5. Root finding II

Last time

- Finding roots
- Fixed-point iteration
- Newton method

Goal for today

- Convergence of iterative methods
- Newton viewed as a fixed-point iteration
- Systems of nonlinear equations

Reminder: Existence and uniqueness of fixed point

- Argued that if $|g'(x^*)| < 1$ then $x_{n+1} = g(x_n)$ would converge to unique fixed point
- Let's see how we can show (prove) that

Reminder: Existence and uniqueness of fixed point

- Argued that if $|g'(x^*)| < 1$ then $x_{n+1} = g(x_n)$ would converge to unique fixed point
- Let's see how we can show (prove) that
- Suppose the conditions from last lecture hold:
  - g is continuous and maps $[a, b]$ into itself
  - $|g'(x)| \leq k \quad \forall x \in (a, b)$, with $k < 1$

Reminder: Existence and uniqueness of fixed point

- Argued that if $|g'(x^*)| < 1$ then $x_{n+1} = g(x_n)$ would converge to unique fixed point
- Let's see how we can show (prove) that
- Suppose the conditions from last lecture hold:
  - g is continuous and maps $[a, b]$ into itself
  - $|g'(x)| \leq k \quad \forall x \in (a, b)$, with $k < 1$
- Then there is a unique fixed point x^* of g in $[a, b]$

Convergence to fixed point

- Since x^* is a fixed point of g:

$$x_{n+1} - x^* = g(x_n) - g(x^*)$$

- Mean value theorem:

$$g(x_n) - g(x^*) = (x_n - x^*)g'(\xi)$$

- So

$$|x_{n+1} - x^*| \leq |x_n - x^*| \cdot |g'(\xi)| \leq k\,|x_n - x^*|$$

- Hence $|x_n - x^*| \leq k^n |x_0 - x^*|$
- So have linear convergence (at least)

Newton method viewed as a fixed-point iteration

- How **design** faster fixed-point iteration for $f(x) = 0$?
- Look for fixed-point algorithm with $g(x) := x - \phi(x) f(x)$
- Impose $g'(x^*) = 0$ at root, i.e. where $f(x^*) = 0$

- $g'(x) = 1 - \phi'(x)f(x) - \phi(x)f'(x)$
- So $g'(x^*) = 1 - \phi(x^*)f'(x^*) = 0$
- So need $\phi(x^*) = \frac{1}{f'(x^*)}$
- Take $\phi(x) := \frac{1}{f'(x)}$

- $g'(x) = 1 - \phi'(x)f(x) - \phi(x)f'(x)$
- So $g'(x^*) = 1 - \phi(x^*)f'(x^*) = 0$
- So need $\phi(x^*) = \frac{1}{f'(x^*)}$
- Take $\phi(x) := \frac{1}{f'(x)}$
- Get Newton method!

Order of convergence for Newton

- What is order of convergence of Newton?

Order of convergence for Newton

- What is order of convergence of Newton?
- Calculate $\delta_{n+1} := x_{n+1} - x^*$ in terms of δ_n

Order of convergence for Newton

- What is order of convergence of Newton?
- Calculate $\delta_{n+1} := x_{n+1} - x^*$ in terms of δ_n
- Obtain

$$\delta_{n+1} = -\frac{1}{2}\frac{f''(x^*)}{f'(x^*)}{\delta_n}^2 + \mathcal{O}(\delta_n^3)$$

- Derivation

Order of convergence for Newton

- What is order of convergence of Newton?
- Calculate $\delta_{n+1} := x_{n+1} - x^*$ in terms of δ_n
- Obtain

$$\delta_{n+1} = -\frac{1}{2}\frac{f''(x^*)}{f'(x^*)}{\delta_n}^2 + \mathcal{O}(\delta_n^3)$$

- Derivation
- So Newton method has **quadratic** convergence
- Assuming $f'(x^*) \neq 0$, i.e. **simple** root (not multiple root)

Difficulties with the Newton method

- What are the problems with the Newton method?
- Unknown domain where it actually converges
- Need to evaluate f'!

Difficulties with the Newton method

- What are the problems with the Newton method?
- Unknown domain where it actually converges
- Need to evaluate f'!
- Will see (later) two different methods to calculate derivatives
- Alternative: *avoid* calculating derivative

Avoiding derivatives: Secant method

- Newton: linear approximation (tangent line) of f near x_n
- Maybe we could find an *alternative* linear approximation?

Secant method II

- Suppose have *two* initial guesses, x_0 and x_1
- Take line ℓ_0 joining $(x_0, f(x_0))$ with $(x_1, f(x_1))$
- **Secant line** (from Latin: *secare* = to cut)
- Find where ℓ_0 intersects x-axis: new estimate x_2
- Repeat, joining *last two* points and **interpolating**

Secant method III

- Line joining:

$$\ell_{n-1}(x) = f(x_{n-1}) + \frac{x - x_{n-1}}{x_n - x_{n-1}} \left[f(x_n) - f(x_{n-1}) \right]$$

Secant method III

- Line joining:

$$\ell_{n-1}(x) = f(x_{n-1}) + \frac{x - x_{n-1}}{x_n - x_{n-1}} \left[f(x_n) - f(x_{n-1})\right]$$

- Solve $\ell_{n-1}(x) = 0$ to find next point x_{n+1}:
- Obtain

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

Secant method III

- Line joining:

$$\ell_{n-1}(x) = f(x_{n-1}) + \frac{x - x_{n-1}}{x_n - x_{n-1}} \left[f(x_n) - f(x_{n-1}) \right]$$

- Solve $\ell_{n-1}(x) = 0$ to find next point x_{n+1}:
- Obtain

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

- Taylor expand:

$$\delta_{n+1} \simeq -\frac{1}{2} \frac{f''(x^*)}{f'(x^*)} \delta_{n-1} \delta_n$$

Secant method IV

- Suppose has order α convergence
- Then $\delta_{n+1} \sim C_1 {\delta_n}^{\alpha}$
- So ${\delta_{n-1}}^{\alpha^2} \sim C_2 {\delta_{n-1}}^{1+\alpha}$
- So $\alpha^2 = 1 + \alpha$, giving $\alpha = \frac{1}{2}(1 + \sqrt{5}) \simeq 1.62$

Secant method IV

- Suppose has order α convergence
- Then $\delta_{n+1} \sim C_1 {\delta_n}^{\alpha}$
- So ${\delta_{n-1}}^{\alpha^2} \sim C_2 {\delta_{n-1}}^{1+\alpha}$
- So $\alpha^2 = 1 + \alpha$, giving $\alpha = \frac{1}{2}(1 + \sqrt{5}) \simeq 1.62$
- Seems to be slower than Newton, but only need *one new evaluation* of f per step
- Better convergence than Newton *per evaluation of* f (usually most expensive part)

Solving systems of equations

- How can we solve a **system** of nonlinear equations
- e.g. 2 equations in 2 unknowns:

$$f(x,y) := x^2 + y^2 - 3 = 0$$
$$g(x,y) := \left(\frac{x}{2}\right)^2 + (y-0.5)^2 - 1 = 0$$

Solving systems of equations

- How can we solve a **system** of nonlinear equations
- e.g. 2 equations in 2 unknowns:

$$f(x, y) := x^2 + y^2 - 3 = 0$$

$$g(x, y) := \left(\frac{x}{2}\right)^2 + (y - 0.5)^2 - 1 = 0$$

- What does solution set of $f(x, y) = 0$ look like?

Solving systems of equations

- How can we solve a **system** of nonlinear equations
- e.g. 2 equations in 2 unknowns:

$$f(x, y) := x^2 + y^2 - 3 = 0$$
$$g(x, y) := \left(\frac{x}{2}\right)^2 + (y - 0.5)^2 - 1 = 0$$

- What does solution set of $f(x, y) = 0$ look like?
- $f(x, y) = 0$ usually gives a **curve**
- Want *joint* roots $f(x^*, y^*) = g(x^*, y^*) = 0$
- **Intersection** points of curves: significantly harder than 1D

Solving systems of equations

- How can we solve a **system** of nonlinear equations
- e.g. 2 equations in 2 unknowns:

$$f(x, y) := x^2 + y^2 - 3 = 0$$
$$g(x, y) := \left(\frac{x}{2}\right)^2 + (y - 0.5)^2 - 1 = 0$$

- What does solution set of $f(x, y) = 0$ look like?
- $f(x, y) = 0$ usually gives a **curve**
- Want *joint* roots $f(x^*, y^*) = g(x^*, y^*) = 0$
- **Intersection** points of curves: significantly harder than 1D
- Multidimensional bisection or interval arithmetic

Vector form

- Rewrite system of equations into vector form:
- Write $f_1 = f; f_2 = g; x_1 = x; x_2 = y$
- Get system $f_i(x_1, \ldots, x_n) = 0$

Vector form

- Rewrite system of equations into vector form:
- Write $f_1 = f; f_2 = g; x_1 = x; x_2 = y$
- Get system $f_i(x_1, \dots, x_n) = 0$
- Write vectors $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{f} = (f_1, \dots, f_n)$
- Vector form: $\mathbf{f}(\mathbf{x}) = \mathbf{0}$

Multidimensional Newton

- Idea: Apply same technique as for 1D Newton

Multidimensional Newton

- Idea: Apply same technique as for 1D Newton
- Initial guess $\mathbf{x}_0$; try to solve $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) = \mathbf{0}$

Multidimensional Newton

- Idea: Apply same technique as for 1D Newton
- Initial guess $\mathbf{x}_0$; try to solve $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) = \mathbf{0}$
- Need **Taylor expansion for higher dimensions**:

$$\mathbf{f}(\mathbf{a} + \boldsymbol{\delta}) = \mathbf{f}(\mathbf{a}) + \mathsf{D}\mathbf{f}(\mathbf{a}) \cdot \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2)$$

Multidimensional Newton

- Idea: Apply same technique as for 1D Newton
- Initial guess $\mathbf{x}_0$; try to solve $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) = \mathbf{0}$
- Need **Taylor expansion for higher dimensions**:

$$\mathbf{f}(\mathbf{a} + \boldsymbol{\delta}) = \mathbf{f}(\mathbf{a}) + \mathsf{D}\mathbf{f}(\mathbf{a}) \cdot \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2)$$

- So $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) \simeq \mathbf{f}(\mathbf{x}_n) + \mathsf{J}(\mathbf{x}_n) \cdot \boldsymbol{\delta}_n$

Multidimensional Newton

- Idea: Apply same technique as for 1D Newton
- Initial guess $\mathbf{x}_0$; try to solve $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) = \mathbf{0}$
- Need **Taylor expansion for higher dimensions**:

$$\mathbf{f}(\mathbf{a} + \boldsymbol{\delta}) = \mathbf{f}(\mathbf{a}) + \mathsf{D}\mathbf{f}(\mathbf{a}) \cdot \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2)$$

- So $\mathbf{f}(\mathbf{x}_n + \boldsymbol{\delta}) \simeq \mathbf{f}(\mathbf{x}_n) + \mathsf{J}(\mathbf{x}_n) \cdot \boldsymbol{\delta}_n$
- Need to solve $\boldsymbol{\delta}_n = -\mathsf{J}(\mathbf{x}_n)^{-1}\mathbf{f}(\mathbf{x}_n)$
- $\mathsf{J} := \mathsf{D}\mathbf{f}(\mathbf{x}_n)$ is **Jacobian matrix** of all partial derivatives

Multidimensional Newton II

- Reduced nonlinear system to **linear system**
- Mathematics: calculate **matrix inverse**

Multidimensional Newton II

- Reduced nonlinear system to **linear system**
- Mathematics: calculate **matrix inverse**
- Numerics: instead **solve linear system** (see later)

Multidimensional Newton II

- Reduced nonlinear system to **linear system**
- Mathematics: calculate **matrix inverse**
- Numerics: instead **solve linear system** (see later)
- For now, use Julia's linear system solver, written \ ("backslash")
- "Magic" black box
- Type of "matrix division"

Solving linear systems in Julia

- To solve linear system $A \cdot \mathbf{x} = \mathbf{b}$ in Julia:

```
using LinearAlgebra    # standard library; no installation re

A = rand(2, 2)    # random matrix
b = rand(2)       # random vector

x = A \ b

residual = (A * x) - b
```

- `A * x` is standard matrix–vector multiplication

Summary

- Proved convergence of iterative methods
- Viewed Newton method as a fixed-point iteration
- Secant method to avoid calculating derivative ("derivative-free")
- Newton method in higher dimensions