Notes Review

Course Name Here

Topic Name Here

Your Name

July 29, 2025

Abstract

This document contains a comprehensive review of notes for [Course/Topic]. It includes key concepts, formulas, examples, and practice problems.

Contents

1 Overview

Topic Summary

Provide a brief overview of the main topic and its importance.

- Main concept 1
- Main concept 2
- Main concept 3

2 TODO

Topic Summary

TODO exam problems

- SS 2023 2c. sparse grids
- SS 2023 3c. space filling curves

3 Key Definitions

Definition 3.1 (Important Term). *Define the important term here. For example:*

*A **function** $f : A \to B$ is a relation that assigns to each element $a \in A$ exactly one element $b \in B$.*

Definition 3.2 (Understand how to convert between trigonometric and exponential using Euler's formula). *Define the important term here. For example:*

*A **function** $f : A \to B$ is a relation that assigns to each element $a \in A$ exactly one element $b \in B$.*

Key Point

Remember that definitions are the foundation of understanding. Make sure you can state them precisely!

4 Main Theorems and Results

Theorem 4.1 (Fundamental Theorem). *State the theorem here. For example:*

Let $f : [a, b] \to \mathbb{R}$ be continuous. Then f attains its maximum and minimum values on $[a, b]$.

Proof. Sketch of proof or key ideas... □

Important Formula

Key formula to remember:

$$\int_a^b f(x)\,dx = F(b) - F(a) \tag{1}$$

where $F'(x) = f(x)$.

5 Examples and Applications

Example 5.1 (Worked Example)**.** *Consider the function $f(x) = x^2$. We want to find...*
Solution:

$$f'(x) = 2x \tag{2}$$
$$f''(x) = 2 \tag{3}$$

Therefore, the function has a minimum at $x = 0$.

Warning

Common mistake: Don't forget to check the domain when finding extrema!

6 Endterm Summer 2025

Example 6.1 (Problem 1a: Quarter-Wave Fourier coefficients are purely imaginary)**.** *Given a real-valued input dataset $f_0, \ldots, f_{2N-1}$ with the symmetry condition $f_{2N-n-1} = -f_n$, show that the Quarter-Wave Fourier coefficients*

$$F_k = \frac{1}{2N} \sum_{n=0}^{2N-1} f_n \omega_{2N}^{-k(n+\frac{1}{2})}$$

have only imaginary values and can be written as

$$F_k = -\frac{i}{N} \sum_{n=0}^{N-1} f_n \sin\left(\frac{\pi k}{N}\left(n + \frac{1}{2}\right)\right)$$

Solution:

1. ***Split the sum:*** *Separate the sum into two parts*

$$F_k = \frac{1}{2N}\left(\sum_{n=0}^{N-1} f_n \omega_{2N}^{-k(n+\frac{1}{2})} + \sum_{n=N}^{2N-1} f_n \omega_{2N}^{-k(n+\frac{1}{2})}\right)$$

2. ***Apply symmetry condition:*** *Use $f_n = -f_{2N-n-1}$ in the second sum and change variables*

3. ***Combine terms:*** *After simplification, obtain*

$$F_k = \frac{1}{2N} \sum_{n=0}^{N-1} f_n \left(\omega_{2N}^{-k(n+\frac{1}{2})} - \omega_{2N}^{k(n+\frac{1}{2})}\right)$$

4. ***Use Euler's formula:*** *Since $e^{-ix} - e^{ix} = -2i\sin(x)$, we get*

$$F_k = -\frac{i}{N} \sum_{n=0}^{N-1} f_n \sin\left(\frac{\pi k}{N}\left(n + \frac{1}{2}\right)\right)$$

Key insight: *The anti-symmetry condition $f_{2N-n-1} = -f_n$ causes the real parts to cancel, leaving only imaginary sine components.*

Example 6.2 (Problem 1b: Symmetry condition for QW-DST coefficients). *Show that the coefficients F_k of the QW-DST satisfy the symmetry condition $F_k = F_{2N-k}$.*

Solution:

Starting formula: *From part (a), we have*

$$F_k = -\frac{i}{N}\sum_{n=0}^{N-1} f_n \sin\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right)$$

Compute F_{2N-k}:

$$F_{2N-k} = -\frac{i}{N}\sum_{n=0}^{N-1} f_n \sin\left(\frac{\pi(2N-k)}{N}\left(n+\frac{1}{2}\right)\right)$$

Simplify the sine argument:

$$\frac{\pi(2N-k)}{N}\left(n+\frac{1}{2}\right) = 2\pi n + \pi - \frac{\pi k}{N}\left(n+\frac{1}{2}\right)$$

Apply sine properties:

- $\sin(2\pi n + x) = \sin(x)$ *(periodicity)*
- $\sin(\pi - x) = \sin(x)$ *(symmetry)*

Therefore: $F_{2N-k} = F_k$

Key insight: *Due to this symmetry, we only need to compute F_k for $k = 0, \ldots, N$ rather than all $2N$ coefficients, reducing computational requirements by half.*

Example 6.3 (Problem 1c: Computing coefficients for QW-DST using real FFT). *We assume a procedure `real-FFT(g,N)` that computes Fourier coefficients G_k efficiently on real dataset g that consists of $2N$ values g_n. Use this procedure to compute coefficients F_k for $k = 0, \ldots, N-1$ from equation (2) for (non-symmetrical) real data $f_0, \ldots, f_{N-1}$ stored in parameter field g.*

Solution: *Pre-process the data using anti-symmetric extension, perform real-FFT, then post-process coefficients using the relationship between DFT and QW-DST coefficients.*

1. ***Create anti-symmetric data sequence of length $2N$:***

$$g_n = f_n, \quad \text{for } n = 0, 1, \ldots, N-1 \tag{4}$$
$$g_{2N-n-1} = -f_n, \quad \text{for } n = 0, 1, \ldots, N-1 \tag{5}$$

2. ***Run real-FFT to compute G_k from g_n:***

$$G_k = \texttt{real-FFT}(g, N), \quad \text{for } k = 0, \ldots, N$$

3. ***Extract QW-DST coefficients from FFT coefficients:***

$$F_k = G_k \omega_{2N}^{-k/2} = G_k e^{-i\pi k/(2N)}, \quad \text{for } k = 0, \ldots, N-1$$

Key insight: *The QW-DST can be computed from a standard FFT by exploiting the anti-symmetric extension and the phase shift relationship $F_k = G_k \omega_{2N}^{-k/2}$ between the transforms.*

Example 6.4 (Problem 2b: Admissibility condition for wavelets)**.** *Given the drawing from part (a), what conclusions can you make about the integral of the obtained wavelet? What does this imply for wavelets in general?*

Solution:

Observation from the graph: *The area above the x-axis equals the area below the x-axis. Therefore:*

$$\int_{-\infty}^{+\infty} \psi_1(t)\, dt = 0$$

Implications for wavelets in general:

1. ***Admissibility condition:*** *All wavelets must satisfy* $\int_{-\infty}^{+\infty} \psi(t)\, dt = 0$
2. ***Zero-mean property:*** *Wavelets oscillate around zero and represent fluctuations relative to the average*
3. ***Complementary roles:***
    - *Scaling functions* $\phi(t)$ *capture average (DC) information*
    - *Wavelets* $\psi(t)$ *capture details and variations*
4. ***Signal processing interpretation:*** *Wavelets act as band-pass filters that:*
    - *Cannot detect constant signals*
    - *Capture oscillations and local changes*
    - *Are ideal for multi-scale analysis while preserving edges and transitions*

Key insight: *The zero-integral property ensures wavelets are orthogonal to constant functions, making them perfect for decomposing signals into different frequency bands and scales.*

Example 6.5 (Problem 2b: Admissibility condition for wavelets)**.** *Given the drawing from part (a), what conclusions can you make about the integral of the obtained wavelet. What does this imply for wavelts in general?*

Solution:

1. Area above x-axis equals area below x-axis $Thismeanstheintegralofthewaveltfun$ (6)

2. This implies the mean is zero.$Thewaveletbasisfunctionswillrepresentfluctuationstotheaverageofthecoar$ (7)

3. The admissibility condition is met $Theintegralofthewaveletoverallvalu$ (8)

(9)

Main idea: Scaling function captures avergae (DC) information, while wavelts capture details. Wavelets act as band-pass filters and cannot detect constant siganls. They capture oscillations and local changes on signals. Intuition: Zero-mean prop makes wavelets ideal for analyzing signals at different scales while preserving important features like edges and transitions.

Example 6.6 (Problem 3b: Hierarchical surpluses for smooth functions)**.** *Consider hierarchical interpolant on* $(N+1)\times(N+1)$ *grids for increasing number of grid points* $N = 2^L$*. Characterize how the size of surpluses decreases for sufficiently smooth functions* f*.*

Solution:

1. Along horizontal/vertical grid lines: *Hierarchical surpluses decrease by* $\mathbf{1/4}$ *from level to level.*

- *These represent 1D refinement where grid spacing halves:* $h \to h/2$
- *Surpluses are proportional to* $h^2 \cdot |f''|$
- *Since* $(h/2)^2 = h^2/4$*, surpluses decrease by factor of* $1/4$

2. Along diagonal grid lines: *Hierarchical surpluses decrease by* $\mathbf{1/16}$.

- *Points added at diagonal positions constitute 2D refinement*
- *Both* x *and* y *directions refined simultaneously*
- *Surplus proportional to* $h_x^2 \cdot h_y^2$
- *When* $h_x \to h_x/2$ *and* $h_y \to h_y/2$*:* $(h_x/2)^2 \cdot (h_y/2)^2 = h_x^2 h_y^2/16$

3. Absolute size depends on derivatives:

- *Horizontal/vertical points: proportional to pure second derivatives* $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}$
- *Diagonal points: proportional to mixed second derivative* $\frac{\partial^2 f}{\partial x \partial y}$

Key insight: *The rapid decrease of surpluses (especially* $1/16$ *for diagonal points) explains why sparse grids can efficiently approximate smooth functions in high dimensions by omitting many grid points while maintaining accuracy.*

Example 6.7 (Problem 4a: Sparse grids data structures)**.** *Name the different data structures used for sparse grids and discuss the following points: hierarchization/dehierarchization (consider data access and traversal complexity), spatial adaptivity, and memory consumption.*

Solution:

1. Arrays *Grid points are stored in a contiguous array. A mapping is needed from hierarchical index (l,i) and flat index j.*

- *Hierarchization/dehierarchization: access by index O(1), but need mapping between hierarchical index (l,i) and flat 1D index. Hierarchical neighbors can be deduced directly from current node's hierarchical index.*
- *Spatial adaptivity: cannot add or delete elements; does not support spatial adaptivity. However, dimensional adaptivity can be accomodated.*

- *Memory consumption: only data is tored i.e.,* $v_j ateachgridpoint.Noneedtostore(l,i)ifcorrectmappingisprovide$

2. Along diagonal grid lines: *Hierarchical surpluses decrease by* $\mathbf{1/16}$.

- *Points added at diagonal positions constitute 2D refinement*
- *Both* x *and* y *directions refined simultaneously*
- *Surplus proportional to* $h_x^2 \cdot h_y^2$
- *When* $h_x \to h_x/2$ *and* $h_y \to h_y/2$*:* $(h_x/2)^2 \cdot (h_y/2)^2 = h_x^2 h_y^2/16$

3. Absolute size depends on derivatives:

- *Horizontal/vertical points: proportional to pure second derivatives* $\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}$
- *Diagonal points: proportional to mixed second derivative* $\frac{\partial^2 f}{\partial x \partial y}$

Key insight: *The rapid decrease of surpluses (especially* $1/16$ *for diagonal points) explains why sparse grids can efficiently approximate smooth functions in high dimensions by omitting many grid points while maintaining accuracy.*

Example 6.8 (Problem 5b: Arithmetic representation of the Peano-Meander Curve)**.** *You are given the parametrization q(t) of the Peano-Meander curve, where t is the representation t in base nine system. Determine the operators Q1,Q4,Q6.*

Solution:

1. Layout the curve *Divide the curve into blocks Q0-Q8.*

2. Along diagonal grid lines: *Identify different patterns and whether reflection/coordinate swap is needed.*

3. Absolute size depends on derivatives:

Key insight: *The rapid decrease of surpluses (especially* $1/16$ *for diagonal points) explains why sparse grids can efficiently approximate smooth functions in high dimensions by omitting many grid points while maintaining accuracy.*

Example 6.9 (Problem 5b 2: Computing coordinates on Peano-Meander curve)**.** *You are given the parametrization q(t) of the Peano-Meander curve. You are given coordinates of q(2/3) and q(1/2).*

Solution:

1. Determine nonary representation *For q(2/3), 2/3 =* $0_9.6.q(2/3) = Q6(00) = (2/32/3).For q(1/2), 1/2 = 0_9.44444...$

2. Write the representation in operator form *q(1/2) = Q4 * Q4 * ... (0 0) = lim n-¿inf* $Q_4^n(00)$

3. Absolute size depends on derivatives:

Key insight: *The rapid decrease of surpluses (especially* $1/16$ *for diagonal points) explains why sparse grids can efficiently approximate smooth functions in high dimensions by omitting many grid points while maintaining accuracy.*

7 Endterm Summer 2023

Example 7.1 (Problem 1: Discrete Fourier Periodicity)**.** *Derive the relationship between* F_k *and* G_{2k} *for an N-periodic dataset f.*

Solution:

$$\text{1. Write the definition of } G_{2k}: \quad G_{2k} = \frac{1}{2N}\sum_{n=0}^{2N-1} f_n e^{-i2\pi n(2k)/(2N)} \tag{10}$$

$$\text{2. Simplify the exponential: } e^{-i2\pi n(2k)/(2N)} = e^{-i2\pi nk/N} \tag{11}$$

$$\text{3. Split the sum into 2 parts: } \sum_{n=0}^{2N-1} = \sum_{n=0}^{N-1} + \sum_{n=N}^{2N-1} \tag{12}$$

$$\text{4. Use periodicity: substitute } n = N + j \text{ where } f_{N+j} = f_j \tag{13}$$

$$\text{5. Combine the sums(both terms become identical)} \tag{14}$$

$$\text{6. Result: } G_{2k} = \frac{1}{2N} \cdot 2\sum_{n=0}^{N-1} f_n e^{-i2\pi nk/N} = F_k \tag{15}$$

Main idea: When applying a 2N-point DFT to N-periodic data, the even-indexed coefficients (G_0, G_2, G_4, , G_{2N-2}) *are identical to the coefficients of an N-point DFT* (F_0, F_1, F_2, F_{N-1})*. Intuition: This happens because the N-periodic data repeats itself in the second half of the 2N samples. Each data value appears exactly twice in the 2N-point transform, effectively doubling its contribution, which when normalized by the 1/(2N) factor, gives the same result as the N-point DFT.*

Example 7.2 (Problem 1b: Discrete Sine Transform)**.** *Given a* $2N$ *data set* $f_{-N+1}, \ldots, f_0, f_1, \ldots, f_N$ *with antisymmetry condition* $f_{-n} = -f_n$ *and all* $f_n \in \mathbb{R}$*, prove that:*

$$F_k = \frac{1}{2N} \sum_{n=-N+1}^{N} f_n \omega_{2N}^{nk} = \frac{-i}{N} \sum_{n=1}^{N-1} f_n \sin\left(\frac{\pi nk}{N}\right)$$

Solution:
Step 1: *Apply antisymmetry to find zero values*

- *At* $n = 0$*:* $f_0 = -f_{-0} = -f_0 \Rightarrow f_0 = 0$
- *At* $n = N$*: By consistency with antisymmetry,* $f_N = 0$

Step 2: *Expand the DFT sum and eliminate zero terms*

$$F_k = \frac{1}{2N} \left[\sum_{n=-N+1}^{-1} f_n \omega_{2N}^{nk} + \sum_{n=1}^{N-1} f_n \omega_{2N}^{nk} \right]$$

Step 3: *Apply antisymmetry to the negative index sum*

- *Substitute* $m = -n$ *in the first sum*
- *Use* $f_{-m} = -f_m$ *to get:*

$$\sum_{n=-N+1}^{-1} f_n \omega_{2N}^{nk} = -\sum_{m=1}^{N-1} f_m \omega_{2N}^{-mk}$$

Step 4: *Combine the sums*

$$F_k = \frac{1}{2N} \sum_{n=1}^{N-1} f_n \left[\omega_{2N}^{nk} - \omega_{2N}^{-nk} \right]$$

Step 5: *Express* ω_{2N} *and apply Euler's formula*

- $\omega_{2N} = e^{2\pi i/2N} = e^{\pi i/N}$
- $\omega_{2N}^{nk} - \omega_{2N}^{-nk} = e^{\pi ink/N} - e^{-\pi ink/N} = 2i \sin\left(\frac{\pi nk}{N}\right)$

Step 6: *Substitute and simplify*

$$F_k = \frac{1}{2N} \sum_{n=1}^{N-1} f_n \cdot 2i \sin\left(\frac{\pi nk}{N}\right) = \frac{i}{N} \sum_{n=1}^{N-1} f_n \sin\left(\frac{\pi nk}{N}\right)$$

Note: *The negative sign in the final result depends on the DFT sign convention used.*
Main idea: *For antisymmetric real data, the DFT reduces to a purely imaginary Discrete Sine Transform, containing only sine terms due to the odd symmetry of the input.*

Example 7.3 (Problem 1c: Computing DST coefficients using FFT)**.** *Given a dataset* f_n *for* $n = 1, \ldots, N-1$*, use the result from part (b) to efficiently compute the Discrete Sine Transform (DST) coefficients using FFT.*
Solution:
Step 1: Preprocessing - Create antisymmetric extension

- *Given: size-N vector with values* $f_1, f_2, \ldots, f_{N-1}$

- *Create size-$2N$ antisymmetric vector:*

$$f_0 = 0 \tag{16}$$
$$f_n = f_n \text{ for } n = 1, \ldots, N-1 \tag{17}$$
$$f_N = 0 \tag{18}$$
$$f_{-n} = -f_n \text{ for } n = 1, \ldots, N-1 \tag{19}$$

Step 2: Apply FFT *Compute F_k using standard FFT on the $2N$-point antisymmetric sequence.*

Step 3: Postprocessing - Extract real DST coefficients *From part (b), we know that for antisymmetric data:*

$$F_k = \frac{-i}{N} \sum_{n=1}^{N-1} f_n \sin\left(\frac{\pi n k}{N}\right)$$

To obtain the real-valued DST coefficients:

$$\hat{F}_k = -Im\{F_k\} = \frac{1}{N} \sum_{n=1}^{N-1} f_n \sin\left(\frac{\pi n k}{N}\right)$$

for $k = 1, \ldots, N-1$.

Key insight: *The antisymmetric extension causes the DFT to produce purely imaginary coefficients proportional to the sine series. By extracting the imaginary part and flipping the sign, we obtain the real-valued DST coefficients efficiently using FFT.*

Example 7.4 (Problem 2: Sparse Grids)**.** *The goal is to represent a 2D function using a sparse grid of level 3. First, specify which basis functions are used by giving exact ranges for level indices $\vec{l}$ and $\vec{i}$. For each basis function used on the sparse grid, mark the corresponding point in the tableau of subspaces shown in Figure 2.1.*

Solution:

1. ***Identify subspaces to include:*** *For sparse grid S_3, use subspaces where $|\vec{l}|_1 = l_1 + l_2 \leq 3 + 1 = 4$.*
   This gives the combinations: $(1,1)$, $(1,2)$, $(1,3)$, $(2,1)$, $(2,2)$, $(3,1)$.

2. ***Determine basis functions for each subspace:***
   Index set: $I_{\vec{l}} = \{\vec{i} : 1 \leq i_d < 2^{l_d},\ \text{all } i_d \text{ odd}\}$
   We only use odd indices to avoid redundancy with coarser levels.

3. ***Mark grid points:*** *Each basis function $\phi_{\vec{l},\vec{i}}$ corresponds to a grid point $\vec{x}_{\vec{l},\vec{i}} = (i_1 \cdot 2^{-l_1}, i_2 \cdot 2^{-l_2})$ in the tableau.*

Main idea: *Sparse grids efficiently approximate multi-dimensional functions without succumbing to the curse of dimensionality. They achieve this by:*

- *Using a hierarchical structure that builds on coarser grids*
- *Selecting only odd-indexed basis functions to avoid redundancy*
- *Balancing computational efficiency with approximation accuracy*

Each grid point corresponds to one basis function in the sparse grid representation.

Example 7.5 (Problem 2b: Coefficients for sparse grid approximation)**.** *Identify which coefficients must be computed to evaluate a 2D function at the point $(1/3, 2/3)$ using sparse grid approximation.*

Solution:

1. ***Identify relevant subspaces:*** *For sparse grid level $n = 3$ in dimension $d = 2$, the subspaces satisfying $|l|_1 \leq n + d - 1 = 4$ are:*
   - $(l_1, l_2) \in \{(1,1), (1,2), (1,3), (2,1), (2,2), (3,1)\}$
2. ***Define index sets I_l:*** *For each level l, the index set contains odd integers:*
   - $I_1 = \{1\}$
   - $I_2 = \{1, 3\}$
   - $I_3 = \{1, 3, 5, 7\}$
3. ***Identify grid points:*** *For each subspace (l_1, l_2) and indices $(i_1, i_2) \in I_{l_1} \times I_{l_2}$, compute:*
   $$x_{l,i} = \left(\frac{i_1}{2^{l_1}}, \frac{i_2}{2^{l_2}}\right)$$
4. ***Select relevant basis functions:*** *The hierarchical basis function $\phi_{l,i}(x)$ centered at $x_{l,i}$ has support on:*
   $$\left[\frac{i_1 - 1}{2^{l_1}}, \frac{i_1 + 1}{2^{l_1}}\right] \times \left[\frac{i_2 - 1}{2^{l_2}}, \frac{i_2 + 1}{2^{l_2}}\right]$$
   Include coefficient $a_{l,i}$ if and only if $(1/3, 2/3)$ lies within this support.

Main idea: *Sparse grids efficiently approximate multi-dimensional functions by:*

- *Using hierarchical basis functions with local support*
- *Including only basis functions whose support contains the evaluation point*
- *Exploiting the tensor product structure while avoiding the curse of dimensionality*

The coefficients to evaluate correspond to basis functions that are non-zero at $(1/3, 2/3)$.

Local support *refers to the region where a basis function is non-zero. Outside this region, the function equals zero.*

Hierarchical basis functions *in sparse grids are hat functions (piecewise linear) that resemble triangular tents. To evaluate f at a given point, we only need coefficients for basis functions whose support contains that point.*

Example 7.6 (Problem 3: Space-filling curves)**.** *The goal is to derive the operators M_0 through M_8 that transform the first iteration of the Meurthe (Peano) curve into each sub-square of the second iteration.*

Solution Approach:

For each operator M_i, determine the transformation needed by analyzing:

1. ***Scaling:*** *Since we divide into a 3×3 grid, scale by $\frac{1}{3}$ in both directions.*
2. ***Rotation/Reflection:*** *Determine what rotation or reflection is needed so the curve segment connects properly with adjacent sub-squares.*
3. ***Translation:*** *Determine the offset to position the transformed curve in the correct sub-square.*

Method: *For each sub-square i:*

- *Identify how the curve should enter and exit the sub-square*
- *Determine what transformation of the original square pattern achieves this*

- *Express as* $M_i(x, y) = (ax + by + c, dx + ey + f)$

Example for M_0***:***

- *Sub-square 0 is at position* $(0, 0)$ *to* $(\frac{1}{3}, \frac{1}{3})$
- *The transformation* $M_0(x, y) = (\frac{y}{3}, \frac{x}{3})$ *swaps* x *and* y *(90° rotation) and scales by* $\frac{1}{3}$
- *This creates the correct curve segment that starts at* $(0, 0)$ *and connects to sub-square 1*

Example 7.7 (Problem 3b: Space-filling curves)**.** *Find the parameter* t *such that* $m(t) = (1/2, 1/2)$.

Solution:

1. ***Locate*** $(1/2, 1/2)$ ***in the grid:*** *In the second iteration's* 3×3 *grid, the point* $(1/2, 1/2)$ *lies in the center sub-square (sub-square 4). Therefore, the first digit is* $n_1 = 4$*, so* t *starts with* $0_9.4\ldots$
2. ***Check if*** $(1/2, 1/2)$ ***is a fixed point:*** *Given* $M_4(x, y) = (-y/3 + 2/3, -x/3 + 2/3)$*, we solve:*
$$1/2 = -y/3 + 2/3 \Rightarrow y = 1/2 \tag{20}$$
$$1/2 = -x/3 + 2/3 \Rightarrow x = 1/2 \tag{21}$$
Therefore $M_4(1/2, 1/2) = (1/2, 1/2)$*, confirming* $(1/2, 1/2)$ *is a fixed point of* M_4.
3. ***Determine subsequent digits:*** *Since* $(1/2, 1/2)$ *is a fixed point of* M_4*, it remains in sub-square 4 at every level. Therefore, all digits are 4:* $t = 0_9.\overline{4}$
4. ***Convert to decimal:***
$$0_9.\overline{4} = \frac{4}{9} + \frac{4}{81} + \frac{4}{729} + \ldots \tag{22}$$
$$= \frac{4}{9} \cdot \frac{1}{1 - \frac{1}{9}} = \frac{4}{9} \cdot \frac{9}{8} = \frac{1}{2} \tag{23}$$
5. ***Verification:*** *Since* $t = 1/2 = 0_9.\overline{4}$*:*
$$m(1/2) = M_4 \circ M_4 \circ M_4 \circ \cdots (0, 0) = \lim_{n \to \infty} M_4^n(0, 0) = (1/2, 1/2)$$
The limit converges to the fixed point of M_4.

Key insights:

- *The center of parameter space (*$t = 1/2$*) maps to the center of the unit square (*$(1/2, 1/2)$*).*
- *Fixed point analysis simplifies the recursive calculation.*
- *Function composition (*$M_4 \circ M_4 \circ \cdots$*) determines the mapping, while addition converts the base-9 representation to decimal.*
- *The expression* $M_4 \circ M_4 \circ M_4 \circ \cdots (0, 0)$ *is the expansion of the Meurthe function* m *applied to* $t = 0_9.\overline{4}$ *(which equals* $1/2$*). For infinite compositions to converge, the limit point must be a fixed point of the operator—a fundamental property of space-filling curves.*

Answer: $t = 1/2$

Example 7.8 (Problem 4: Haar Wavelet Transform in 1D)**.** *The goal is to implement a Python function that performs the Haar wavelet transform on a 1D data sequence. The Haar wavelet is used in multi-resolution data analysis, where each element in the transformed array provides information about the data at different scales.* ***Solution Approach:*** *The Haar wavelet transform recursively decomposes a signal into averages (low-frequency components) and differences (high-frequency details). Given a filter function that processes pairs of elements, we apply it recursively to achieve a complete multi-resolution decomposition.* ***Implementation:***

```python
def dwt_1d(c, jmax):
"""
Performs 1D Discrete Wavelet Transform using Haar wavelets.
Parameters:
- c: Input array (modified in-place)
- jmax: Current size of the array to process

After transformation:
- c[0]: Overall average of the entire signal
- c[1]: Low-frequency detail (difference between halves)
- c[2:]: Higher-frequency details at successive scales
"""
# Base case: nothing to do for single element
if jmax == 1:
    return

# Apply filter to current level
# This computes averages in first half, differences in second half
filter_1d(c, jmax)

# Recursively process the averages (first half)
# This creates a multi-resolution representation
dwt_1d(c, jmax // 2)
```

How it works:

1. ***Level 1:*** *Process all elements, storing averages in the first half and differences in the second half*
2. ***Level 2:*** *Process only the averages from Level 1, further decomposing them*
3. ***Continue recursively*** *until only one average remains*

Example: *For array [2, 4, 6, 8]:*

- *Initial: [2, 4, 6, 8]*
- *After Level 1: [3, 7, -1, -1] (averages: 3, 7; differences: -1, -1)*
- *After Level 2: [5, -2, -1, -1] (overall average: 5; scale differences: -2, -1, -1)*

The final array contains the signal decomposed into different frequency components, useful for compression, denoising, and analysis.

Example 7.9 (Problem 4b: Haar Wavelet Transform in 2D)**.** *Implement a Python function that performs the 2D Haar wavelet transform using the provided 1D implementation.*

Solution Approach: *The 2D Haar wavelet transform applies the 1D transform along rows and columns, creating a multi-resolution decomposition with four quadrants at each level.*

Implementation:

```python
def dwt_2d(image, jmax):
    """
    Performs 2D Discrete Wavelet Transform using Haar wavelets.

    Parameters:
    - image: 2D array of size 2^p x 2^p (modified in-place)
    - jmax: Current size of the subimage to process

    After transformation, the image contains:
    - Top-left (LL): Low frequencies in both directions
    - Top-right (LH): Vertical edges (horizontal high-freq)
    - Bottom-left (HL): Horizontal edges (vertical high-freq)
    - Bottom-right (HH): Diagonal edges and noise
    """
    # Base case
    if jmax <= 1:
        return

    # Apply 1D DWT along all rows
    for row in range(jmax):
        row_data = image_row(image, row, jmax)
        dwt_1d(row_data, jmax)
        for col in range(jmax):
            image[row][col] = row_data[col]

    # Apply 1D DWT along all columns
    for col in range(jmax):
        column_data = image_col(image, col, jmax)
        dwt_1d(column_data, jmax)
        for row in range(jmax):
            image[row][col] = column_data[row]

    # Recursive call on top-left quadrant (LL)
    dwt_2d(image, jmax // 2)
```

How it works:

1. ***Row transformation:*** *Apply 1D DWT to each row, splitting the image into left (low-frequency) and right (high-frequency) halves.*
2. ***Column transformation:*** *Apply 1D DWT to each column, creating four quadrants:*
   - ***LL (top-left):*** *Averages of averages - smoothed version*
   - ***LH (top-right):*** *Row averages, column differences - vertical edges*
   - ***HL (bottom-left):*** *Row differences, column averages - horizontal edges*
   - ***HH (bottom-right):*** *Differences of differences - diagonal edges and noise*
3. ***Recursive decomposition:*** *Process the LL quadrant recursively to create a multi-resolution representation.*

Key insight: *Averages act as low-pass filters (preserving smooth variations), while differences act as high-pass filters (capturing rapid changes and edges).*

Further questions: *Does it matter if you process rows first or columns first?*

Example 7.10 (Problem 4c: Deriving formula for inverse Haar wavelet coefficients)**.** *We are given the formulas which describe the averaging and differencing performed by the forward Haar* *wavelet transform:*

$$c_j^{(l)} = \frac{1}{2}c_{2j}^{(l+1)} + \frac{1}{2}c_{2j+1}^{(l+1)} \quad \textit{(averaging)} \tag{24}$$

$$d_j^{(l)} = \frac{1}{2}c_{2j}^{(l+1)} - \frac{1}{2}c_{2j+1}^{(l+1)} \quad \textit{(differencing)} \tag{25}$$

Using these formulas, we derive formulas for the coefficients of the inverse Haar wavelet transform.

Solution: *We transform the given forward transform formulas into a system of equations and solve for the finer level coefficients* $c_{2j}^{(l+1)}$ *and* $c_{2j+1}^{(l+1)}$.

1. ***Add equations (1) and (2):***

$$c_j^{(l)} + d_j^{(l)} = \frac{1}{2}c_{2j}^{(l+1)} + \frac{1}{2}c_{2j+1}^{(l+1)} + \frac{1}{2}c_{2j}^{(l+1)} - \frac{1}{2}c_{2j+1}^{(l+1)} \tag{26}$$

$$= c_{2j}^{(l+1)} \tag{27}$$

2. ***Subtract equation (2) from equation (1):***

$$c_j^{(l)} - d_j^{(l)} = \frac{1}{2}c_{2j}^{(l+1)} + \frac{1}{2}c_{2j+1}^{(l+1)} - \frac{1}{2}c_{2j}^{(l+1)} + \frac{1}{2}c_{2j+1}^{(l+1)} \tag{28}$$

$$= c_{2j+1}^{(l+1)} \tag{29}$$

Result: *The inverse Haar wavelet transform formulas are:*

$$\boxed{c_{2j}^{(l+1)} = c_j^{(l)} + d_j^{(l)}} \quad \textit{(even indices)} \tag{30}$$

$$\boxed{c_{2j+1}^{(l+1)} = c_j^{(l)} - d_j^{(l)}} \quad \textit{(odd indices)} \tag{31}$$

Key insight: *The inverse transform reconstructs finer-level coefficients by combining the coarse-level averages with the detail coefficients. Adding recovers even-indexed coefficients, while subtracting recovers odd-indexed coefficients.*

Warning

Common mistake: Don't forget to check the domain when finding extrema!

8 Problem Solving Strategies

8.1 General Approach

1. **Understand the problem:** Read carefully and identify what's given and what's asked
2. **Plan:** Choose appropriate methods/theorems
3. **Execute:** Carry out the calculations
4. **Check:** Verify your answer makes sense

8.2 Specific Techniques

Technique 1: Integration by Parts

When to use: Products of functions where one becomes simpler when differentiated
Formula: $\int u\,dv = uv - \int v\,du$
Example types:

- $\int xe^x\,dx$
- $\int x\sin x\,dx$
- $\int \ln x\,dx$

9 Quick Reference Sheet

9.1 Formulas at a Glance

Concept	Formula
Derivative of x^n	nx^{n-1}
Chain Rule	$(f\circ g)'(x) = f'(g(x))\cdot g'(x)$
Product Rule	$(fg)' = f'g + fg'$
Quotient Rule	$\left(\frac{f}{g}\right)' = \frac{f'g-fg'}{g^2}$

9.2 Common Pitfalls

Forgetting the chain rule

Sign errors in integration by parts

Not checking endpoints for absolute extrema

10 Practice Problems

10.1 Basic Problems

1. Find the derivative of $f(x) = x^3\sin(2x)$
2. Evaluate $\int_0^\pi x\cos x\,dx$
3. Find all critical points of $g(x) = x^3 - 3x^2 + 2$

10.2 Advanced Problems

1. Prove that if f is differentiable and $f'(x) > 0$ for all $x \in (a, b)$, then f is strictly increasing on (a, b).
2. Find the volume of the solid of revolution...

11 Summary and Key Takeaways

Key Point

Basis Functions in Sparse Grids

1. **Start with 1D hat functions:** The piecewise linear hat function at level l and position i is:
$$\phi_{l,i}(x) = \max(0, 1 - |2^l \cdot x - i|)$$
This creates a tent shape with peak value 1 at $x = i \cdot 2^{-l}$, linearly decreasing to 0 at neighboring grid points.

2. **2D basis functions via tensor product:**
$$\phi_{\vec{l},\vec{i}}(\vec{x}) = \phi_{l_1,i_1}(x_1) \cdot \phi_{l_2,i_2}(x_2)$$

3. **Hierarchical construction:** Each level adds new information (hierarchical surplus) not captured by coarser levels:
$$W_l = V_l - V_{l-1}$$
where V_l is the space of piecewise linear functions on grid level l.

4. **Basis functions represent differences:** Each basis function captures what's new at its level, not redundant information from coarser grids.

5. **Why odd indices only:** Even indices coincide with points from coarser levels. For example, at level 2, index $i = 2$ gives point $2 \cdot 2^{-2} = 0.5$, which already exists at level 1 with $i = 1$.

11.1 Connections to Other Topics

- This material is fundamental for [next topic]
- It builds upon [previous topic]
- Applications include [real-world example]

A Additional Resources

- Textbook: Chapter X, Sections Y-Z
- Online resource: `https://example.com`
- Practice problems: Problem set

B Detailed Proofs