Radial Basis Function Networks (RBFNs) are widely used for **function interpolation** and **approximation** due to their ability to model complex, high-dimensional functions.

- **Interpolation:** RBFNs can be trained to **pass exactly through given data points**.
- **Approximation:** RBFNs can be used to **approximate** a function when there is noise or when exact fitting is unnecessary.

Applications:

✓ Engineering (surface fitting, robotics)

✓ Finance (trend prediction)

✓ Medical imaging (image reconstruction)

✓ Machine learning (data classification, regression)

2. Understanding Function Interpolation with RBFNs

Definition of Interpolation

Given a dataset of **N points** (x_i, y_i), interpolation constructs a function $f(x)$ such that:

$$f(x_i) = y_i \quad \forall i \in \{1, 2, ..., N\}$$

This means the function **exactly** passes through the given data points.

Using RBFNs for Interpolation

An RBFN models the function as:

$$f(x) = \sum_{i=1}^{N} w_i \phi(||x - c_i||)$$

where:

- c_i are **centers** (typically chosen as data points).
- w_i are **weights** to be determined.
- $\phi(r)$ is a **radial basis function** (commonly Gaussian):

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Solving for Weights (w_i)

Since interpolation requires **exact fitting**, we set up the system:

$$\Phi W = Y$$

where:

- Φ is the **activation matrix** with entries $\Phi_{ij} = \phi(||x_i - c_j||)$.
- W is the vector of unknown weights.
- Y is the vector of target outputs.

3. Understanding Function Approximation with RBFNs

Definition of Approximation

In **approximation**, we do not require an exact fit but instead minimize the **error** between predicted values and true values:

$$E = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

This is useful when:

- Data contains **noise**.
- An exact fit leads to **overfitting**.
- The number of centers c_i is **smaller than the number of data points**.

Using RBFNs for Approximation

$$f(x) = \sum_{i=1}^{M} w_i \phi(||x - c_i||)$$

where $M < N$ (fewer centers than data points).

To find the weights, we use **Least Squares Approximation** instead of exact matrix inversion:

$$W = (\Phi^T \Phi)^{-1} \Phi^T Y$$

This finds the **best approximation** in a **least squares sense**.

Choosing Centers for Approximation

- **Clustering methods (K-Means, Random Selection)** to select a subset of data points as centers.
- **Regularization (Ridge Regression, L2 norm)** to avoid overfitting.
- **Cross-validation** to tune spread parameter σ.

Key Properties of RBFN for Approximation

- Handles noisy data better.
- Less prone to overfitting.
- Faster computation than exact interpolation.
- Generalizes well to unseen data.