Linear Algebra Lecture Notes

University Course

Contents

1 Matrix Derivative Rules

Function	**Gradient w.r.t. x**
$\mathbf{b}^T\mathbf{x}$ or $\mathbf{x}^T\mathbf{b}$	$\mathbf{b}$
$\mathbf{x}^T A\mathbf{x}$	$(A + A^T)\mathbf{x}$; if $A = A^T$: $2A\mathbf{x}$
$\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\|\mathbf{x}\|$	$\frac{\mathbf{x}}{\|\mathbf{x}\|}$, if $\mathbf{x} \neq 0$
$\mathbf{b}^T A\mathbf{x}$	$A^T\mathbf{b}$

2 Matrix Rank

Definition 2.1 *(Full Rank Matrix) A matrix $A \in \mathbb{R}^{m\times n}$ is said to be of full rank if:*

- *For $m \leq n$: rank$(A) = m$ (full row rank)*
- *For $m \geq n$: rank$(A) = n$ (full column rank)*
- *For $m = n$: rank$(A) = n = m$ (full rank square matrix)*

A square matrix is full rank if and if and only if it is invertible

3 Jordan Canonical Form

For any $n \times n$ matrix A, there exists an invertible matrix P such that:

$$J = P^{-1}AP \tag{1}$$

where J is the Jordan canonical form of A.

- **Block Diagonal:** J is a block diagonal matrix composed of Jordan blocks.
- **Jordan Blocks:** Each block J_i has the form:

$$J_i = \begin{pmatrix} \lambda_i & 1 & 0 & \cdots & 0 \\ 0 & \lambda_i & 1 & \cdots & 0 \\ 0 & 0 & \lambda_i & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda_i \end{pmatrix} \tag{2}$$

  where λ_i is an eigenvalue of A.

Key Properties:

- **Existence:** Every square matrix has a Jordan canonical form.
- **Uniqueness:** The JCF is unique up to the ordering of Jordan blocks.
- **Eigenvalues:** The diagonal entries of J are the eigenvalues of A.
- **Algebraic Multiplicity:** Size of the corresponding Jordan block.
- **Geometric Multiplicity:** Number of Jordan blocks for each distinct eigenvalue.

- **Diagonalizability:** A is diagonalizable if and only if J is diagonal (all Jordan blocks are 1×1).
- **Minimal Polynomial:** Degree of the largest Jordan block for each distinct eigenvalue.
- **Nilpotent Part:** $(J - \lambda I)$ is nilpotent for each Jordan block.

4 Positive Definite Matrices

A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called **positive definite** if for all nonzero vectors $\vec{x} \in \mathbb{R}^n$, the quadratic form $\vec{x}^T A \vec{x} > 0$. If the inequality is non-strict, i.e., $\vec{x}^T A \vec{x} \geq 0$, then A is called **positive semidefinite** (PSD).

For a real symmetric matrix A, the following are equivalent and characterize positive definiteness:

- All eigenvalues of A are strictly positive.
- All leading principal minors of A are strictly positive (Sylvester's criterion).
- A admits a unique Cholesky decomposition $A = LL^T$, where L is a lower triangular matrix with positive diagonal entries.
- $\vec{x}^T A \vec{x} > 0$ for all $\vec{x} \neq 0$.

Geometrically, a positive definite matrix defines a strictly convex quadratic form, meaning the surface $f(\vec{x}) = \vec{x}^T A \vec{x}$ curves upward in every direction. This property is crucial in optimization, where such forms guarantee unique global minima.

Covariance matrices are classic examples of symmetric positive semidefinite matrices. Given a random vector $\vec{X} \in \mathbb{R}^n$ with finite second moments, the covariance matrix is defined as:

$$\Sigma = \mathrm{Cov}(\vec{X}) = \mathbb{E}\left[(\vec{X} - \mathbb{E}[\vec{X}])(\vec{X} - \mathbb{E}[\vec{X}])^T\right]$$

Proof that covariance matrices are positive semidefinite: Let $\vec{x} \in \mathbb{R}^n$. Consider the quadratic form:

$$\vec{x}^T \Sigma \vec{x} = \vec{x}^T \, \mathbb{E}\left[(\vec{X} - \mathbb{E}[\vec{X}])(\vec{X} - \mathbb{E}[\vec{X}])^T\right] \vec{x}$$

By linearity of expectation:

$$= \mathbb{E}\left[\vec{x}^T (\vec{X} - \mathbb{E}[\vec{X}])(\vec{X} - \mathbb{E}[\vec{X}])^T \vec{x}\right]$$

Recognizing the scalar product inside:

$$= \mathbb{E}\left[\left(\vec{x}^T (\vec{X} - \mathbb{E}[\vec{X}])\right)^2\right] \geq 0$$

Since this is the expectation of a square, it is always non-negative. Therefore, $\vec{x}^T \Sigma \vec{x} \geq 0$ for all $\vec{x}$, and Σ is positive semidefinite.

5 Matrix Factorizations

5.1 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental tool in linear algebra that generalizes the eigendecomposition of square matrices to any $m \times n$ matrix. It is widely used in numerical analysis, data compression, and machine learning.

Definition

Let $A \in \mathbb{R}^{m \times n}$. Then the SVD of A is:

$$A = U \Sigma V^T$$

where:

- $U \in \mathbb{R}^{m \times m}$ is an orthogonal matrix (columns are called **left singular vectors**),
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (columns are **right singular vectors**),
- $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix with non-negative real numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ on the diagonal, called **singular values**.

Interpretation

The matrix A can be expressed as a sum of rank-one matrices:

$$A = \sum_{i=1}^{r} \sigma_i \, \mathbf{u}_i \mathbf{v}_i^T$$

where $r = \text{rank}(A)$. Each term represents a scaled projection in the direction of $\mathbf{u}_i$ and $\mathbf{v}_i$.

Geometric Meaning

The decomposition describes the action of A as:

1. a rotation or reflection by V^T,
2. followed by scaling by Σ,
3. followed by a rotation or reflection by U.

Key Properties

- The number of nonzero singular values equals the rank of A.
- $\sigma_i = \sqrt{\lambda_i}$ where λ_i are eigenvalues of $A^T A$.
- $\|A\|_2 = \sigma_1$ (spectral norm).
- $\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$ (Frobenius norm).

Low-Rank Approximation

Eckart–Young Theorem: The best rank-k approximation to A in the Frobenius norm is given by truncating the SVD:

$$A_k = \sum_{i=1}^{k} \sigma_i \, \mathbf{u}_i \mathbf{v}_i^T$$

This A_k minimizes $\|A - B\|_F$ over all matrices B of rank at most k.

Applications

- **Principal Component Analysis (PCA)**: SVD is used to find principal directions.
- **Data compression**: keep only the top k singular values.
- **Noise reduction**: discard small singular values.
- **Text analysis**: Latent Semantic Analysis in NLP.
- **Solving ill-posed problems**: truncated SVD for numerical stability.

5.2 LDL$^{\mathbf{T}}$ Decomposition

Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric matrix. Then, under mild conditions, A admits an **LDL$^{\mathbf{T}}$ decomposition**:

$$A = LDL^T$$

where:

- L is a unit lower triangular matrix (i.e., lower triangular with 1s on the diagonal),
- D is a diagonal matrix,
- L^T is the transpose of L.

This decomposition always exists when A is symmetric and nonsingular. It is especially useful in numerical algorithms because it avoids computing square roots, unlike the Cholesky decomposition.

5.3 Cholesky Decomposition

Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric positive definite matrix. Then A admits a unique **Cholesky decomposition**:

$$A = LL^T$$

where L is a lower triangular matrix with strictly positive diagonal entries.

The Cholesky decomposition is efficient for solving linear systems $A\vec{x} = \vec{b}$, particularly in large-scale problems. It is also widely used in numerical optimization and probabilistic modeling, such as in sampling from multivariate normal distributions.

If A is symmetric but not positive definite, the Cholesky decomposition does not exist in the real domain.

There is a close relationship between the Cholesky and LDL$^{\mathrm{T}}$ decompositions. If $A = LDL^T$ is the LDL$^{\mathrm{T}}$ decomposition of a positive definite matrix, then one can construct the Cholesky factor as:

$$A = (L\sqrt{D})(L\sqrt{D})^T$$

where $\sqrt{D}$ is the diagonal matrix formed by taking square roots of the entries of D.

6 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that seeks to find a new orthogonal basis, called principal components, that maximizes the variance of the projected data. Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ with n observations and p features, PCA aims to find a linear transformation $\mathbf{W} \in \mathbb{R}^{p \times k}$ that projects $\mathbf{X}$ onto a lower k-dimensional subspace while maximizing the variance of the projected data.

Step 1: Data Centering Center the data by subtracting the mean of each feature:

$$\bar{\mathbf{X}} = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T \tag{3}$$

where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ is the mean vector and $\mathbf{1}$ is a vector of ones.

Step 2: Covariance Matrix Computation Calculate the sample covariance matrix:

$$\mathbf{C} = \frac{1}{n-1}\bar{\mathbf{X}}^T\bar{\mathbf{X}} \tag{4}$$

Step 3: Eigendecomposition Since the covariance matrix C is symmetric and positive semidefinite, we can perform eigenvalue decomposition.

$$\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T \tag{5}$$

where:

- $\mathbf{V} \in \mathbb{R}^{p \times p}$ is an orthogonal matrix whose columns are the eigenvectors of $\mathbf{C}$
- $\boldsymbol{\Lambda} \in \mathbb{R}^{p \times p}$ is a diagonal matrix containing the eigenvalues of $\mathbf{C}$

Explicitly, these matrices have the following form:

$$\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_p] \tag{6}$$

where $\mathbf{v}_i$ are the eigenvectors of $\mathbf{C}$, and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{bmatrix} \tag{7}$$

where λ_i are the eigenvalues of $\mathbf{C}$, typically arranged in descending order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. We can use Cholesky Decomposition on Λ since it's positive semidefinite.

$$\mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T = \mathbf{V}\delta^T\delta\mathbf{V}^T = (\delta\mathbf{V}^T)^T(\delta\mathbf{V}^T) \tag{8}$$

Hence, we can write

$$(\delta\mathbf{V}^T)^T(\delta\mathbf{V}^T) = \bar{\mathbf{X}}^T\bar{\mathbf{X}}, \tag{9}$$

and hence

$$\bar{\mathbf{X}} = \delta\mathbf{V}^T \tag{10}$$

or

$$\delta = \bar{\mathbf{X}}\mathbf{V}. \tag{11}$$

The new projected data δ is called the Principal Components (PCs) or Scores while the eigenvectors matrix $\mathbf{V}$ is called the Loadings. Note that, in the `scikit-learn` package, the notations are `delta = pca.fit_transform(X)` and `V_T = pca.components_`.

Step 4: Sorting Eigenvectors Sort the eigenvectors in descending order of their corresponding eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \tag{12}$$

Step 5: Dimensionality Reduction Choose the first k eigenvectors to form the projection matrix:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k], \mathbf{W} \in \mathbb{R}^{p \times k} \tag{13}$$

$$\hat{\delta} = [\delta_1, \delta_2, \ldots, \delta_k], \delta \in \mathbb{R}^{n \times k} \tag{14}$$

Step 6: Data Projection Reconstruct the centered data using the new k-dimensional subspace:

$$\mathbf{Y} = \hat{\delta}\mathbf{W}^T, \mathbf{Y} \in \mathbb{R}^{n \times p} \tag{15}$$

6.1 Data Centering in PCA

Consider a dataset $\mathbf{X} \in \mathbb{R}^{n \times p}$ with n observations and p features. PCA on non-centering data referes to eigendecomposition on the $\frac{1}{n-1}\mathbf{X}^T\mathbf{X}$ matrix instead of the covariance.

1. The PCA is performed on the matrix:

$$\hat{\mathbf{S}} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} \tag{16}$$

2. We can decompose $\mathbf{X}$ into its mean part and the centered part:

$$\mathbf{X} = \mathbf{1}\boldsymbol{\mu}^T + \mathbf{X}_c \tag{17}$$

   where $\mathbf{1}$ is a column vector of ones, $\boldsymbol{\mu}$ is the mean vector, and $\mathbf{X}_c$ is the centered data.

3. Substituting this into the covariance matrix:

$$\hat{\mathbf{S}} = \frac{1}{n-1}(\mathbf{1}\boldsymbol{\mu}^T + \mathbf{X}_c)^T(\mathbf{1}\boldsymbol{\mu}^T + \mathbf{X}_c) \tag{18}$$

$$= \frac{1}{n-1}(n\boldsymbol{\mu}\boldsymbol{\mu}^T + \boldsymbol{\mu}\mathbf{1}^T\mathbf{X}_c + \mathbf{X}_c^T\mathbf{1}\boldsymbol{\mu}^T + \mathbf{X}_c^T\mathbf{X}_c) \tag{19}$$

4. As n approaches infinity, the dominant term becomes $\boldsymbol{\mu}\boldsymbol{\mu}^T$, assuming the mean is non-zero.

5. The eigenvector corresponding to the largest eigenvalue of $\boldsymbol{\mu}\boldsymbol{\mu}^T$ is proportional to $\boldsymbol{\mu}$ itself.

6. Therefore, as n gets large, the first principal component of the uncentered data will align more closely with the mean vector $\boldsymbol{\mu}$, rather than the direction of maximum variance in the centered data.

One may think why we need to think about the case of performing PCA on this weird matrix $\hat{S}$. This is due to we can quickly perform PCA and SVD decompoistion at the same time if the data is centered. We can compute the SVD of X:

$$X = U\,S\,V^T, \tag{20}$$

where

- $U \in \mathbb{R}^{n \times p}$ has orthonormal columns (left singular vectors),
- $S = \mathrm{diag}(s_1, s_2, \ldots, s_{\min(n,p)})$ contains the singular values $s_i \geq 0$,
- $V \in \mathbb{R}^{p \times p}$ has orthonormal columns (right singular vectors).

From this decomposition,

$$C = \frac{1}{n-1} X^T X = \frac{1}{n-1} V S U^T U S V^T = V \frac{S^2}{n-1} V^T,$$

showing that

- the right singular vectors V are the principal directions (eigenvectors) of C,
- the eigenvalues satisfy $\lambda_i = \dfrac{s_i^2}{n-1}$.

Moreover, the PC scores can be written

$$X V = U S V^T V = U S,$$

so that the columns of US are the principal component vectors (scores).

Summary

- If $X = U S V^T$, then the columns of V are the principal directions (eigenvectors) of the covariance matrix.
- The columns of $U S$ are the principal components (scores).
- The singular values s_i relate to the covariance eigenvalues by

  $$\lambda_i = \frac{s_i^2}{n-1},$$

  and λ_i measures the variance explained by the ith PC.
- *This entire framework is valid only if X is centered*, i.e. its column means have been subtracted so that $X^T X/(n-1)$ is the true sample covariance.

6.2 Weak Stationarity and the Covariance Matrix

When applying PCA to time series data, it is essential that each series be *weakly stationary*. A univariate process $\{x_t\}$ is weakly stationary if and only if:

1. **Constant mean:**

   $$\mathbb{E}[x_t] = \mu \quad \text{for all } t.$$

2. **Finite, time–invariant variance:**

   $$\mathrm{Var}(x_t) = \mathbb{E}[(x_t - \mu)^2] = \sigma^2 < \infty \quad (\text{same for all } t).$$

3. **Autocovariance depends only on lag:**

   $$\mathrm{Cov}(x_t,\, x_{t+h}) = \mathbb{E}\big[(x_t - \mu)(x_{t+h} - \mu)\big] = \gamma(h),$$

   i.e. a function of the lag h alone, *not* of the time index t.

Implications for covariance estimation

- If the series are *not* weakly stationary (e.g. raw prices with trends or time-varying volatility), then the sample covariance

$$\widehat{\mathrm{Cov}}(x^i, x^j) \;=\; \frac{1}{n-1}\sum_{t=1}^{n}\big(x_t^i - \bar{x}^i\big)\big(x_t^j - \bar{x}^j\big)$$

  will depend heavily on the chosen time window and may not converge to a stable population value.

- Covariance-based techniques (PCA, Mahalanobis distance, factor analysis) applied to non-stationary data often pick up *spurious* structure (trends, regime shifts) instead of genuine co-movement.

- Therefore, before constructing or interpreting a covariance matrix for time series, one should first transform the data (e.g. taking log-returns, demeaning, or variance-stabilizing) so that the resulting series are approximately weakly stationary.

7 PCA in Financial Markets: Factor Interpretation, Trading, and Pitfalls

This section connects the mathematical PCA machinery to its use in financial markets. Fixed–income markets, in particular, are well described by a small set of *uncorrelated linear factors* that capture the dominant co-movements of the yield curve. PCA provides a statistical method to estimate these factors, their loadings, and the hedging ratios needed to neutralise them.

7.1 Factor Representation and PCA Interpretation

A generic K-factor model for n assets can be written as:

$$\begin{pmatrix} y_1^t \\ \vdots \\ y_n^t \end{pmatrix} = \sum_{i=1}^{K} \alpha_i^t \begin{pmatrix} f_{i1} \\ \vdots \\ f_{in} \end{pmatrix} + \begin{pmatrix} \epsilon_1^t \\ \vdots \\ \epsilon_n^t \end{pmatrix},$$

where α_i^t are time-varying factor values, f_{ij} are loadings, and ϵ_i^t are idiosyncratic residuals.

PCA provides a direct statistical analogue:

- **Factors** = principal components (scores)
- **Loadings** = eigenvectors (columns of $\mathbf{V}$)
- **Residuals** = deviations from PCA reconstruction

In practice, one must remember:

- eigenvalues may be numerically unstable,
- eigenvectors must be orthogonal,
- eigenvectors are defined up to a sign (x and $-x$ are equivalent).

7.2 Applications to Yield Curves: Interpretation and Hedging

PCA is especially effective in fixed–income markets. Using weekly Bund yields (2y, 5y, 7y, 10y) as an example, the first three principal components typically correspond to:

- **PC1 — Level**: all loadings have the same sign.
- **PC2 — Slope**: short and long maturities load with opposite signs.
- **PC3 — Curvature**: mid–maturities load opposite to the wings.

The largest loading on the first eigenvector identifies the pivot point of level shifts. PCs may correlate with macroeconomic drivers (other bond markets, FX, equity indices), although PC3 is often uncorrelated.

Hedging Using PCA Loadings Let there be m traded maturities with price sensitivities (BPVs)

$$BPV_j = \frac{\partial P_j}{\partial y_j}, \qquad j = 1, \ldots, m,$$

and PCA eigenvectors

$$\mathbf{v}_i = (e_{i,1}, e_{i,2}, \ldots, e_{i,m})^\top.$$

A hedge uses notional vector

$$\mathbf{n} = (n_1, \ldots, n_m)^\top.$$

Neutralising PC exposure requires setting the portfolio's sensitivity to the chosen PCs to zero. For example, to hedge a *5y–10y steepening* (mostly a slope factor):

$$n_5 BPV_5 e_{1,5} + n_{10} BPV_{10} e_{1,10} = 0.$$

Solving for the hedge ratio:

$$\frac{n_5}{n_{10}} = \frac{BPV_{10}}{BPV_5} \left(\frac{e_{1,5}}{e_{1,10}} \right).$$

This ensures that the two instruments exactly offset their PC1 exposures.

More generally, hedging against PC1 and PC2 simultaneously requires solving:

$$\mathbf{Hn} = 0, \qquad \mathbf{H} = \begin{pmatrix} BPV_1 e_{1,1} & \cdots & BPV_m e_{1,m} \\ BPV_1 e_{2,1} & \cdots & BPV_m e_{2,m} \end{pmatrix}.$$

In some applications, this is written compactly (for selected instruments) as:

$$\begin{pmatrix} n_2 \\ n_{10} \end{pmatrix} = \begin{pmatrix} BPV_2 l_{12} \\ BPV_{10} l_{22} \end{pmatrix},$$

where l_{ij} denotes the required PCA loading ratios.

PCA-Based Trade Construction A typical PCA-based trading workflow includes:

- choosing relevant maturities and data horizon (e.g., 1 year),
- running PCA on the full tenor set, identifying meaningful maturities,
- re-running PCA on a reduced universe for robustness,
- checking numerical stability,
- inspecting eigenvalues, eigenvectors, and PC time series,
- testing correlations between PCs and macro drivers,
- computing hedge ratios and expected carry for trade construction.

7.3 Pitfalls: Correlation Breakdown and Instability

Despite their mathematical appeal, PCA factors are not guaranteed to be stable through time.

Correlation Breakdown The PCs are uncorrelated *over the full sample*, but may become correlated during subperiods. Such breakdowns can cause hedges to fail. Traders therefore check PC correlations specifically around the trade entry window.

Eigenvector Instability Eigenvectors may rotate over time, leading to:

- misaligned hedges,
- incorrect economic interpretation of factors.

This instability is especially relevant for stressed market periods or when the yield–curve regime shifts.

8 Kalman Filter

The Kalman Filter is a powerful algorithm used for state estimation in dynamic systems. It operates in two main phases: Predict and Update.

8.1 Model Components

- **State-Transition Model:** Describes how the state evolves over time.
- **Observation Model:** Relates the true state to the observed measurements.
- **Control-Input Model (B_k):** Represents the effect of control inputs on the state.
- **Control Vector (u_k):** The vector of control inputs.
- **Process Noise Covariance (Q_k):** Covariance of the process noise (w_k).
- **Observation Noise Covariance (R_k):** Covariance of the observation noise.

8.2 Predict Step

This step projects the state and covariance estimates from the previous time step to the current time step.

- **Predicted State Estimate:**

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$$

(Note: The document had $\frac{\beta_h \mu}{h}$ which is likely a typo and should represent $B_k u_k$ in standard Kalman filter notation.)

- **Predicted Covariance Estimate:**

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

(Note: The document used F_k^+ which is interpreted as F_k^T for the transpose of the state-transition matrix.)

8.3 Update Step

This step corrects the predicted estimates using the current observation.

- **Measurement Residual (Innovation):**

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1}$$

This represents the difference between the actual observation (z_k) and the predicted observation ($H_k \hat{x}_{k|k-1}$).

- **Residual Covariance:**

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

This is the covariance of the measurement residual. (Note: The document used $H^\dagger$ which is interpreted as H_k^T for the transpose of the observation model matrix.)

- **Kalman Gain:**

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

The Kalman gain determines how much the predictions are corrected based on the new measurement. The document implies K_k is the Kalman gain. It "depends on which is more noisy", meaning it balances the uncertainty in the prediction and the measurement.

- **Updated State Estimate:**

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

This updates the state estimate based on the residual and Kalman gain. The document mentions "update based on".

- **Updated Covariance Estimate:**

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

This updates the covariance estimate.

8.4 Main Assumptions

- **Linearity and Time-Invariance:** The system is assumed to be linear and time-invariant in its state-space form.
- **Noise Properties:** The state noise and measurement noise are assumed to be zero-mean and independent of each other.