SVI Calibration Project

Produced by Hui Yangyifan on 2025/05/13.

This project is to try on the parameters calibration for SVI model, especially for the raw SVI parameterization of total implied variance. Three different calibrators are replicated:

1. SLSQP method (`slsqp`). Reference from *Ferhati, Tahar, titled 'Robust Calibration for SVI Model Arbitrage Free,' available at SSRN 3543766 (2020)*.
2. Differential Evolution method (`de`). Reference from the provided sample code in demo_svi_calibrator.txt.
3. Quasi-Explicit method (`qe`). Reference from *Zeliade Systems, Quasi-explicit calibration of Gatheral's SVI model, Zeliade white paper, 2009*, with sample code on github SVI-Volatility-Surface-Calibration.

For each calibrator, calibration results and visualization are presented; we also produce evaluation metrics and stability analysis.

Reference

Reference papers can be reached in the `reference` folder.

Data

The real data for AAPL.US is in `dataset` folder. Each excel book contains three sheets for one trading day: spot price, forward prices, and implied volatility grids.

We process the raw data books with `src/data_prep.py` , store the data we use in `dataset/data_packs.pkl` . We also produce some raw plots for forward curves and smile curves for initial visualization, saved in `results/forward_curve` and `results/raw_smile` .

Codes

All the source codes for this project can be reached in the `src` folder.

Configuration

See `config.py`.

- Prepare directory paths to use.
- Retrieve tha list of trading dates of the real data.
- Prepare a class `SuppressOutput`, which is the context manager to suppress stdout, used in optimization framework for `verbose=False`.

Data Preparation

See `data_prep.py`.

- Read and process the raw data books from `dataset` folder, save processed data as dictionaries in `dataset/data_packs.pkl`.
- Produce some raw plots for forward curves and smile curves for initial visualization, saved in `results/forward_curve` and `results/raw_smile`.

Visualization

See `plot_curves.py`. Include functions:

- `plot_forward_curve`: Plot the forward curve for a given date and spot price.
- `plot_raw_smile`: Plot the implied volatility smile for a given date and expiry, i.e., plot the implied volatility against the strikes.
- `plot_single_fit`: Plot the SVI model fit against market data of a slice, i.e., one maturity.
- `plot_multi_fit`: Plot the SVI model fit against market data of a surface, i.e., multiple maturities.
- `plot_svi_surface`: Plot a 3D surface of the SVI model across multiple maturities.
- `plot_all_residuals`: Plot SVI model fit residuals for all maturities in a grid layout with residual panels below each plot.

Calibrators

Three calibrators are packed in classes, saved in different files.

- SLSQP method (`slsqp`) replicated in `svi_calibrator_slsqp.py`.
- Differential Evolution method (`de`) replicated in `svi_calibrator_de.py`
- Quasi-Explicit method (`qe`) replicated in `svi_calibrator_qe.py`.

Remark: Due to time limit, main efforts are put in `svi_calibrator_slsqp.py`, so it is more concise and reusable. The other two methods are generated by *AI coders* with appropriate prompts and imitation of

the `slsqp` class, fewer modifications and verifications are made for them manually. But all of the three calibrators can run smoothly to produce required results. The latter two needs further work.

Some of the main methods in the classes:

- Arbitrage free conditions: `butterfly_arbitrage_constraint` , `calendar_spread_constraint` .
- Standard calibration: `calibrate_single` , `calibrate_multi` , using initial values and methodologies given in the reference. However, the convergence may not be guaranteed in that case!
- Robust calibration: `robust_calibration` , `robust_calibrate_multi` , which handles potential convergence issues by approaches like
  - Multiple calibration attempts with different initial guesses
  - Error scaling by strike (giving more weight to ATM points)
  - Parameter bounds to prevent extreme values
  - Fallback options when optimization fails
- Result storage: `get_parameters_dataframe` .

Evaluation and Stability Analysis

See the class `SVICalibrationEvaluator` in `evaluation_metrics.py` .

- Evaluate calibration quality by some quantitative measures or metrics:
  - Global fitness: RMSE, MAE, R squared.
  - Local fitness: Region-specific RMSE, ATM/left wing/right wing.
  - Arbitrage check.
- Evaluate residuals for all maturities, draw a plot with subplots over all maturities and calculate some summary statistics.

See the class `SVIStabilityAnalyzer` in `stability_analysis.py` .

- Parameter sensitivity analysis: test parameter sensitivity to 1% data perturbation, report Coefficient of Variation (CV) for different parameters.
- Maturity stability analysis: test parameter changes between consecutive maturities, report the relative changes.
- Overall summary: take means over different maturities or metrics.

Workflow

See `workflow.py` for a whole running process. Procedures are as follows:

- **Data Loading**: Load pre-processed data from pickle files for different dates.

- **Calibration**: Use assigned `method` for SVI model fitting across multiple maturities, all use **robust calibrations**.
- **Visualization**: Create plots of fitted curves, volatility surfaces, and residual analyses.
- **Evaluation**: Calculate quality metrics like RMSE, R squared, and arbitrage risk indicators.
- **Stability Analysis**: Assess parameter sensitivity and stability across maturities.
- **Result Storage**: Save all parameters and metrics to CSV files in organized directories.

Use `multiprocessing` to run results over different dates and methods.

Stress Data test, 2025-04-08

Show a demo of calibration results in `Test_2025-04-08.ipynb` .

Results

Results, including plots and dataframes, are saved in `results` folder. The sub-folders classify the results by different calibrators and different trading dates.

Summary on some results (may not be comprehensive or general):

- We take $\epsilon = 1e-3$, which is not too small, otherwise the results can be very unstable.
- All three calibrators can successfully produce *arbitrage free* results by robust calibration. Across all maturities, the average RMSE is low and average R squared is high.
- `slsqp` seems to produce the most stable parameters. `de` also produces stable parameters. However, `qe` produces extremely unstable parameters with extremely high stability score.
- We temporarily have no idea on the reasons behind. We also need to check the correctness of coding and formulation.

Problems To-do List

Due to time limit, there are some problems to tackle left, including but not limited to:

- Further analysis of the results. Better to have a case-by-case analysis on results of different trading dates and different smile curves.
- Interpolation or filling or use a different source for complement, for those missing values in implied volatility grids.
- Further work on three calibrators:
  - Further manually check and modify the codes details.
  - Further verify **robust calibration** approaches.

    - Pack the common functions reusable out in a `utils.py` file, reformulate the three classes to make them more concise.
- Further work on evaluation and stability analysis:
    - Modify some quality measures：for example, the definition for **left/right wing** is not accurate, the given data has few right wings data.
    - Modify sensitivity test: the current one can be not that stable itself.
    - Add more sensitivity metrics or tests.
- More numerical experiments on fitting.
- Workflow procedures standardization:
    - Import loggings/logger writers.
    - Further modify comments.