

HT2026 Continuous Maths notes

Remaining **TODOs**: 44

Contents

1. Derivatives and Taylor's Theorem	2
1.1. Continuity and differentiation	2
1.2. Taylor's theorem for univariate functions	3
1.3. Partial and vector derivatives	4
1.4. Vector fields and the Jacobian	5
1.5. Taylor's theorem for multivariate functions	7
2. Optimisation	9
2.1. Optimisation in 1 dimension	9
2.2. Positive/negative (semi)definiteness	10
2.3. Unconstrained optimisation over \mathbb{R}^n	12
2.4. Convexity	12
2.5. Optimisation tricks	15
2.6. Optimisation over \mathbb{R}^n with equality constraints	15
2.7. Inequality constrained optimisation (over \mathbb{R}^n)	16
3. Algorithms for numerical integration	18
3.1. 1-dimensional integration	18
3.2. Integration in d dimensions	21
4. Accuracy	27
4.1. Iterative methods	28
5. Algorithms for root-finding	30
5.1. 1-dimensional root-finding	30
5.2. d -dimensional root finding	35
6. Algorithms for optimisation	37
6.1. 1-dimensional minimisation	37
6.2. d -dimensional minimisation	39
Index	43

1. Derivatives and Taylor's Theorem

1.1. Continuity and differentiation

Definition 1.1.1: Let $f : D \rightarrow \mathbb{R}, D \subseteq \mathbb{R}$.

f is **continuous** at x if $\lim_{h \rightarrow 0} f(x+h) = f(x)$.

f is continuous if it is continuous at every $x \in D$.

Definition 1.1.2: Let $f : D \rightarrow \mathbb{R}, D \subseteq \mathbb{R}^n$.

f is continuous at \mathbf{x} if $\lim_{\mathbf{h} \rightarrow 0} f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x})$.

f is continuous if it is continuous at every $\mathbf{x} \in D$.

Theorem 1.1.3: If f and g are continuous, so are:

- $f + g$
- $cf \quad \forall c \in \mathbb{R}$
- $f^n \quad \forall n \in \mathbb{N}$
- fg
- $f \circ g$
- $\max(f, g)$
- $|f|$
- $\exp(f)$
- $f^\alpha \quad \forall \alpha \in \mathbb{R}$, when f strictly positive
- $\log f$, where f strictly positive
- $\frac{f}{g}$, where g nonzero

Definition 1.1.4: Let $f : D \rightarrow \mathbb{R}, D \subseteq \mathbb{R}$.

f is **differentiable** at x if $\lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} = d$ exists. Then d is the **derivative** of f at x .

f is differentiable if it is differentiable at every $x \in D$.

Remark: If a function is not continuous, it is not differentiable.

Definition 1.1.5: A **secant** is a line that connects two points on a surface.

A **tangent** can be thought of as a limit of secants as the distance between the two points tend to 0.

Remark: If f and g are differentiable, so are all the combinations that preserve continuity mentioned earlier, with the exception of $\max(f, g)$ and $|f|$.

Remark: The derivative is a linear operator.

Theorem 1.1.6 (Quotient rule):

$$\frac{d}{dx} \frac{f}{g} = \frac{g \frac{df}{dx} - f \frac{dg}{dx}}{g^2}.$$

Theorem 1.1.7 (Derived quotient rule):

$$\frac{d}{dx} \left(\frac{f}{g^k} \right) = \frac{g \frac{df}{dx} - k f \frac{dg}{dx}}{g^{k+1}}.$$

Proof: Exercise; see problem sheet 0

□

1.2. Taylor's theorem for univariate functions

Theorem 1.2.1 (Taylor's Theorem (univariate)): For $f : \mathbb{R} \rightarrow \mathbb{R}$, $x_0 \in \mathbb{R}$,

$$f(x) = \underbrace{\sum_{i=0}^k \frac{(x-x_0)^i}{i!} \frac{d^i f}{dx^i}(x_0)}_{\text{Taylor polynomial}} + \underbrace{\frac{(x-x_0)^{k+1}}{(k+1)!} \frac{d^{k+1} f}{dx^{k+1}}(\xi)}_{\text{Error term}}$$

for some $\xi \in (x_0, x)$.

The **Taylor polynomial** (of degree k) is also written $\hat{f}_k(x)$, and the **error term**, or **Lagrange remainder term** is denoted by $e_{k+1}(x, x_0)$.

So, we can write $f(x) = \hat{f}_k(x) + e_{k+1}(x, x_0)$ for any $k \in \mathbb{N}$ so long as the first $k+1$ derivatives of f all exist and are continuous on (x_0, x) .

For small $e_{k+1}(x, x_0)$, $f(x) \approx \hat{f}_k(x)$.

Remark: $\hat{f}_k(x)$ is the unique k -order polynomial that agrees with f as to its first k derivatives at x_0 .

Remark: Taylor's Theorem is **not** the same as a Taylor series; the Taylor series is an infinite sum and does not always converge to the correct answer. (If $e_{k+1}(x, x_0)$ tends to 0 for large k and small enough $(x_0 - x)$, the series does converge and that function is **analytic**).

1.3. Partial and vector derivatives

Definition 1.3.1: Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$.

$\frac{\partial f}{\partial x}$ is the **partial derivative** of $f(x, y)$ with respect to x , as if y were constant.

$\frac{\partial^2 f}{\partial y \partial x}$ means differentiate first with respect to x , then y . In general this is not equal to $\frac{\partial^2 f}{\partial x \partial y}$, but often it is.

Definition 1.3.2: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Then the **vector derivative** is

$$\frac{df}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

It turns out that this makes sense to be a derivative, because

$$\lim_{\mathbf{h} \rightarrow \mathbf{0}} \frac{f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x}) - \left(\frac{df}{d\mathbf{x}}\right)^\top \mathbf{h}}{\|\mathbf{h}\|} = 0.$$

Remark: Some standard derivatives for vectors that are analogous to scalars:

- $\frac{d}{d\mathbf{x}} \mathbf{a}^\top \mathbf{x} = \mathbf{a}$
- $\frac{d}{d\mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{x} = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x}$
- $\frac{d}{d\mathbf{x}} \|\mathbf{x}\| = \frac{\mathbf{x}}{\|\mathbf{x}\|}$

Remark: Similarly to scalar derivatives, the vector derivative is linear and has a product and quotient rule.

The chain rule for $f \circ g, f : \mathbb{R}^n \rightarrow \mathbb{R}, g : \mathbb{R} \rightarrow \mathbb{R}$ is the same as for scalars:

$$\frac{d}{d\mathbf{x}}(g \circ f) = \left(\frac{dg}{dx} \circ f\right) \frac{df}{d\mathbf{x}}.$$

For $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n, g : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\frac{d}{d\mathbf{x}}(g \circ \mathbf{f}) = \mathbf{J}(\mathbf{f})^\top \left(\frac{dg}{dx} \circ \mathbf{f}\right)$$

where \mathbf{J} is the Jacobean - see Definition 1.4.2.

Remark: The type of the vector derivative of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is

$$\frac{df}{d\mathbf{x}} : \mathbb{R}^n \rightarrow \mathbb{R}^n.$$

Also

$$\frac{d}{d\mathbf{x}} : (\mathbb{R}^n \rightarrow \mathbb{R}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R}^n).$$

Definition 1.3.3: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Then the **Hessian** of f is

$$\mathbf{H}(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

Remark: The Hessian is analogous to the second derivative of a scalar function.

Theorem 1.3.4 (Clairaut's theorem): If everything is continuous, $\mathbf{H}(f) = \mathbf{H}(f)^\top$.

1.4. Vector fields and the Jacobian

Definition 1.4.1: A **vector field** or **vector-valued function** is a function $\mathbf{f} : D \rightarrow \mathbb{R}^m$ for $D \subseteq \mathbb{R}^n$.

Remark: A vector field \mathbf{f} can be decomposed into

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix}.$$

Definition 1.4.2: The derivative of a vector field $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is the **Jacobian**, $\mathbf{J}(\mathbf{f}) : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times n}$, where

$$\mathbf{J}(f) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

Remark: For $f : \mathbb{R}^n \rightarrow \mathbb{R}$,

$$\mathbf{J}(f) = \left(\frac{df}{dx} \right)^\top.$$

Remark: The Jacobean in a linear operator, i.e. $\mathbf{J}(\mathbf{A}f) = \mathbf{A}\mathbf{J}(f)$ for constant \mathbf{A} .

Definition 1.4.3: The Jacobian has a **dot-product rule**:

$$\mathbf{J}(f^\top g) = g^\top \mathbf{J}(f) + f^\top \mathbf{J}(g).$$

Definition 1.4.4: The Jacobian has a **scalar-vector product** rule:

$$\mathbf{J}(fg) = g \frac{df^\top}{dx} + f \mathbf{J}(g).$$

Definition 1.4.5: The Jacobian has a chain rule, called the **partial chain rule**:

$$\mathbf{J}(g \circ f) = (\mathbf{J}(g) \circ f) \mathbf{J}(f).$$

Remark: Some standard Jacobians:

$$\begin{aligned} \mathbf{J}(x) &= I; \\ \mathbf{J}(\mathbf{A}x) &= \mathbf{A}. \end{aligned}$$

Remark: The Hessian is a Jacobean:

$$\mathbf{H}(f) = \mathbf{J} \left(\frac{df}{dx} \right)^\top.$$

Luckily because of Clairaut's Theorem, the transpose can usually be ignored.

Note that this basically just says that the second derivative is the derivative of the derivative.

Remark: Given vectors \mathbf{x}_i ,

$$\sum \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X},$$

where the rows of \mathbf{X} are the \mathbf{x}_i^\top s.

Moreover, given scalars d_i ,

$$\sum d_i \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{D} \mathbf{X},$$

with \mathbf{X} as before and \mathbf{D} a diagonal matrix with the d_i s as the diagonal entries.

These identities can come in handy when computing the Jacobean.

1.5. Taylor's theorem for multivariate functions

Theorem 1.5.1 (Taylor's Theorem for multivariate functions): For $f : \mathbb{R}^n \rightarrow \mathbb{R}, k \in \mathbb{N}$, if $\mathbf{x} = \mathbf{x}_0 + \mathbf{h}$,

$$f(\mathbf{x}) = \left. \begin{aligned} & f(\mathbf{x}_0) \\ & + \left[h_1 \frac{\partial}{\partial x_1}(\mathbf{x}_0) + \dots + h_n \frac{\partial}{\partial x_n} f \right](\mathbf{x}_0) \\ & + \frac{1}{2!} \left[\left(h_1 \frac{\partial}{\partial x_1} + \dots + h_n \frac{\partial}{\partial x_n} \right)^2 f \right](\mathbf{x}_0) \\ & + \dots \\ & + \frac{1}{k!} \left[\left(h_1 \frac{\partial}{\partial x_1} + \dots + h_n \frac{\partial}{\partial x_n} \right)^k f \right](\mathbf{x}_0) \end{aligned} \right\} k \text{th order Taylor polynomial, } \hat{f}_k$$

$$+ \underbrace{\frac{1}{(k+1)!} \left[\left(h_1 \frac{\partial}{\partial x_1} + \dots + h_n \frac{\partial}{\partial x_n} \right)^{k+1} f \right](\mathbf{x}_0 + \xi \mathbf{h})}_{\text{Error term}} \text{ for some } \xi \in (0, 1).$$

Here, the brackets with partial derivatives inside are carrying out operator algebra - they are not multiplication, but follow the same distributive rules so can be thought of as so.

The Taylor polynomials up to degree 2 can be written a bit more nicely:

$$\begin{aligned}\hat{f}_0(\mathbf{x}) &= f(\mathbf{x}_0); \\ \hat{f}_1(\mathbf{x}) &= f(\mathbf{x}_0) + \mathbf{h}^\top \frac{df}{d\mathbf{x}}(\mathbf{x}_0); \\ \hat{f}_2(\mathbf{x}) &= f(\mathbf{x}_0) + \mathbf{h}^\top \frac{df}{d\mathbf{x}}(\mathbf{x}_0) + \frac{1}{2} \mathbf{h}^\top \mathbf{H}(f)(\mathbf{x}_0) \mathbf{h}.\end{aligned}$$

For higher powers, we need tensors which are beyond the scope of this course.

The error terms (up to $k = 2$) can be written in a similar fashion but evaluated at $\mathbf{x}_0 + \xi \mathbf{h}$ for some $\xi \in (0, 1)$.

Theorem 1.5.2: $\text{err}_{k+1}(\mathbf{x}, \mathbf{x}_0)$ is small in the sense that

$$\lim_{\|\mathbf{x} - \mathbf{x}_0\| \rightarrow 0} \frac{\text{err}_{k+1}(\mathbf{x}, \mathbf{x}_0)}{\|\mathbf{x} - \mathbf{x}_0\|^k} = 0.$$

(Proof not needed.)

2. Optimisation

Definition 2.1: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

The canonical optimisation problem is to find

$$\min_{x \in F} f(x) \text{ or } \arg \min_{x \in F} f(x)$$

.

Note that $\arg \min$ is a set.

$F \subset \mathbb{R}^n$ is the **feasible region**.

If $F = \mathbb{R}^n$, the optimisation is **unconstrained**.

If $F \subset \mathbb{R}^n$, the optimisation is **constrained**. The standard form for constraints is:

$$F = \left\{ x \in \mathbb{R}^n \mid \underbrace{g_1(x) = 0, \dots, g_l(x) = 0}_{\text{equality constraints}}, \underbrace{h_1(x) \geq 0, \dots, h_m(x) \geq 0}_{\text{inequality constraints}} \right\}.$$

Definition 2.2: If F is empty, there is no solution and the optimisation problem is **inconsistent**.

Remark: The problem can also have no solution if, in the feasible region, the function tends to $-\infty$ or there is a vertical asymptote.

2.1. Optimisation in 1 dimension

Solutions to $\arg \min_{x \in \mathbb{R}} f(x)$ satisfy the 1st-order condition $\frac{df}{dx} = 0$. But so do non-solutions!

Stationary points can be classified by Taylor's theorem.

If $\frac{df}{dx}(x_0) = 0$, then $f(x) = f(x_0) + \frac{x-x_0}{2} \frac{d^2f}{dx^2}(\xi)$ for some $\xi \in (x_0, x)$. So if the second derivative is positive, then for some region around ξ , $f(x) > f(x_0)$ around x_0 , so x is a minimum. Similarly, if the second derivative is negative, then $f(x) < f(x_0)$ around x_0 so x is a maximum.

If the second derivative is 0, this doesn't necessarily mean that it is a point of inflection; we need to look for the first nonzero derivative. If the first nonzero derivative is odd, then it is a stationary point of inflection - since the function changes concavity around the stationary point - but if the first nonzero derivative is an even derivative, use the same rules as for the second derivative.

However, this doesn't always work (e.g. $f(x) = e^{-\frac{1}{x^2}}$, $f(0) = 0$ has all of its derivatives at 0 equal to 0).

For constrained optimisations, do the same but also consider the endpoints.

2.2. Positive/negative (semi)definiteness

Definition 2.2.1: Given a symmetric matrix \mathbf{A} , \mathbf{A} is

- **positive definite** if $\mathbf{x}^\top \mathbf{A} \mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$
- **positive semidefinite** if $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0$ for all $\mathbf{x} \neq \mathbf{0}$
- **negative definite** if $\mathbf{x}^\top \mathbf{A} \mathbf{x} < 0$ for all $\mathbf{x} \neq \mathbf{0}$
- **negative semidefinite** if $\mathbf{x}^\top \mathbf{A} \mathbf{x} \leq 0$ for all $\mathbf{x} \neq \mathbf{0}$
- **indefinite** otherwise

Theorem 2.2.2 (Spectral theorem): If \mathbf{A} is an $n \times n$ symmetric matrix, then there exists a basis (for \mathbb{R}^n) of orthogonal eigenvectors of \mathbf{A} .

Proof: See LA. □

Proposition 2.2.3:

These are equivalent to certain conditions on the eigenvalues: positive definite iff all eigenvalues strictly positive, positive semidefinite iff all eigenvalues positive, etc.

Proof (for positive definiteness):

Let \mathbf{A} be a symmetric matrix. By the spectral theorem, there exists an orthogonal basis of \mathbb{R}^n $\langle \mathbf{x}_1, \dots, \mathbf{x}_n \rangle$; w.l.o.g. let the \mathbf{x}_i s be unit vectors. Let $\lambda_1, \dots, \lambda_n$ be the corresponding eigenvalues.

First suppose that $\lambda_1, \dots, \lambda_n > 0$. By the spectral theorem, $\langle \mathbf{x}_1, \dots, \mathbf{x}_n \rangle$ forms an orthogonal basis for \mathbb{R}^n , so for any $\mathbf{v} \in \mathbb{R}^n$, $\exists c_1, \dots, c_n \in \mathbb{R}$ s.t. $\mathbf{v} = c_1 \mathbf{x}_1 + \dots + c_n \mathbf{x}_n$.

Then for any $\mathbf{v} \neq \mathbf{0}$,

$$\begin{aligned} \mathbf{v}^\top \mathbf{A} \mathbf{v} &= c_1^2 \mathbf{x}_1^\top \mathbf{A} \mathbf{x}_1 + \dots + c_n^2 \mathbf{x}_n^\top \mathbf{A} \mathbf{x}_n \text{ by distributivity of mat-vec mult.} \\ &= c_1^2 \lambda_1 \mathbf{x}_1^\top \mathbf{x}_1 + \dots + c_n^2 \lambda_n \mathbf{x}_n^\top \mathbf{x}_n \\ &= c_1^2 \lambda_1 + \dots + c_n^2 \lambda_n \text{ because the } \mathbf{x}_i \text{ s unit vectors.} \end{aligned}$$

Since $(*)^2 \geq 0$ and the λ_i s are strictly positive, and there exists at least one non-zero c_i , the sum of the products of positive numbers must be positive, so $\mathbf{v}^\top \mathbf{A} \mathbf{v}$ is positive definite.

Conversely, suppose that \mathbf{A} is positive definite.

Then, for all $1 \leq i \leq n$,

$$\begin{aligned} \mathbf{x}_i^\top \mathbf{A} \mathbf{x}_i &> 0 \text{ because } \mathbf{x}_i \neq \mathbf{0} \text{ and } \mathbf{A} \text{ positive definite} \\ \implies \lambda_i \mathbf{x}_i^\top \mathbf{x}_i &> 0 \\ \implies \lambda_i &> 0 \text{ because } \mathbf{x}_i \text{ unit vector.} \end{aligned}$$

□

Remark:

Given a symmetric 2×2 matrix $A := \begin{pmatrix} a & b \\ b & c \end{pmatrix}$ with eigenvalues λ_1, λ_2 ,

$$\lambda_1 \lambda_2 = ac - b^2.$$

Hence

- $ac - b^2 < 0 \implies \mathbf{A}$ indefinite;
- $ac - b^2 > 0 \wedge a > 0 \implies \mathbf{A}$ positive definite;
- $ac - b^2 = 0 \wedge a + c \geq 0 \implies \mathbf{A}$ positive semidefinite
- $ac - b^2 > 0 \wedge a < 0 \implies \mathbf{A}$ negative definite;
- $ac - b^2 = 0 \wedge a + c \leq 0 \implies \mathbf{A}$ negative semidefinite.

Definition 2.2.4: The pivots of a matrix are the entries on the diagonal when in echelon form, obtained without row multiplication or row swaps.

Remark: A symmetric matrix \mathbf{A} is

- positive definite if all its pivots > 0
- positive semidefinite if all its pivots are ≥ 0
- negative definite if all its pivots < 0
- negative semidefinite if all its pivots ≤ 0
- indefinite otherwise

but the definiteness conditions hold only if the pivots can be found without row swaps

Remark: Definiteness works a bit like signs for scalars. Pos def equivalent to positive, pos semidef equivalent to nonnegative, etc. Incl. $(\text{pos def})^{-1}$ pos def (and this always exists, in this case).

Remark: If \mathbf{A} is positive (semi)definite then so are

- $c\mathbf{A}$ for any $c > 0$
- \mathbf{A}^{-1} , which is guaranteed to exist if \mathbf{A} is positive definite
- \mathbf{ABA} for any positive (semi)definite B
- any upper-left submatrix of \mathbf{A} **TODO** intuition
- $\mathbf{C}^\top \mathbf{A} \mathbf{C}$, if \mathbf{C} is of full rank, regardless of if \mathbf{C} is square or not - but if c not of full rank, we can still guarantee positive semidefiniteness **TODO** intuition
- $\mathbf{A} - \mathbf{v}\mathbf{v}^\top$, if $\mathbf{v}^\top \mathbf{A}^{-1} \mathbf{v} < 1$

Remark: For any symmetric matrix \mathbf{C} , $\mathbf{C} - \lambda\mathbf{I}$ is positive semidefinite if $\lambda \leq$ the smallest eigenvalue of \mathbf{C} . It is positive definite if the inequality is strict.

Remark: The zero matrix is both positive semidefinite and negative semidefinite.

TODO: lookup: diagonally dominant

2.3. Unconstrained optimisation over \mathbb{R}^n

Solutions to $\arg \min_{x \in \mathbb{R}^n}$, if they exist, satisfy the 1st-order condition $\frac{df}{dx} = \mathbf{0}$. This is in general not easy to solve as it is a system of n (not necessarily linear) equations.

In higher dimensions, stationary points can be local minima, local maxima or saddle points.

Definition 2.3.1: A **saddle point** is a stationary point at which moving in some directions gives a more positive result, and in another direction decreases.

We can classify these higher-dimensional stationary points using the Hessian, similarly to how we used the second derivative for scalar functions.

Theorem 2.3.2: If $\mathbf{H}(f)(\mathbf{x})$ is positive definite, then $\mathbf{H}(f)(\mathbf{x}^*)$ is positive definite for \mathbf{x}^* close to \mathbf{x} .

Hence if the Hessian at \mathbf{x} is positive definite, the stationary point at \mathbf{x} is a local minimum. If the Hessian is negative definite, the stationary point is a local maximum. If the Hessian is indefinite, the stationary point is a saddle point.

If the Hessian is semidefinite, you cannot determine anything other than that the point is **not** a local minimum/maximum. For better analysis we would need to use the 3rd-order Taylor approximation.

Fact: the outer product of a vector with itself is always positive semidefinite. Proof: $\mathbf{v}^\top (\mathbf{b}\mathbf{b}^\top) \mathbf{v} = (\mathbf{v}^\top \mathbf{b})^2 \geq 0$.

2.4. Convexity

Definition 2.4.1: A set $D \subseteq \mathbb{R}^n$ is **convex** if

$$\begin{aligned} \forall \mathbf{x}_1, \mathbf{x}_2 \in D, \alpha \in (0, 1), \\ (1 - \alpha)\mathbf{x}_1 + \alpha\mathbf{x}_2 \in D. \end{aligned}$$

That is, the line connecting any $\mathbf{x}_1, \mathbf{x}_2 \in D$ lies wholly in D .

Definition 2.4.2: A function $f : D \rightarrow \mathbb{R}$ for convex $D \subseteq \mathbb{R}^n$ is **convex** if

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in D, \alpha \in (0, 1), \\ f((1 - \alpha)\mathbf{x}_1 + \alpha\mathbf{x}_2) \leq (1 - \alpha)f(\mathbf{x}_1) + \alpha f(\mathbf{x}_2).$$

That is, any point on f between \mathbf{x}_1 and \mathbf{x}_2 lies below the secant between \mathbf{x}_1 and \mathbf{x}_2 .

Equivalently, every secant on f lies above the surface of f .

If the inequality is strict, f is **strictly convex** if the inequality is the other way around, f is **concave** or **strictly concave** in a similar manner.

Theorem 2.4.3: If f is convex, every stationary point is a global minimum.

Theorem 2.4.4: If f is strictly convex, there is at most one stationary point, and that is the global minimum.

Remark: If carrying out unconstrained optimisation on a convex function, just solve the first-order condition (once you have established that there is a minimum).

Remark: In one dimension, f is convex if $\frac{d^2f}{dx^2} \geq 0$ everywhere; if the inequality is strict then f is strictly convex. The implication does not hold in reverse.

Remark: Convexity does not imply the existence of a minimum!

Remark: For higher dimensions, f is convex iff $\mathbf{H}(f)$ is positive semidefinite everywhere; if (but not only if) $\mathbf{H}(f)$ is positive definite everywhere, then f is strictly convex.

Remark:

Some common convex and concave functions:

- Linear functions are both convex and concave.
- $|x|^p$ for $p > 1$ is strictly convex
- e^x is strictly convex

- $\log x$ is strictly concave
- for convex f , $-f$ is concave
- $\mathbf{x} \mapsto \mathbf{x}^\top \mathbf{A} \mathbf{x}$ is convex if \mathbf{A} is positive semidefinite, strictly convex if \mathbf{A} is positive definite
- for convex f, g , $f + g$ is convex
- for convex f , $f(\mathbf{A}\mathbf{x} + \mathbf{b})$ is convex (but strict convexity is only preserved if \mathbf{A} has full rank)
- for convex f, g , $\max(f, g)$ is convex
- for convex f , $\exp(f)$ is convex

Theorem 2.4.5: For $f : \mathbb{R} \rightarrow \mathbb{R}, g : \mathbb{R} \rightarrow \mathbb{R}$,

$$f \text{ convex} \wedge \left\{ \begin{array}{l} f \text{ increasing} \wedge g \text{ convex} \\ f \text{ decreasing} \wedge g \text{ concave} \end{array} \right\} \implies f \circ g \text{ convex};$$

$$f \text{ concave} \wedge \left\{ \begin{array}{l} f \text{ increasing} \wedge g \text{ concave} \\ f \text{ decreasing} \wedge g \text{ convex} \end{array} \right\} \implies f \circ g \text{ concave}.$$

Proof: $f \circ g$ convex iff $(f \circ g)'' \geq 0$ everywhere.

$$\frac{d}{dx}(f \circ g) = \left(\frac{df}{dx}(g(x)) \right) \cdot \frac{dg}{dx}.$$

$$\frac{d^2}{dx^2}(f \circ g) = \left(\frac{d^2f}{dx^2}(g(x)) \right) \cdot \left(\frac{dg}{dx} \right)^2 + \frac{df}{dx}(g(x)) \frac{d^2g}{dx^2}.$$

$\left(\frac{dg}{dx} \right)^2$ is always positive, so we require $\frac{d^2f}{dx^2}$ to be positive everywhere, i.e. f is convex. We then require $\frac{df}{dx}$ and $\frac{d^2g}{dx^2}$ to have the same signs everywhere, so either f is increasing and g is convex, or f is decreasing and g is concave.

The proof for concavity is similar. □

Remark: These conditions are sufficient but not necessary.

Remark: Theorem 2.4.5 can be used also for $f : \mathbb{R} \rightarrow \mathbb{R}, g : \mathbb{R}^n \rightarrow \mathbb{R}$.

Theorem 2.4.6:

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}, g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ such that

$$\mathbf{g} := \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix}$$

with $g_1, \dots, g_n : \mathbb{R}^m \rightarrow \mathbb{R}$.

Then if f is (strictly) convex, and for each $i \in \{1, \dots, n\}$, either

$$\begin{cases} f \text{ (strictly) increasing in its } i\text{th argument} \wedge g_i \text{ (strictly) convex, or} \\ f \text{ (strictly) decreasing in its } i\text{th argument} \wedge g_i \text{ (strictly) concave,} \end{cases}$$

then $f \circ \mathbf{g}$ is (strictly) convex.

Theorem 2.4.7 (Jenson's inequality): For a random variable X and convex function f ,

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(x)].$$

(This is not on the syllabus but is useful)

2.5. Optimisation tricks

Theorem 2.5.1: If g is strictly increasing, then

$$\arg \min f(x) = \arg \min g(f(x)).$$

Theorem 2.5.2: If g is injective/1-to-1 then

$$\arg \min f(x) = g(\arg \min f(g(x))).$$

2.6. Optimisation over \mathbb{R}^n with equality constraints

Remark: Given a minimisation problem with an equality constraint, it can sometimes be reduced to an unconstrained problem:

- Given $g(x, y) = x + y + 1$,

$$\min_{x \in \mathbb{R}^2 \mid g(x)=0} f(x) = \min_{x \in \mathbb{R}} f(x, -1 - x)$$

- Given $g(x, y) = x^2 + y^2 - 1$,

$$\min_{x \in \mathbb{R}^2 \mid g(x)=0} f(x_1, x_2) = \min_{\theta \in (0, 2\pi]} f(\sin \theta, \cos \theta)$$

But usually, this isn't the case.

Theorem 2.6.1 (Lagrange's theorem / method of Lagrange multipliers): For a minimisation problem with one equality constraint, minimise $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$, the stationary points must satisfy

$$\frac{df}{d\mathbf{x}} = \lambda \frac{dg}{d\mathbf{x}},$$

i.e. a first order condition is $\frac{d}{d\mathbf{x}}(f(\mathbf{x}) - \lambda g(\mathbf{x})) = \mathbf{0}$.

Proof:

TODO

□

Definition 2.6.2: The **Lagrangian** of such a minimisation problem is

$$\Lambda(\lambda, \mathbf{x}) = f(\mathbf{x}) - \lambda g(\mathbf{x}).$$

The stationary points of $f(\mathbf{x})$ where $g(\mathbf{x}) = \mathbf{0}$ are the stationary points of Λ .

Remark: It is not simple to determine if the stationary points of the Lagrangian are a minimum, maximum or saddle point of the problem. It is not sufficient to look at the definiteness of the Hessian of Λ , or the convexity of f .

TODO: look up bordered Hessian

Theorem 2.6.3: For $\min f(\mathbf{x})$ s.t. $g_1(\mathbf{x}) = 0, \dots, g_l(\mathbf{x}) = 0$,

the Lagrangian is

$$\Lambda(\boldsymbol{\lambda}, \mathbf{x}) = f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - \dots - \lambda_l g_l(\mathbf{x}).$$

Then the stationary points of the minimisation problem are the stationary points of the Lagrangian.

2.7. Inequality constrained optimisation (over \mathbb{R}^n)

Definition 2.7.1: Given a minimisation problem $\min f(\mathbf{x})$ subject to $h(\mathbf{x}) \geq 0$, at the solution, h is **tight** if $h(\mathbf{x}) = 0$, and **slack** if $h(\mathbf{x}) > 0$.

To solve such a minimisation, solve:

- for tight constraints via the previous section;
- for slack constraints, by solving the usual first-order condition $\frac{df}{d\mathbf{x}} = \mathbf{0}$ (as usual), and ignoring any solutions outside of the feasible region.

Either way, $\frac{df}{d\mathbf{x}} = \mu \frac{dh}{d\mathbf{x}}$, with μ possibly zero.

Theorem 2.7.2: The Lagrange multiplier is always positive.

Proof: **TODO**

□

Corollary 2.7.3 (Complementary slackness): At a minimum of an minimisation problem with constraint $h(\mathbf{x}) = 0$ and Lagrange multiplier μ ,

$$[\mu \geq 0 \wedge h(\mathbf{x}) = 0] \vee [\mu = 0 \wedge h(\mathbf{x}) > 0].$$

Equivalently, $\mu h(\mathbf{x}) = 0$.

Theorem 2.7.4: When optimising f with one inequality constraint, if f is convex and the feasible set is convex, then either every stationary point inside the constraint is a global minimum, or one of the stationary points on the constraint is a global minimum.

TODO KKT conditions (not on syllabus)

TODO: what to do when mix of equality and inequality constraints

3. Algorithms for numerical integration

3.1. 1-dimensional integration

Here we will consider approximating the integral of f over a small interval $[a, b]$, with $m = \frac{a+b}{2}$.

The simplest approximation is the 0th order Taylor approximation; this is equal to the approximation using the 1st order Taylor approximation.

Definition 3.1.1: The **midpoint rule**: define

$$M_1[f, a, b] := \int_a^b \hat{f}_1(x) dx = (b-a)f(m).$$

The subscript 1 indicates that it is operating over 1 strip. The notation for this (and other methods in this section) is not standard.

Theorem 3.1.2: The error of the midpoint rule,

$$\text{err}(M_1)[f, a, b] = M_1[f, a, b] - \int_a^b f(x) dx,$$

is bounded by

$$-\frac{(b-a)^3}{24} \overline{D}_2 \leq \text{err}(M_1)[f, a, b] \leq -\frac{(b-a)^3}{24} D_2,$$

where

$$D_k = \min_{x \in (a,b)} \frac{d^k f}{dx^k},$$

$$\overline{D}_k = \max_{x \in (a,b)} \frac{d^k f}{dx^k}.$$

Proof:

$$\begin{aligned} \text{err}(M_1)[f, a, b] &= M_1[f, a, b] - \int_a^b f(x) dx \\ &= (b-a)f(m) - \int_a^b f(m) + (x-m) \frac{df}{dx}(m) + \frac{(x-m)^2}{2} \frac{d^2 f}{dx^2}(\xi) dx \\ &\quad \text{for some } \xi \in (a, b) \\ &= (b-a)f(m) - (b-a)f(m) - [(x-m)]_a^b \frac{df}{dx}(m) - \left[\frac{(x-m)^3}{6} \right]_a^b \frac{d^2 f}{dx^2}(\xi) \\ &= - \left(\frac{(b - \frac{a+b}{2})^3}{6} - \frac{(a - \frac{a+b}{2})^3}{6} \right) \frac{d^2 f}{dx^2}(\xi) \end{aligned}$$

$$= -\frac{(b-a)^3}{24} \frac{d^2 f}{dx^2}(\xi).$$

We can bound $\frac{d^2 f}{dx^2}(\xi)$ by $\underline{D}_2 \leq \frac{d^2 f}{dx^2}(\xi) \leq \overline{D}_2$, so

$$-\frac{(b-a)^3}{24} \overline{D}_2 \leq \text{err}(M_1)[f, a, b] \leq -\frac{(b-a)^3}{24} \underline{D}_2.$$

□

Definition 3.1.3: Instead of Taylor's theorem, we can use **polynomial interpolation** - a polynomial that agrees with f at some points.

Definition 3.1.4: The **Trapezium rule** creates a trapezium with the axis and the endpoints of the strip. It is generally worse than the midpoint rule, so will not be assessed in the course.

$$T_1[f, a, b] = \frac{b-a}{2}(f(a) + f(b)).$$

Theorem 3.1.5: The error bound for the trapezium rule is

$$\frac{1}{12}(b-a)^3 \underline{D}_2 \leq \text{err}(T_1)[f, a, b] \leq \frac{1}{12}(b-a)^3 \overline{D}_2.$$

Remark: The error bound is twice as big as the midpoint rule.

Definition 3.1.6: **Simpson's rule** interpolates a parabola from the endpoints and midpoints. The derivation is quite complicated but the resulting formula is very simple: it is just a weighted average. Although this is over a single strip, Simpson's rule is usually considered to operate over 2 strips, for notational consistency (as we will see later).

$$S_2[f, a, b] = \frac{b-a}{6}(f(a) + 4f(m) + f(b)).$$

Theorem 3.1.7: The error bound for Simpson's rule is

$$\frac{1}{2880}(b-a)^5 \underline{D}_4 \leq \text{err}(S_2)[f, a, b] \leq \frac{1}{2880}(b-a)^5 \overline{D}_4.$$

Remark: This error bound is much better, if a and b are close. But this error bound only applies if the function has at least 4 derivatives; otherwise, the method can still be used, but we can't bound the error in this manner.

Remark: Boole's rule gives a similar formula for interpolating a quartic polynomial, but is a bit more complicated and doesn't offer much improvement over Simpson's rule.

Higher order Taylor approximations might not be a great choice for a better approximation because:

- you need to evaluate high derivatives
- Taylor approximations are local
- functions might not have enough derivatives/be analytic

Higher-order polynomial interpolation also might not be a good idea:

- it is not *necessarily* numerically unstable (as commonly believed), but it can be if you do it badly
- the Runge phenomenon - some functions get worse and worse approximations as the order of the interpolated polynomial increases because polynomial interpolation works worse near the ends of the interval
- but Chebyshev interpolations are much better! These have interpolation points closer together near to the ends of the interval

Definition 3.1.8: The **Runge function** is

$$f(x) = \frac{1}{1 + 25(2x - 1)^2}.$$

Higher-order polynomial interpolation on the Runge function, using evenly spaced points, is *really* bad.

A better method is just to split the interval into smaller strips and use an easier method on each of them. This is guaranteed to converge to the right answer regardless of which method you use, so long as the function is continuous.

Definition 3.1.9: The **composite midpoint rule** is essentially just an average of all the heights of the $n + 1$ endpoints of n strips, multiplied by the width of the region.

$$M_n[f, a, b] = \frac{b - a}{n} \left(f\left(\frac{x_0 + x_1}{2}\right) + \dots + f\left(\frac{x_{n-1} + x_n}{2}\right) \right).$$

Theorem 3.1.10: The error bound for the composite midpoint rule is

$$-\frac{(b-a)^3}{24n^2} \underline{D}_2 \leq \text{err}(M_n)[f, a, b] \leq \frac{(b-a)^3}{24n^2} \underline{D}_2.$$

Remark: The main computation cost is n evaluations of f ; the error is $O(n^{-2})$ so this is pretty good for the amount of work we do.

Definition 3.1.11: The **composite Simpson's rule** is again akin to a weighted average, now with coefficients 1, 4, 2, 4, ..., 4, 1, with the 1s and 4s as before and the 2s coming from where the endpoints are repeated.

$$S_n[f, a, b] = \frac{b-a}{3n} (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \dots + 4f(x_{n-1}) + f(x_n)).$$

Theorem 3.1.12: The error bound for the composite Simpson's rule is

$$\frac{(b-a)^5}{180n^4} \underline{D}_4 \leq \text{err}(S_n)[f, a, b] \leq \frac{(b-a)^5}{180n^4} \underline{D}_4.$$

Remark: The main computational cost is again in evaluations of f , this time $n + 1 = O(n)$. But now the error bound is $O(n^{-4})$ which is even better relative to the amount of work we do.

We can make improvements on these methods:

- Pick more optimal strip endpoints (Gaussian quadrature - **TODO** what is this?)
- Extrapolate a sequence of T_n or M_n (Romberg integration - **TODO** what is this?)
- Recursive subdivision (reuse evaluation, do more [via thinner strips] on difficult bits) - based on 4th derivative being large, or use the a posteriori error estimate

$$|\text{err}(S_{2n})[f, a, b]| \approx \frac{1}{16} |\text{err}(S_n)[f, a, b]| \approx \frac{1}{15} |S_{2n}[f, a, b] - S_n[f, a, b]|$$

TODO proof, how, why??

- Or just do more work where f is large, as the errors have a greater effect there!

Also note that these methods can't help with integrals with $\pm\infty$ in either of the limits.

3.2. Integration in d dimensions

In two dimensions, $\int_R f(x, y) \, d(x, y)$ is the volume under a surface in region $R \subseteq \mathbb{R}^2$.

Theorem 3.2.1 (Fubini's Theorem):

$$\int_R f(x, y) \, d(x, y) = \int_{x_0}^{x_1} \int_{y_0}^{y_1} f(x, y) \, dy \, dx$$

for sufficiently nice f and R (we need to be able to express the bounds nicely). This “split” integral is called an iterated integral.

This extends to higher dimensions as expected.

As long as everything is integrable, the order doesn't matter.

We can use the methods we already know to approximate higher-dimensional integrals, as long as we can split the integral into iterated integrals.

Remark: The midpoint rule can be extended to multiple dimensions by converting into an iterated integral via Fubini's theorem, and then applying the 1-dimensional midpoint rule to successive inner integrals. For a d -dimensional integral I of the function f over the unit hypercube, the midpoint rule with n strips in each dimension gives

$$M_n[f, \mathbf{0}, \mathbf{1}] = \frac{1}{n^d} \sum_{i_1=0}^n \cdots \sum_{i_d=0}^n f\left(\frac{2i_1-1}{2n}, \dots, \frac{2i_d-1}{2n}\right).$$

This is just taking the average of the value of f at the midpoints of each hypercube in a grid over the region.

Remark: Simpson's rule can also be extended to higher dimensions: for 2 dimensions,

$$S_n[f, \mathbf{0}, \mathbf{1}] = \frac{1}{9n^2} \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} w_{ij} f\left(\frac{i}{n+1}, \frac{j}{n+1}\right)$$

where

$$\mathbf{W} = \begin{pmatrix} 1 & 4 & 2 & 4 & \cdots & 4 & 1 \\ 4 & 16 & 8 & 16 & \cdots & 16 & 4 \\ 2 & 8 & 4 & 8 & \cdots & 8 & 2 \\ 4 & 16 & 8 & 16 & \cdots & 16 & 4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 4 & 16 & 8 & 16 & \cdots & 16 & 4 \\ 1 & 4 & 2 & 4 & \cdots & 4 & 1 \end{pmatrix}.$$

Remark: In d dimensions,

$$\text{err}(M_n)[f, R] = O(dn^{-2})$$

$$\text{err}(S_n)[f, R] = O(dn^{-4}).$$

This seems quite good, but we do $O(n^d)$ evaluations of the function; so relative to the work, the error becomes steadily worse.

This is called the “curse of dimensionality”... all of these methods improve extremely slowly in high dimensions.

Definition 3.2.2: **Monte Carlo integration** does the midpoint rule, but using random points according to some distribution.

Consider sampling random points along a scalar function:

$$X \sim \text{Unif}[a, b] \left(\text{i.e. p.d.f } p(x) = \frac{1}{b-a} \text{ for } x \in [a, b], 0 \text{ otherwise} \right).$$

$$\text{Note that } \mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)p(x) dx = \frac{1}{b-a} \int_a^b f(x) dx,$$

$$\text{and also that } \lim_{N \rightarrow \infty} \left[\frac{1}{N} \sum_{i=1}^N f(x_i) \right] = \mathbb{E}[f(X)].$$

Therefore, given X_1, \dots, X_N i.i.d. according to $\text{Unif}[a, b]$, for large enough N ,

$$\int_a^b f(x) dx \approx \frac{b-a}{N} \sum_{i=1}^N f(X_i).$$

We extend this to d dimensions, using the notation

$$MC_N[f, R] := A(R) \frac{1}{N} \sum_{i=1}^N f(\mathbf{X}_i)$$

for large N to approximate $\int_R f(\mathbf{x}) d\mathbf{x}$, with $\mathbf{X}_1, \dots, \mathbf{X}_N$ i.i.d. uniformly distributed across the region R , and $A(R)$ the area of the region R .

Remark: This is called Monte Carlo integration because it is a Monte Carlo algorithm; the answer is random.

Remark: This is in contrast to Las Vegas algorithms, which have random running time but nonrandom answer.

Lemma 3.2.3:

Monte Carlo integration is unbiased (on average it is correct).

Proof: Take $\mathbf{X}_1, \dots, \mathbf{X}_N$ i.i.d. with pdf

$$p(\mathbf{x}) = \begin{cases} \frac{1}{A(R)} & \text{for } \mathbf{x} \in R, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \mathbb{E}[MC_N[f, R]] &= \mathbb{E}\left[A(R) \frac{1}{N} \sum_{i=1}^N f(\mathbf{X}_i)\right] \\ &= \frac{A(R)}{N} \sum_{i=1}^N \mathbb{E}[f(\mathbf{X}_i)] \text{ by linearity of expectation} \\ &= \frac{A(R)}{N} \sum_{i=1}^N \int_R \frac{f(\mathbf{x})}{A(R)} d\mathbf{x} \\ &= \int_R f(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

□

Proposition 3.2.4:

$$\text{Var}(MC_N[f, R]) = \frac{V}{N}$$

for a constant V . Moreover, V can be approximated by \hat{V} that can be expressed in terms of $A(R)$, N , and the $f(\mathbf{X}_i)$ s.

Proof:

$$\begin{aligned} \text{Var}(MC_N[f, R]) &= \text{Var}\left(A(R) \frac{1}{N} \sum_{i=1}^N f(\mathbf{X}_i)\right) \\ &= \frac{A(R)^2}{N^2} \text{Var}\left(\sum_{i=1}^N f(\mathbf{X}_i)\right) \\ &= \frac{A(R)^2}{N^2} \sum \text{Var}(f(\mathbf{X}_i)) \text{ because } X_i\text{s indep because } \mathbf{X}_i\text{s i.i.d.} \\ &= \frac{A(R)^2}{N} \text{Var}(f(\mathbf{X}_i)) \text{ because all variances equal because } \mathbf{X}_i\text{s i.i.d.} \\ &= \frac{V}{N} \text{ with } V := A(R)^2 \text{Var}(f(\mathbf{X}_i)). \end{aligned}$$

$$\begin{aligned}
V &= A(R)^2 \operatorname{Var}(f(\mathbf{x})) \\
&= A(R)^2 \left(\mathbb{E}[f(\mathbf{X}_i)^2] - \mathbb{E}[f(\mathbf{X}_i)]^2 \right) \\
&= A(R)^2 \left(\int_R \frac{f(\mathbf{x})^2}{A(R)} d\mathbf{x} - \left(\int_R \frac{f(\mathbf{x})}{A(R)} d\mathbf{x} \right)^2 \right) \\
&\approx \frac{A(R)^2}{N} \sum_i f^2(\mathbf{X}_i) - \left(\frac{A(R)}{N} \sum_i f(\mathbf{X}_i) \right)^2 \\
&=: \hat{V}.
\end{aligned}$$

□

Definition 3.2.5: The **standard error** is the square root of the estimated variance,

$$\operatorname{SE} = \sqrt{\frac{\hat{V}}{N}} = O(N^{-\frac{1}{2}})$$

Remark: This all looks quite bad, but none of it is dependent on the dimension, so this is actually quite good! The goodness of the algorithm only depends on the area of the region we are integrating over, and the “wiggleness” of the function.

Lemma 3.2.6:

Monte Carlo integration is consistent:

$$\mathbb{P}(|\operatorname{err}(MC_N)[f, R]| \geq \varepsilon) \xrightarrow[N \rightarrow \infty]{} 0 \quad \forall \varepsilon.$$

Proof: Let $Y = MC_N[f, R]$.

Then

$$\mathbb{E}[Y] = \int_R f(\mathbf{x}) d\mathbf{x}, \operatorname{Var}(Y) = \frac{V}{N} \text{ for some } V \in \mathbb{R}.$$

Then by Chebyshev’s inequality,

$$\mathbb{P}(|Y - \mathbb{E}[Y]| \geq c) \leq \frac{V}{Nc^2} \xrightarrow[N \rightarrow \infty]{} 0$$

□

Theorem 3.2.7 (Central Limit Theorem): If Y_1, \dots, Y_N i.i.d., then

$$\mathbb{E}[Y] = \mu \wedge \text{Var}(Y) = \sigma^2 > 0 \implies \frac{1}{N} \sum_{i=1}^N Y_i \rightarrow N\left(\mu, \frac{\sigma^2}{N}\right)$$

(This is not a precise statement.)

That is, a large sample of i.i.d. random variables tends to a normal distribution.

Lemma 3.2.8:

For large N , the error is distributed approximately according to $N\left(0, \frac{\hat{V}}{N}\right)$.

Proof:

$$\begin{aligned} \text{Let } Y_i &= f(\mathbf{X}_i)A(R) - \int_R f(\mathbf{x}) \, d\mathbf{x} \\ &= \text{err}(MC_1)[f, R]. \end{aligned}$$

Then $\mathbb{E}[Y_i] = 0$ and $\text{Var}(Y_i) = \frac{V}{1} \approx \hat{V}$.

Then

$$\begin{aligned} \text{err}(MC_N)[f, R] &= \frac{1}{N} \sum_i Y_i \\ &\rightarrow N\left(0, \frac{V}{N}\right) \text{ by the central limit theorem} \\ &\approx N\left(0, \frac{\hat{V}}{N}\right) \text{ by Lemma 3.2.6.} \end{aligned}$$

□

Corollary 3.2.9: The error of Monte Carlo integration is within 1 standard error about 68% of the time; within 2 standard errors 95% of the time; and within 3 standard error about 99.7% of the time.

Remark:

- There is no guarantee on the error (you could get really unlucky)
- Sampling even from strangely shaped regions is actually easy - just sample from a tight box around it, and reject any samples that are outside the region
- This sampling technique could be quite time consuming in higher dimensions

- Finding the area of the region could be quite difficult - but we can estimate it by considering the size of the sampling box and also how many samples were rejected. (Note that this is also a Monte Carlo integral! of the function $\begin{cases} 1, & x \in R \\ 0, & x \notin R \end{cases}$). We therefore need to adjust the standard error.

Remark: Some improvements we could make:

- Divide R into subregions, perhaps recursively (stratified sampling)
- Sample some parts of R more densely, using the pdf of \mathbf{X}_i as g , then

$$MC_N[f, R] = \frac{1}{N} \sum_i \frac{f(\mathbf{X}_i)}{g(\mathbf{X}_i)}$$

(importance sampling)

- Subtract an easy integral (a control variate) to make the Monte Carlo integral smaller, so that the error is smaller

These don't change the asymptotic error, but can give a better constant term.

4. Accuracy

Definition 4.1: Approximating u by \tilde{u} ,

- $\tilde{u} - u$ is the **error**
- $|\tilde{u} - u|$ is the **absolute error**
- $\frac{|\tilde{u} - u|}{|u|}$ is the **relative error**.

For vector \mathbf{u} , replace $|*|$ with the Euclidean norm $\|*\|$.

Definition 4.2: If $f(x)$ is an ideal function of x , and $\hat{f}(x)$ is an approximate implementation of f , then

- $\hat{f}(x) - f(x)$ is the **forward error**
- $\tilde{x} - x$ is the **backward error**, where $\hat{f}(x) = f(\tilde{x})$.

Definition 4.3: A **floating-point number** is a binary rational approximation to real numbers:

$$\tilde{x} = \pm \left(1 + \frac{m}{2^p}\right) \cdot 2^e$$

where m is the **mantissa**, p is the fixed **precision** (of the mantissa), and e is the **exponent**.

The IEEE 754 standard specifies:

- single precision floats: 23 bit mantissa precision, 8 bit exponent, 1 sign bit
- double precision: 52 bit mantissa precision, 11 bit exponent, 1 sign bit

Definition 4.4: **Machine epsilon** is the ε such that every $x \in \mathbb{R}$ is representable with a relative error of no more than ε .

Note that ε is the relative error when we try to represent

$$x = \overbrace{1.000\dots01}_p \cdot 2^e.$$

Then

$$\varepsilon = \frac{\left| 1 \cdot 2^e - \overbrace{1.000\dots01}_p \cdot 2^e \right|}{|1.000\dots1 \cdot 2^e|} \approx 2^{-(p+1)}.$$

TODO go back over this

Machine epsilon is therefore 2^{-24} for floats, and 2^{-53} for doubles.

Some people define ε to be the smallest ε such that $1 + \varepsilon$ is representable. This is double the definition used here.

Remark: Every $x \in \mathbb{R}$ can be represented as a floating point number with relative error no more than ε .

Remark: If the result of a computation is orders of magnitude smaller than the inputs, we can get “catastrophic cancellation” leading to very large relative errors. E.g. subtraction of close numbers can round really badly for large inputs.

Definition 4.5: **Truncation error** occurs when approximating an infinite process by a finite process.

Roundoff error is due to floating-point storage and computation.

4.1. Iterative methods

Definition 4.1.1: Starting with an initial estimate x_0 approximating unknown x^* , we repeatedly refine it with $x_{n+1} = f(x_n)$, stopping when some criterion is met, e.g. $|x_{n+1} - x_n| < \delta$.

The error at step n is $\varepsilon_n = x_n - x^*$.

If $\varepsilon_n \rightarrow 0$ as $n \rightarrow \infty$:

- x_n has **linear convergence** if for some $0 < a < 1$,

$$\frac{|\varepsilon_{n+1}|}{|\varepsilon_n|} \rightarrow a$$

- x_n has **sublinear convergence** if

$$\left| \frac{\varepsilon_{n+1}}{\varepsilon(n)} \right| \rightarrow 1$$

- \triangleright in particular x_n has **logarithmic convergence** if

$$\frac{|\varepsilon_{n+2} - \varepsilon_{n+1}|}{|\varepsilon_{n+1} - \varepsilon_n|} \rightarrow 1$$

- x_n converges superlinearly if

$$\frac{|\varepsilon_{n+1}|}{|\varepsilon_n|} \rightarrow 0$$

- \triangleright in particular x_n has **order- q convergence** for $q \geq 1$ if

$$\frac{|\varepsilon_{n+1}|}{|n|^q} \rightarrow a$$

for some $a > 0$

Remark: Linear convergence is called that not because the error is $O(n)$, but because the amount of work you need to do to get an extra decimal place is constant, i.e. on a log plot we get a straight line.

Example:

- $\varepsilon_n = 2^{-n}$ converges linearly
- $\varepsilon_n = \frac{1}{n^k}$ converges logarithmically (a straight line on a log-log graph)
- $\varepsilon_n = 2^{-2^n}$ converges superlinearly, and in fact converges quadratically. On a log plot, this is an inverted parabola.

Remark: This can all be extended to vectors easily, by replacing $|\cdot|$ with $\|\cdot\|$.

Remark: Before finding how quickly an iterative method converges, first determine if it converges at all: assume that it does converge, and show it converges to the right thing, then find an expression for ε_{n+1} in terms of ε_n , then in terms of ε_0 with induction, and show that that tends to 0.

Lemma 4.1.2: If x_n converges with order q , then x_{2n} converges with order q^2 .

Proof: **TODO**

□

5. Algorithms for root-finding

5.1. 1-dimensional root-finding

Definition 5.1.1: A **root** of $f : \mathbb{R} \rightarrow \mathbb{R}$ is x^* with $f(x^*) = 0$.

Definition 5.1.2: A **bracket** is an interval (a_0, b_0) with

$$f(a_0)f(b_0) < 0.$$

Theorem 5.1.3: If f is continuous, every bracket contains at least one root.

Remark: This does not hold in higher dimensions.

Remark: We can effectively use binary search to find roots using brackets. However, we might not find an exact zero (because floating point), so we terminate when the interval is small enough, guess x^* is the midpoint of the interval, and can bound x^* by the bracket.

This is called interval bisection.

Iterating, we get a sequence x_0, x_1, \dots with

$$|\varepsilon_n| < \frac{|b_0 - a_0|}{2^{n+1}}$$

so this method converges linearly.

Definition 5.1.4: **Newton's method** in 1 dimension approximates f by its first-order Taylor polynomial.

$$\hat{f}(x) = f(x_0) + (x - x_0) \frac{df}{dx}(x_0).$$

Take a guess x_0 , then find the root of the first-order Taylor polynomial at x_0 ; this is x_1 , i.e.

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

TODO derive this explicitly to make super clear

This converges quadratically *for some* x_0 . We can also get stuck at a fixed point, or hit a turning point and divide by zero, or diverge.

Remark: Some limitations of Newton's method:

- you need to be able to evaluate the derivative
- fails if $\frac{df}{dx}(x_n) = 0$ (throw an error)
- iteration can get stuck in a loop
- iteration can diverge

Theorem 5.1.5: The forward error estimate for Newton's method is

$$|x^* - x_n| \approx |x_{n+1} - x_n|.$$

Proof:

$$\begin{aligned} \hat{f}(x^*) &= f(x_n) + (x^* - x_n) \frac{df}{dx}(x_n) \\ &\approx 0 \text{ for root } x^*. \\ \therefore |x_n - x^*| &\approx \left| \frac{f(x_n)}{\frac{df}{dx}(x_n)} \right| \\ &= |x_{n+1} - x_n| \text{ by the definition of } x_{n+1}. \end{aligned}$$

□

Remark: So we might choose to terminate when successive iterations are sufficiently close, by some parameter *tol*, often 2ϵ or 4ϵ , preferably by some relative error.

We can also look at backward error, $|f(x_n)| < tol$.

There's no one best way of doing a stopping condition, you should think about it.

Remark: You also must stop at a sensible upper bound of steps so you don't get stuck in an infinite loop (ideally with a warning that convergence didn't happen).

You must also stop if an iteration step is poorly defined.

Lemma 5.1.6:

Let $I_c = (x^* - c, x^* + c)$ for some $c \in \mathbb{R}$.

Let

$$A(c) = \frac{\max_{\beta \in I_c} \left| \frac{d^2 f}{dx^2}(\beta) \right|}{\min_{\alpha \in I_c} \left| \frac{df}{dx}(\alpha) \right|}.$$

Then if $x_n \in I_c$, then $|\varepsilon_{n+1}| \leq \frac{A(c)}{2} \varepsilon_n^2$.

Proof:

Suppose $x_n \in I_c$.

$$\begin{aligned} |\varepsilon_{n+1}| &= |x_{n+1} - x^*| \\ &= \left| x_n - \frac{f(x_n)}{f'(x_n)} - x^* \right| \\ &= \left| \frac{(x_n - x^*)f'(x_n) - f(x_n)}{f'(x_n)} \right| \\ &= \frac{|(f(x^*) - (f(x_n) + (x^* - x_n)f'(x_n)))|}{|(f'(x_n))|} \text{ because } f(x^*) = 0 \\ &= \frac{\left| \frac{1}{2}(x_n - x^*)^2 f''(\xi) \right|}{|f'(x_n)|} \text{ for some } \xi \in (x^*, x_n), \text{ by Taylor's theorem} \\ &= \frac{1}{2} \varepsilon_n^2 \frac{|f''(\xi)|}{|f'(x_n)|} \\ &\leq \frac{A(c)}{2} \varepsilon_n^2. \end{aligned}$$

□

Lemma 5.1.7:

For $A(c), I_c$ as before, if $x_0 \in I_c$ and $\frac{cA(c)}{2} < 1$ then x_n converges to x^* at least quadratically.

Proof:

Suppose $\frac{cA(c)}{2} < 1$ and that $x_n \in I_c$ as before. Note that $|\varepsilon_n| < c$ from the definition of I .

$$\begin{aligned}
 |\varepsilon_{n+1}| &\leq \frac{A(c)}{2} \varepsilon_n^2 \text{ from Lemma 5.1.6} \\
 &= |\varepsilon_n| \frac{|\varepsilon_n| A(c)}{2} \\
 &\leq |\varepsilon_n| \text{ because } |\varepsilon_n| \frac{A(c)}{2} < c \frac{A(c)}{2} < 1 \text{ by supposition} \\
 \therefore x_{n+1} &\in I.
 \end{aligned}$$

Then letting $\rho = \frac{|\varepsilon_n| A(c)}{2}$, by induction, $x_n \in I$ for all n , and $|\varepsilon_n| \leq \rho^n |x_0|$. $\rho < 1$ so $|\varepsilon_n| \rightarrow 0$ as $n \rightarrow \infty$.

Therefore $x_n \rightarrow x^*$.

Moreover,

$$\frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^2} \leq \frac{A(c)}{2},$$

so we have *at least* quadratic convergence (it might be better if it actually tends to 0; we don't have equality). □

Remark: The important thing here is that we have proven that the x_n s stay within I_c .

Remark: The condition on c here, $\frac{cA(c)}{2} < 1$, is a sufficient condition for I_c to be a basin of convergence.

Lemma 5.1.8: Let $J_c = (x_0 - c, x_0 + c)$, and let

$$B(c) = \frac{\max_{\beta \in J_c} \left| \frac{d^2 f}{dx^2}(\beta) \right|}{\min_{\alpha \in J_c} \left| \frac{df}{dx}(\alpha) \right|}.$$

Then if $x^* \in J_c$ and $\frac{cB(c)}{2} < 1$, then $x_n \rightarrow x^*$ quadratically.

Proof:

Suppose that $x^* \in J_c$ and $\frac{cB(c)}{2} < 1$.

From Lemma 5.1.6,

$$|\varepsilon_{n+1}| \leq \varepsilon_n^2 \frac{A(c)}{2} < |\varepsilon_n|,$$

because the interval contained within J_c with x^* at its centre, I_d , must have $d \leq c$, so $A(d) \leq A(c)$.

Note that although we cannot guarantee that x_n stays within J_c , we can guarantee that it stays in J_{2c} , if $\frac{cB(c)}{2} < 1$, by the same reasoning as Lemma 5.1.7.

TODO make this a bit nicer from lecture notes □

Remark: We might find such an x_0, c by finding a suitable bracket.

Theorem 5.1.9: There exists an x_0, c such that J_c is a basin of convergence if $\frac{df}{dx}(x^*) \neq 0$.

Proof: Given a bracket containing $(x^* - c, x^* + c)$, we can shrink the interval down such that $A(c) \rightarrow \frac{f''(x^*)}{f'(x^*)}$. If $f'(x^*) \neq 0$, then this is finite and we can find small c such that $\frac{1}{2}cA(c) < 1$; the same is true for $B(c)$. □

Remark: The convergence result doesn't work for double roots (it might still converge, but only linearly).

We can fix it by using the iteration $x_{n+1} = x_n - 2\frac{f(x_n)}{f'(x_n)}$, which will converge quadratically to double roots.

Definition 5.1.10: The **secant method** uses a linear interpolation to approximate f .

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

TODO justification

Remark: This is a “quasi-Newton method” where we use an estimate for the derivative.

Remark: This only needs one evaluation of f at each iteration.

Remark: This can converge superlinearly, in fact order- ϕ in good cases. (proof: exercise)

Remark: In theory this can diverge or get stuck in a loop, but it is more stable than Newton's method, because it relies on the previous 2 iterations.

TODO forward error (same as Newton)

Remark: We should be able to do two iterations of the secant method for no more than the cost of 1 iteration of Newton's method, which would converge with order ≈ 2.6 . This is also nice because we just don't want to evaluate the derivative of f .

Remark: Some other 1-dimensional root finding methods:

- Halley's method - higher order Taylor approximation. But difficult because we might end up with no roots or many roots. But cubic convergence in good cases.
- Muller's method - higher order polynomial interpolation - same problems as Halley's method, but no derivatives needed and order ≈ 1.84 when it works
- Inverse quadratic interpolation, i.e. find $x = ay^2 + by + c$, and the root is at c (always!). Order ≈ 1.84 convergence in good cases.

But the best method is Brent's method:

- maintains a bracket
- uses inverse quadratic interpolation unless iteration undefined or outside the bracket
- uses the secant method as second choice
- falls back to bisection if these do not sufficiently shrink the bracket (only happens at at most half of iterations)
- derivative free
- always converges, at order ≈ 1.84 in good cases.

5.2. d -dimensional root finding

Definition 5.2.1: A **root** of $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ is \mathbf{X}^* with $f(\mathbf{X}^*) = \mathbf{0}$.

If $k < d$, the problem is **underconstrained**, so there are probably infinitely many solutions.

If $k > d$, the problem is **unconstrained**, so there are probably no (exact) solutions.

We will consider $k = d$.

Remark: There isn't really such a thing as a bracket in higher dimensions, because although we might have roots of components, we might not find simultaneous roots of these.

Remark: If instead we take a hyperrectangle with a bracket for components on opposite sides, then we can guarantee a simultaneous root.

TODO diagram

But this is not very useful, because we can't algorithmically prove that a function is positive/negative all the way along a line.

Remark: For higher-dimensional iterative methods, the sensible termination conditions are the same as they were in one dimension, but with norms rather than absolute value, and a tolerance parameter should probably rely on d .

Definition 5.2.2: Given $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \dots \\ f_d(\mathbf{x}) \end{pmatrix}$, we approximate each component by its first-order Taylor polynomial:

$$\hat{f}_i(\mathbf{x}) = f_i(\mathbf{x}_0) + \left(\frac{df_i}{d\mathbf{x}}(\mathbf{x}_0) \right)^\top (\mathbf{x} - \mathbf{x}_0) = 0.$$

TODO juggle equations

We can put all of these together to get

$$\mathbf{J}(\mathbf{f})(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) = -\mathbf{f}(\mathbf{x}_0).$$

So we get an iteration

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}(\mathbf{f})(\mathbf{x}_n)^{-1} \mathbf{f}(\mathbf{x}_n).$$

Each iteration, this requires 1 evaluation of \mathbf{f} , d^2 derivative, and an inversion, which is $O(d^3)$.

Remark: This is not the curse of dimensionality; it only blows up polynomially with d , not exponentially.

Remark: We still have existence of a basin of quadratic convergence, if:

- $\mathbf{f}, \mathbf{J}(\mathbf{f}), \frac{\partial^2 f_i}{\partial x_j \partial x_k}$ are continuous
- **TODO**

Remark: If $\|f(x_{n+1})\| < \|f(x_n)\|$, we should backtrack, **TODO**

Definition 5.2.3: In d dimensions, the equivalent to the secant method is **Broyden's method**:

$$\begin{aligned}\Delta y &= f(x_n) - f(x_{n-1}) \\ \hat{\mathbf{J}}_n &= \hat{\mathbf{J}}_{n-1} + \mathbf{TODO} \\ x_{n+1} &= \mathbf{TODO}\end{aligned}$$

TODO complexity

The convergence is order $q > 1$ in good cases.

Remark: Rather than calculating \mathbf{J}_0 , we can let $\hat{\mathbf{J}}_0 = \mathbf{I}$; this does converge correctly.

Theorem 5.2.4:

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^\top)^{-1} = \mathbf{TODO}$$

Proof: Exercise

□

Remark: This means that we can start from $\hat{\mathbf{J}}_0^{-1}$ and never need to calculate any inverses. This gives $O(d^2)$ for Broyden's method.

6. Algorithms for optimisation

6.1. 1-dimensional minimisation

Remark: If we can show that a function is convex, we can use a root finding algorithm to find the minimum. We can find a bracket for the minimum by finding a bracket for a root of the derivative (but for a minimum it;s got to be negative on the left and positive on the right).

Definition 6.1.1: We can find a bracket for a minimum without evaluating the derivative: if there is $a < z < b$ s.t. $f(z) < f(a), f(z) < f(b)$, then (a, b) is a bracket containing the minimum.

We can make an algorithm for reducing the bracket, by keeping track of the middle value and testing at a new point z' , and discarding one of the endpoints. This is called **golden section search**.

Let's ignore the case of a tie between the middle values.

The placement of z' is quite important: we want ideally the same amount to be chopped off whether $z' > z$ or not.

If we say that we want $c - z' = \phi(c - z)$, then as we're repeating using the same ratios, also $c - z = \phi(c - a)$.

Then

$$\begin{aligned} c - a &= c - z + z - a \\ \implies \frac{1}{\phi}(c - z') &= \phi(c - z') + (c - z'). \end{aligned}$$

Therefore $\frac{1}{\phi} = \phi + 1$

Proposition 6.1.2: This converges linearly.

Proof: **TODO**

□

Remark: Our termination condition should be $|c - a| \leq tol|z|$, where tol should only be about $O(\sqrt{\epsilon})$.

Definition 6.1.3: **Successive parabolic interpolation** carries out golden section search but by choosing z' to be the minimum of the parabola passing through a, z, b .

This is order ≈ 1.32 , but it can go wrong in many ways (exercise).

Remark: Finding a bracket isn't necessarily easy, but if the function is convex, we can use a geometrically expanding bracket to find a bracket. Even better, we can keep the golden ratio, with the geometrically expansion.

Definition 6.1.4: **Brent's method** for minimisation:

- combines successive parabolic interpolation with golden section search, keeping track of 6 points
- is derivative-free and superlinear in good cases.

6.2. d -dimensional minimisation

Given a position \mathbf{x}_n , choose direction \mathbf{d}_n and step length α_n , and set $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{d}_n$.

Define $\mathbf{g}_n = \frac{df}{dx}(\mathbf{x}_n)$, with \mathbf{g} standing for **gradient**.

We would like \mathbf{d}_n to be a descent direction:

$$\begin{aligned} \frac{df(\mathbf{x}_n + \alpha_n \mathbf{d}_n)}{d\alpha} &< 0 \\ \implies \left(\frac{df}{dx} \circ \mathbf{g} \right)^\top \mathbf{J}_\alpha(\mathbf{g}) \\ &= \frac{df}{dx}(\mathbf{x}_n + \alpha \mathbf{d}_n)^\top \mathbf{d}_n \\ &= \mathbf{g}_n \text{ TODO} \end{aligned}$$

We'd also like consecutive directions to be roughly orthogonal, so that they explore the space optimally.

α_n should loosely minimise $f(\mathbf{x}_n + \alpha_n \mathbf{d}_n)$

TODO infinite travel

Definition 6.2.1: **Coordinate gradient descent** chooses alternating basis vectors for the direction. It's not very good.

Definition 6.2.2: **Gradient descent** finds the steepest downhill direction.

$$\begin{aligned} \min \mathbf{d}_n^\top \mathbf{g}_n \text{ s.t. } 1 - \|\mathbf{d}_n\| \geq 0. \Lambda(\mathbf{d}, \mu) &= \mathbf{d}_n^\top \mathbf{g} - \mu(1 - \mathbf{d}_n^\top \mathbf{d}_n) \\ \frac{d\Lambda}{d\mathbf{d}_n} &= \mathbf{g} + 2\mu \mathbf{d} = \mathbf{0} \\ \implies \mathbf{d} &= -\frac{\mathbf{g}}{2\mu} \\ \implies \mathbf{d} &= -\mathbf{g} \end{aligned}$$

So just go in the direction of the negative gradient.

To find a good step length, overshoot and backtrack (typically halving) until the step length is acceptable.

Our acceptable condition is

$$f(\mathbf{x}_n + \alpha_n \mathbf{d}_n) < f(\mathbf{x}_n) + \sigma \alpha_n \mathbf{g}_n^\top \mathbf{d}_n.$$

This is called the **Armijo rule**. σ is typically 0.1 to 0.0001. It says that we want the new value of the function to be below a line that is somewhere in between the 0th order and 1st order approximation to f at \mathbf{x}_n .

It can be proven that this always works if $\sigma < 1$.

We could take an initial guess for α_n to be

$$\alpha_{n-1} \frac{\mathbf{g}_{n-1}^\top \mathbf{d}_{n-1}}{\mathbf{g}_n^\top \mathbf{d}_n}.$$

TODO justification

We still need a large initial step length. We could even forward-track until we get something that doesn't meet the Armijo rule.

TODO overall iteration

TODO termination condition.

TODO complexity

TODO convergence

Definition 6.2.3: **Conjugate gradient descent** uses **conjugate directions** that satisfy $\mathbf{d}_n^\top \mathbf{H} \mathbf{d}_n = 0$.

$$\mathbf{d}_n = -\mathbf{g}_n + \mathbf{d}_{n-1} \frac{(\text{TODO})}{\text{TODO}}$$

TODO justification, what does this mean?

Remark: Conjugate gradient descent isn't actually on the syllabus but it's useful and quite simple (only one line of code changed compared to gradient descent).

Remark: We can also use Newton's method:

Approximate f by its second-order Taylor polynomial

$$\hat{f}(\mathbf{x}) = f(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^\top \mathbf{g}_0 + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H}(f)(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0).$$

Differentiating this gives

$$\frac{d\hat{f}}{d\mathbf{x}} = \mathbf{0} + \mathbf{g}_n - \mathbf{0} + \mathbf{H}_0(\mathbf{x} - \mathbf{x}_0).$$

where $\mathbf{H}_n = \mathbf{H}(f)(\mathbf{x}_n)$.

Therefore $\mathbf{x}_1 - \mathbf{x}_0 = -\mathbf{H}_0^{-1} \mathbf{g}_0$. But remember that computing matrix inverses is slow, so better to say solve $\mathbf{H}_0 \mathbf{d}_n = -\mathbf{g}_n$ for \mathbf{d}_n .

TODO expression for \mathbf{d}_n

This has problem-independent step length (**TODO** why?).

TODO (dis)advantages.

Remark: Newton's method doesn't always give a descent direction.

$$\begin{aligned} \mathbf{d}_n &= -H_n^{-1}g_n \\ g_n^\top \mathbf{d}_n &= -g_n^\top H_n^{-1}g_n \\ < 0 \text{ if } H_n^{-1} \text{ pos def} &\iff H_n \text{ pos def.} \end{aligned}$$

Therefore Newton's method can converge to maxima or saddle points equally as much as minima.

TODO compare the two Newton's methods (they're the same)

Remark: We could use a quasi-Newton method whereby we use an approximation of the Hessian, e.g. by using Broyden's method on the derivative. However, this is bad because it doesn't necessarily give a symmetric approximation for the Hessian.

We want $\hat{\mathbf{H}}_n$:

- to satisfy secant equation **TODO**
- to be symmetric
- to be close to $\hat{\mathbf{H}}_{n-1}$
- to be positive definite.

Positive definiteness requires the curvature condition $\Delta \mathbf{x}^\top \Delta \mathbf{g} > 0$, which we can't always guarantee.

Definition 6.2.4: **BFGS update** is **TODO** (not in spec).

If the curvature condition fails, reset $\hat{\mathbf{H}}_n$.

Best to start with $\hat{\mathbf{H}}_0 = \mathbf{I}$, and reset to the identity if the curvature condition fails. When the approximate Hessian is the identity, it is just gradient descent; so it does gradient descent until it gets to a convex section of the function, at which point it starts approximating the Hessian and goes a lot quicker.

TODO advantages

TODO disadvantages

Remark: There is a way to only update the inverse Hessians, like we did for Broyden's method. It's a bit more complicated because this uses a rank-2 update rather than a rank-1 update.

Remark: We can make a rank- m approximation of the Hessians and get $O(md)$ memory. We call this L-BFGS.

Index

Absolute error	27	Positive definite	10
Armijo rule	39	Positive semidefinite	10
BFGS update	41	Precision	27
Backward	27	Relative error	27
Bracket	30	Root	30, 35
Brent's method	38	Roundoff error	28
Broyden's method	37	Runge function	20
Composite Simpson's rule	21	Saddle point	12
Composite midpoint rule	20	Scalar-vector product	6
Concave	13	Secant	2
Conjugate directions	40	Secant method	34
Conjugate gradient descent	40	Simpson's rule	19
Constrained	9	Slack	16
Continuous	2	Standard error	25
Convex	12, 13	Strictly concave	13
Coordinate gradient descent	39	Strictly convex	13
Derivative	2	Sublinear convergence	29
Differentiable	2	Successive parabolic interpolation	38
Dot-product rule	6	Tangent	2
Error	27	Tight	16
Exponent	27	Trapezium rule	19
Feasible region	9	Truncation error	28
Floating-point number	27	Unconstrained	9, 35
Forward error	27	Underconstrained	35
Golden section search	38	Vector derivative	4
Gradient descent	39	Vector field	5
Hessian	5	Vector-valued function	5
Inconsistent	9		
Indefinite	10		
Jacobian	5		
Lagrangian	16		
Linear convergence	28		
Logarithmic convergence	29		
Machine epsilon	28		
Mantissa	27		
Midpoint rule	18		
Monte Carlo integration	23		
Negative definite	10		
Negative semidefinite	10		
Newton's method	30		
Order-q convergence	29		
Partial chain rule	6		
Partial derivative	4		
Polynomial interpolation	19		