
Outline

Objective and Learning Outcomes

Recall
- Linearization
- Intermediate Value Theorem

Newton’s Method
- Roots of Nonlinear Equations
- Stationary Points and Local Optima

Summary and Next Steps
Newton’s Method I

Objective and Learning Outcomes

Objective

▶ Introduction to Newton’s method for scalar functions.

Learning Outcomes

▶ You will understand
  ▶ roots of nonlinear functions
  ▶ linearization
  ▶ convergence
  ▶ stationary points and local optima

▶ You will be able to
  ▶ implement Newton’s method
  ▶ investigate convergence of Newton’s method.
Outline

Objective and Learning Outcomes

Recall
Linearization
Intermediate Value Theorem

Newton’s Method
Roots of Nonlinear Equations
Stationary Points and Local Optima

Summary and Next Steps
Recall

Linearization

The solution of linear equations amounts to simple scalar division. The solution of nonlinear equations can be challenging.

Many numerical methods for nonlinear problems are built on local (at $\tilde{x}$) replacement of the target function with a linear (affine; in $\Delta x$) approximation derived from the truncated Taylor series expansion and “hoping” that

$$f(\tilde{x} + \Delta x) \approx f(\tilde{x}) + f'(\tilde{x}) \cdot \Delta x,$$

i.e., hoping for a reasonably small remainder.

The solution of a sequence of linear problems is then expected to yield an iterative approximation of the solution to the nonlinear problem. Newton’s method is THE example.
Recall

Intermediate Value Theorem and Bisection

Let \( y = f(x) \) be continuous within a neighborhood \( x + \Delta x \) of \( x \) taking values \( f(x) \) and \( f(x + \Delta x) \) at the endpoints of the interval. Then \( f \) takes all values between \( f(x) \) and \( f(x + \Delta x) \) over the same interval.

If \( f(x) \) and \( f(x + \Delta x) \) have different signs, then \( f \) has a root within the interval bounded by \( x \) and \( x + \Delta x \), i.e., \( \exists \tilde{x} : f(\tilde{x}) = 0 \).

The simplest possible root finding algorithm follows from iterative / recursive bisection of the the interval \([x, x + \Delta x]\).

Unfortunately, the bisection algorithm converges slowly and does not generalize to higher dimensions. Hence, we are going to investigate superior alternatives.

Note: \( \tilde{x} \) not necessarily unique; \( f \) can take values outside of \([f(x), f(x + \Delta x)]\)
Outline

Objective and Learning Outcomes

Recall
  Linearization
  Intermediate Value Theorem

Newton’s Method
  Roots of Nonlinear Equations
  Stationary Points and Local Optima

Summary and Next Steps
Consider a nonlinear equation \( y = f(x) = 0 \) at some (starting) point \( x \).

Building on the assumption that \( f(x + \Delta x) \approx f(x) + f'(x) \cdot \Delta x \) the root finding problem for \( f \) can be replaced locally by the root finding problem for the linearization

\[
\bar{f}(\Delta x) = f(x) + f'(x) \cdot \Delta x.
\]

The right-hand side is a straight line intersecting the \( y \)-axis in \((\Delta x = 0, \bar{f}(\Delta x) = f(x))\).

Solution of

\[
\bar{f}(\Delta x) = f(x) + f'(x) \cdot \Delta x = 0
\]

for \( \Delta x \) yields

\[
\Delta x = -\frac{f(x)}{f'(x)}
\]

implying \( f(x + \Delta x) \approx 0 \).
Newton’s Method
Iterative Algorithm

If the new iterate is not close enough to the root of the nonlinear function, i.e, \( |f(x + \Delta x)| > \epsilon \) for some measure of accuracy of the numerical approximation \( \epsilon > 0 \), then it becomes the starting point for the next iteration yielding the recurrence

\[
x = x - \frac{f(x)}{f'(x)}
\]

Convergence of this Newton[-Raphson] method is not guaranteed in general. Damping of the magnitude of the next step may help.

\[
x = x - \alpha \cdot \frac{f(x)}{f'(x)} \quad \text{for } 0 < \alpha \leq 1.
\]

The damping parameter \( \alpha \) is often determined by line search (e.g, recursive bisection yielding \( \alpha = 1, 0.5, 0.25, \ldots \)) such that decrease in absolute function value is ensured.
template<typename T> T f(const T &x);

template<typename T> T dfdx(const T &x);

template<typename T>
void solve(T& x, const T& eps) {
    while (fabs(f(x))>eps) x-=f(x)/dfdx(x);
}

int main(int, char **v[]) {
    double x=std::stof(v[1]);
    solve(x,1e-12);
    std::cout << "x=" << x << std::endl
    << "f(x)=" << f(x) << std::endl
    << "dfdx(x)=" << dfdx(x) << std::endl
    return 0;
}
Newton’s Method

Convergence of Fixed-Point Iteration

Newton’s method can be regarded as a fixed point iteration

\[ x = g(x) = x - \frac{f(x)}{f'(x)}. \]

If at the solution

\[ |g'(x)| < 1, \]

then there exists a neighborhood containing values of \( x \) for which the fixed-point iteration converges to this solution.

The convergence rate of a fixed-point iteration grows linearly with decreasing values of \( |g'(x)| \).

For \( |g'(x)| = 0 \) we get at least quadratic convergence; cubic for \( |g'(x)| = |g''(x)| = 0 \) and so forth.
Newton’s Method
Formulation as Fixed-Point Iteration

Newton’s method becomes

\[ x = g(x) = x - \frac{f(x)}{f'(x)} \]

yielding

\[ g'(x) = f(x) \cdot \frac{f''(x)}{(f'(x))^2} \cdot f(x) \cdot (\ldots) \]

At the solution \( f(x) = 0 \) implies \( g'(x) = 0 \). Assuming a \textbf{simple root} \( (f(x) = 0, f'(x) \neq 0) \) the second derivative of \( g \) becomes equal to

\[ g''(x) = f'(x) \cdot \frac{f''(x)}{(f'(x))^2} + f(x) \cdot (\ldots) \]

implying \textbf{quadratic convergence} within the corresponding neighborhood of the solution if \( f''(x) \neq 0 \) as well as convergence after a single iteration for linear \( f \).
Newton’s Method
Convergence (Example)

Let $f(x) = \cos(x)$. from

$$x = g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{\cos(x)}{-\sin(x)}$$

follows

$$g'(x) = f(x) \cdot \frac{f''(x)}{(f'(x))^2} g'(x) = \cos(x) \cdot \frac{-\sin(x)}{(\cos(x))^2}.$$ 

At $\tilde{x} = 1$ we get $|g'(x)| \approx 0.41$ and hence convergence to the nearest root at $\frac{\pi}{2} \approx 1.57$.

At $\tilde{x} = 3$ we get $|g'(x)| \approx 49.21$ suggesting divergence. However, the second iterate $\tilde{x} \approx -4.02$ yields $|g'(x)| \approx 0.69$ resulting in convergence to the root closest to $-4.02$, that is, $-\frac{3\pi}{2} \approx -4.71$. 

Stationary Points and Local Optima

Conditions

Let $y = f(x)$ be twice continuously differentiable over the domain of interest.

▶ $\tilde{x}$ is a stationary point (necessary condition for local optimality) of $f$ if

$$f'(\tilde{x}) \equiv \frac{df}{dx}(\tilde{x}) = 0$$

▶ $\tilde{x}$ is a local minimum (sufficient condition for local optimality) of $f$ if

$$f''(\tilde{x}) \equiv \frac{d^2f}{dx^2}(\tilde{x}) > 0 \quad \text{(strict convexity)}$$

▶ $\tilde{x}$ is a local maximum (sufficient condition for local optimality) of $f$ if

$$f''(\tilde{x}) \equiv \frac{d^2f}{dx^2}(\tilde{x}) < 0 \quad \text{(strict concavity)}$$

$f'' = 0$ indicates a non-simple stationary point.
Consider the nonlinear optimization problem

$$\min_{x \in \mathbb{R}} f(x)$$

for $f : \mathbb{R} \rightarrow \mathbb{R}$.

Starting from some initial estimate for the stationary point $\tilde{x}$ the steepest descent method iteratively takes steps into descent directions. In the scalar case the choice is between stepping toward $-\infty$ or $\infty$.

The first derivative $f'(x)$ indicates local increase ($f'(\tilde{x}) > 0$) or decrease ($f'(\tilde{x}) < 0$) of the function value.

Aiming for decrease the next step should be toward $-\infty$ if $f' > 0$ or toward $\infty$ if $f' < 0$. No further local decrease in the function value can be achieved for $f' = 0$ (necessary optimality condition).
The step size is typically damped in order to ensure continued progress toward \( \min_{x \in \mathbb{R}} f(x) \) yielding the recurrence

\[
x := x - \alpha \cdot f'(x) \quad \text{while } |f'(x)| > \epsilon
\]

Comments on line search apply as above.

Validation of a local minimum at \( \tilde{x} \) requires \( f''(\tilde{x}) > 0 \). Similarly, a local maximum is found if \( f''(\tilde{x}) < 0 \).
template<typename T> T f(const T &x);
template<typename T> T dfdx(const T &x);
template<typename T> T ddfdxx(const T &x);

template<typename T>
void solve(T& x, const T& eps) {
    T g=dfdx(x), y=f(x), y_prev;
    while (fabs(g)>eps) {
        y_prev=y;
        double alpha=2.;
        while (y_prev<=y&&alpha>eps) {
            T x_trial=x; alpha/=2;
            x_trial-=alpha*g; y=f(x_trial);
        }
        x-=alpha*g; g=dfdx(x);
    }
}
Newton’s Method
Local Optima

Consider the nonlinear optimization problem

$$\min_{x \in \mathbb{R}} f(x)$$

for $f : \mathbb{R} \to \mathbb{R}$. Application of Newton’s method to

$$f'(x) = 0$$

yields

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

Comments on convergence and line search apply as above. Validation of a local minimum at $\tilde{x}$ requires $f''(\tilde{x}) > 0$. Similarly, a local maximum is found if $f''(\tilde{x}) < 0$. 
Newton’s Method

Local Optima: Implementation

```cpp
template<typename T> T f(const T &x);
template<typename T> T dfdx(const T &x);
template<typename T> T ddfdx(const T &x);

template<typename T>
void solve(T& x, const T& eps) {
    while (fabs(dfdx(x)) > eps)
        x = dfdx(x) / ddfdx(x);
}
```
Outline

Objective and Learning Outcomes

Recall
Linearization
Intermediate Value Theorem

Newton’s Method
Roots of Nonlinear Equations
Stationary Points and Local Optima

Summary and Next Steps
Newton’s Method I
Summary and Next Steps

Summary

- Newton’s method for scalar functions
- roots of nonlinear equations
- stationary points and local optima

Next Steps

- Inspect sample code.
- Run further experiments.
- Continue the course to find out more ...