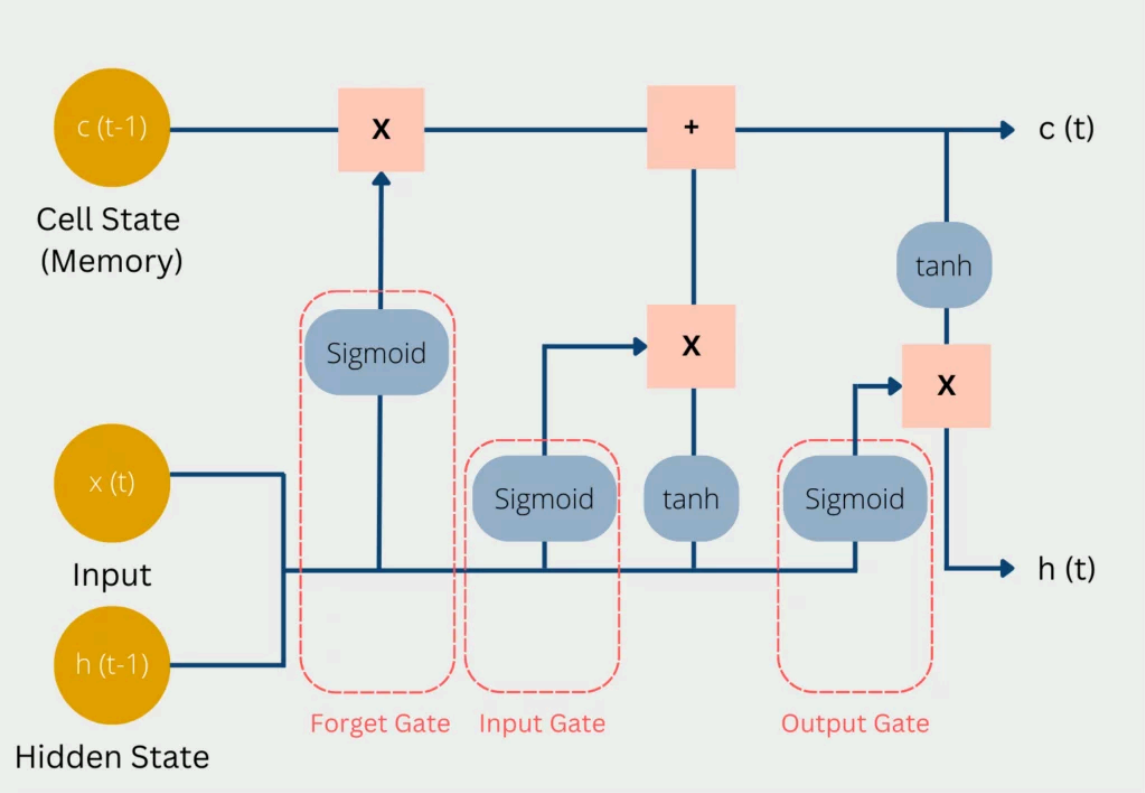
```
        test_output = model(test_seq)
    print(f'RNN Predicted next value: {test_output.item():.4f}')
```

In [5]:
```
train_rnn_model()
```

```
RNN Model Training Completed
RNN Predicted next value: -0.2637
```

LONG SHORT MEMORY NETWORKS



Long Short-Term Memory Networks (LSTMs): LSTMs are a type of recurrent neural network (RNN) architecture that is specifically designed to avoid the long-term dependency problem, which standard RNNs suffer from. They were introduced by Hochreiter and Schmidhuber in 1997 and have been refined and popularized since then.

How LSTMs Work:

LSTMs work by introducing a memory cell and three types of gates (input gate, forget gate, and output gate) to control the flow of information. Here's a breakdown of these components:

1. **Memory Cell**: This cell stores values over arbitrary time intervals. The LSTM can read from, write to, and erase information from the cell, controlled by the gates.
2. **Input Gate**: Controls how much of the new information flows into the memory cell.
3. **Forget Gate**: Controls how much of the past information to forget.
4. **Output Gate**: Controls how much of the information from the memory cell is used to compute the output of the LSTM unit.

Each gate is a neural network layer with its weights, biases, and activation functions. They use the sigmoid function to output a value between 0 and 1, determining how much

information to pass through.

Use Cases of LSTMs:

LSTMs are widely used in various sequence prediction problems due to their ability to remember long-term dependencies. Some common use cases include:

- **Time Series Forecasting**: Predicting stock prices, weather conditions, and other time-dependent data.
- **Natural Language Processing (NLP)**: Language modeling, text generation, machine translation, and speech recognition.
- **Anomaly Detection**: Identifying unusual patterns in data, which is useful in fraud detection and system monitoring.
- **Video Analysis**: Understanding sequences of video frames for tasks such as activity recognition and video captioning.

In [ ]:
```python
#Sample code for an LSTM model

class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTMModel, self).__init__()
        self.hidden_size = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        h0 = torch.zeros(1, x.size(0), self.hidden_size)
        c0 = torch.zeros(1, x.size(0), self.hidden_size)
        out, _ = self.lstm(x, (h0, c0))
        out = self.linear(out[:, -1, :])
        return out

def train_lstm_model():
    # Generate synthetic data
    data = generate_data(100)
    data = [(torch.tensor(x, dtype=torch.float32).unsqueeze(0),
             torch.tensor(y, dtype=torch.float32).unsqueeze(0))
            for x, y in data]
    dataloader = torch.utils.data.DataLoader(data, batch_size=1, shuffle=True)

    # Model, loss, optimizer
    model = LSTMModel(1, 50, 1)
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    # Training loop
    for epoch in range(100):
        for seq, target in dataloader:
            optimizer.zero_grad()
            output = model(seq)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()

    print('LSTM Model Training Completed')

    # Testing the model
    test_seq = torch.tensor(np.sin(np.linspace(0, 100, 100)), dtype=torch.float32).
    with torch.no_grad():
```