Fibonacci Harmonic Trading System: Implementation Guide

This implementation guide provides detailed instructions on setting up, configuring, and deploying the Fibonacci Harmonic Trading System. Follow these steps to get the system up and running in your environment.

Table of Contents

System Requirements

Hardware Requirements

- **Minimum**: 4GB RAM, dual-core CPU
- **Recommended**: 8GB+ RAM, quad-core CPU
- **Optional**: CUDA-compatible GPU for FFT acceleration

Software Requirements

- **Python**: 3.8 or higher
- **Operating System**:
  - Linux (primary development platform)
  - Windows (fully supported)
  - macOS (supported with TA-Lib installation caveats)
- **Database**: SQLite (included)

Installation

1. Clone the Repository

bash Copy

```bash
git clone https://github.com/yourusername/fibonacci-harmonic-trading.git
cd fibonacci-harmonic-trading
```

2. Create a Virtual Environment

bash Copy

```bash
# Create a virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows:
venv\Scripts\activate
# On Linux/macOS:
source venv/bin/activate
```

3. Install TA-Lib

TA-Lib is used for technical indicators and can be challenging to install:

Windows

bash Copy

```bash
# Download and install the wheel from https://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib
pip install path\to\downloaded\TA_Lib-0.4.0-cp38-cp38-win_amd64.whl
```

Linux

bash Copy

```bash
# Install dependencies
sudo apt-get install build-essential
sudo apt-get install libta-lib-dev

# Install the Python wrapper
pip install TA-Lib
```

macOS

bash Copy

```
# Using Homebrew
brew install ta-lib

# Install the Python wrapper
pip install TA-Lib
```

4. Install TvDatafeed

The system uses TvDatafeed to get data from TradingView:

bash Copy

```
pip install --upgrade --no-cache-dir git+https://github.com/rongardF/tvdatafeed.git
```

5. Install Remaining Dependencies

bash Copy

```
# Install all other dependencies
pip install -r requirements.txt
```

6. Install as Development Package

bash Copy

```
# Install the package in development mode
pip install -e .
```

Configuration

The system uses a configuration file in JSON format. Create a `config.json` file in the `config` directory:

json

Copy

```
{
  "general": {
    "default_exchange": "NSE",
    "default_interval": "daily",
    "default_lookback": 5000,
    "symbols_file_path": "./data/symbols/default_symbols.csv",
    "report_dir": "./reports",
    "default_source": "tradingview"
  },
  "performance": {
    "use_gpu": false,
    "max_workers": 5
  },
  "notifications": {
    "telegram_enabled": true,
    "telegram_token": "YOUR_TELEGRAM_BOT_TOKEN",
    "telegram_chat_id": "YOUR_CHAT_ID"
  },
  "tradingview": {
    "username": "YOUR_TRADINGVIEW_USERNAME",
    "password": "YOUR_TRADINGVIEW_PASSWORD"
  },
  "data": {
    "cache_dir": "./data/cache",
    "cache_expiry_days": 30
  },
  "analysis": {
    "fib_cycles": [21, 34, 55, 89, 144, 233],
    "min_period": 10,
    "max_period": 250,
    "power_threshold": 0.2,
    "cycle_tolerance": 0.15,
    "crossover_lookback": 5,
    "detrend_method": "diff",
    "window_function": "hanning",
    "gap_threshold": 0.01
  },
  "web": {
    "host": "0.0.0.0",
    "port": 8050,
    "debug": true
```

```
  }
}
```

Configuration Sections

- **general**: Basic system settings
- **performance**: Performance-related settings
- **notifications**: Telegram notification settings
- **tradingview**: TradingView credentials
- **data**: Data storage and caching settings
- **analysis**: Analysis parameters
- **web**: Web dashboard settings

Data Setup

1. Create Symbol List

Create a CSV file with symbols to analyze. For example, create `data/symbols/default_symbols.csv` with:

csv Copy

```
symbol,exchange,name,sector,industry
NIFTY,NSE,Nifty 50,Index,Index
BANKNIFTY,NSE,Bank Nifty,Index,Index
RELIANCE,NSE,Reliance Industries,Energy,Oil & Gas
TCS,NSE,Tata Consultancy Services,Technology,IT Services
INFY,NSE,Infosys,Technology,IT Services
```

2. Set Up Data Source

The system primarily uses TradingView as a data source. Ensure your credentials are correctly set in the configuration file.

For other data sources:

1. **Yahoo Finance**: No setup required, but add market suffix to symbols (e.g., "RELIANCE.NS")
2. **Local CSV files**: Place CSV files in `data/local` directory with OHLCV format

Running the System

1. Start the Web Dashboard

bash Copy

```
# Start the web dashboard
python run.py
```

This will start the web dashboard on the port specified in the configuration (default: 8050). Open a browser and navigate to `http://localhost:8050` to access the dashboard.

2. Run Single Symbol Analysis

bash Copy

```
# Analyze a single symbol
python run.py --scan NIFTY --interval daily
```

3. Run Batch Analysis

bash Copy

```
# Analyze multiple symbols
python run.py --scan "NIFTY,BANKNIFTY,RELIANCE" --interval daily

# Or use a symbols file
python run.py --scan-file data/symbols/default_symbols.csv --interval daily
```

4. Generate Reports

bash Copy

```
# Generate a comprehensive report
python run.py --report --output reports/daily_report.html
```

Telegram Bot Setup

1. Create a Telegram Bot

1. Open Telegram and search for "@BotFather"
2. Send the command `/newbot`
3. Follow the instructions to create a bot

4. Copy the API token provided

2. Get Your Chat ID

1. Search for "@userinfobot" in Telegram
2. Start a chat with this bot
3. The bot will provide your chat ID

3. Update Configuration

Update the `notifications` section in your configuration:

json Copy

```
"notifications": {
  "telegram_enabled": true,
  "telegram_token": "YOUR_TELEGRAM_BOT_TOKEN",
  "telegram_chat_id": "YOUR_CHAT_ID"
}
```

4. Start the Telegram Bot

bash Copy

```
python run_telegram_bot.py
```

5. Test the Bot

Send the command `/start` to your bot in Telegram. It should respond with a welcome message.

Backtesting

1. Run a Basic Backtest

bash Copy

```
python run_backtest.py --symbol NIFTY --interval daily --start-date 2020-01-01 --end-date 2023-
```

2. Run a Parameter Optimization

bash

Copy

```
python run_backtest.py --symbol NIFTY --interval daily --optimize --params "cycle_tolerance,mi
```

3. Generate a Backtest Report

bash

Copy

```
python run_backtest.py --report backtest_results.json --output backtest_report.html
```

Troubleshooting

Common Issues

TradingView Login Problems

Copy

```
Error: Failed to log in to TradingView
```

Solution:

1. Verify your credentials in the configuration file
2. Try logging in to TradingView in your browser to make sure there are no captchas/verification steps
3. Try using a different data source temporarily (e.g., Yahoo Finance)

TA-Lib Installation Failures

Copy

```
Error: Could not find a version that satisfies the requirement TA-Lib
```

Solution:

1. Install from unofficial wheels (Windows): https://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib
2. Build from source (Linux/macOS): Follow the instructions in the Installation section

Data Cache Issues

Copy

```
Error: Database is locked
```

Solution:

1. Check if another process is using the database
2. Delete the cache file: `rm data/cache/market_data.db`
3. Restart the application

Not Enough Data

Copy

```
Error: Not enough data for analysis
```

Solution:

1. Increase the lookback period in the configuration
2. Try a different data source
3. Check if the symbol exists on the exchange

Logging

The system uses Python's logging module. To increase logging verbosity:

python

Copy

```python
import logging
logging.basicConfig(level=logging.DEBUG)
```

Log files are stored in the `logs` directory. Check these files for detailed error messages.

Extending the System

Adding a New Data Source

1. Create a new module in `data/sources/` directory
2. Implement the required interface methods:
   - `fetch_data(symbol, exchange, interval, lookback)`

- `is_available()`

3. Register your source in `data/fetcher.py`

Creating Custom Indicators

1. Add new indicator functions to `utils/indicators.py`
2. Ensure the function follows the pattern:

python Copy

```python
def custom_indicator(data, param1=default1, param2=default2):
    # Implementation
    return result_series
```

3. Use your indicator in the analysis modules

Developing New Visualization Components

1. Create a new module in `visualization/` directory
2. Implement the plot generation functions
3. Register in the appropriate dashboard component

Adding New Trading Strategies

1. Create a strategy class in `strategies/` directory
2. Implement the required methods:
   - `generate_signals(data, parameters)`
   - `evaluate_performance(signals, data)`
3. Register the strategy in the configuration

Customize the Web Dashboard

1. Modify the layout in `web/layouts.py`
2. Add new callbacks in `web/callbacks.py`
3. Create new visualization components in `web/components/`