PyForecast

Oil & Gas Decline Curve Analysis
User Guide

Automated Hyperbolic Decline Curve Fitting
with Regime Detection and Data Validation

January 2026

Table of Contents

1. Introduction
2. Quick Start
3. Decline Curve Theory
4. Fitting Parameters
    4.1 B-Factor (Hyperbolic Exponent)
    4.2 Initial Decline Rate (Di)
    4.3 Terminal Decline (Dmin)
5. Regime Detection
6. Configuration
7. Data Validation
8. Output Formats
9. Command Reference
10. Programmatic Usage
Appendix A: Parameter Quick Reference

1. Introduction

PyForecast is an automated decline curve analysis (DCA) tool for oil and gas production forecasting. It fits hyperbolic decline models to historical production data and generates forecasts suitable for reserves estimation and economic analysis.

Key Features

- Hyperbolic decline curve fitting with configurable b-factor bounds
- Automatic regime change detection (workovers, refracs, RTPs)
- Per-product configuration (oil, gas, water)
- Recency-weighted fitting to favor recent production trends
- ARIES-compatible forecast export formats
- Comprehensive data validation and quality checks
- Batch processing with parallel execution
- Interactive visualization with semi-log plots

Supported Input Formats

- Enverus production exports (CSV/Excel)
- ARIES production data format
- Auto-detection of file format

2. Quick Start

Installation

```
pip install pyforecast
```

Basic Usage

Process production data with default settings:

```
pyforecast process production.csv -o output/
```

This will:

1. Load wells from the input file
2. Fit decline curves for oil, gas, and water
3. Export ARIES-format forecasts
4. Generate interactive plots
5. Create a validation report

Custom Configuration

Generate a configuration file to customize fitting parameters:

```
pyforecast init -o config.yaml
```

Then run with your configuration:

```
pyforecast process production.csv -o output/ --config config.yaml
```

Single Well Analysis

Plot and analyze a single well:

```
pyforecast plot production.csv --well-id "42-001-12345" --product oil
```

3. Decline Curve Theory

The Hyperbolic Decline Equation

PyForecast uses the Arps hyperbolic decline equation, the industry standard for production forecasting:

$$q(t) = qi / (1 + b * Di * t)^(1/b)$$

Where:

- q(t) = Production rate at time t
- qi = Initial production rate at t=0
- Di = Initial decline rate (fraction/month)
- b = Hyperbolic exponent (dimensionless)
- t = Time (months)

Decline Types

The b-factor determines the decline behavior:

b Value	Decline Type	Characteristics
b = 0	Exponential	Constant % decline; steepest; most conservative
0 < b < 1	Hyperbolic	Declining % decline rate; typical for most wells
b = 1	Harmonic	Decline rate proportional to rate; slowest standard
b > 1	Super-harmonic	Very slow decline; transient flow; use caution

Industry Guidance

Most conventional oil wells have b-factors between 0.3 and 0.8. Unconventional (tight oil/shale) wells often show b > 1 in early time, transitioning to lower values as boundary-dominated flow develops.

4. Fitting Parameters

PyForecast allows fine-grained control over decline curve fitting through per-product configuration. Understanding these parameters is essential for generating reliable forecasts.

4.1 B-Factor (Hyperbolic Exponent)

The b-factor controls the curvature of the decline. PyForecast constrains b within configurable bounds to prevent unrealistic forecasts.

Configuration

```
oil:
  b_min: 0.01    # Lower bound (near-exponential)
  b_max: 1.5     # Upper bound (super-harmonic)

gas:
  b_min: 0.01
  b_max: 2.0     # Gas often shows higher b in tight reservoirs
```

Recommended Ranges by Play Type

Play Type	Typical b Range	Recommended b_max
Conventional oil	0.3 - 0.8	1.0
Conventional gas	0.4 - 0.9	1.2
Tight oil (Bakken, etc.)	0.8 - 1.5	1.5 - 2.0
Shale gas (Marcellus, etc.)	1.0 - 2.0	2.0 - 2.5
CBM / Coal seam gas	0.5 - 1.0	1.2

> **Important**
>
> High b-factors (>1.5) can lead to optimistic forecasts. Always validate fits visually and consider using terminal decline (Dmin) as a constraint.

What Happens at the Bounds

- b at b_min: Near-exponential decline. May indicate mature well or data issues.
- b at b_max: May indicate transient flow or fitting to noise. Review carefully.

4.2 Initial Decline Rate (Di)

The initial decline rate represents how fast production is declining at the start of the forecast period. PyForecast reports Di as a monthly fraction but you can also think of it annually.

Unit Conversion

Monthly Di	Annual Di	Interpretation
0.01	12%	Very slow decline
0.05	60%	Moderate decline
0.08	96%	Steep decline (typical tight oil)
0.10	120%	Very steep (early-time transient)

PyForecast automatically estimates Di from the data using log-linear regression as an initial guess, then refines it through optimization.

4.3 Terminal Decline (Dmin)

Terminal decline (Dmin) sets a floor on the decline rate. When the instantaneous decline drops to Dmin, the model switches to exponential decline at that rate. This prevents unrealistically long tails in high-b forecasts.

Configuration

```
oil:
  dmin: 0.06    # 6% annual terminal decline

gas:
  dmin: 0.05    # 5% annual terminal decline
```

Typical Values

Fluid	Typical Dmin	Rationale
Oil	5-8%	Based on field decline statistics
Gas	4-6%	Gas wells often decline more slowly late-life
Water	5-10%	Varies widely by drive mechanism

5. Regime Detection

Wells often experience production increases due to workovers, refracs, or return-to-production (RTP) events. PyForecast automatically detects these regime changes and fits only the most recent decline period.

How It Works

1. Fits a trend line to recent production history
2. Extrapolates the trend forward
3. Detects when production exceeds the trend by a threshold
4. Confirms the change is sustained (not a spike)
5. Resets the fitting window to start from the new regime

Configuration

```
regime:
  threshold: 1.0         # 100% increase triggers detection
  window: 6              # Months of data for trend fitting
  sustained_months: 2    # Months elevation must persist
```

Parameter Guidance

Parameter	Low Value	High Value
threshold	More sensitive (0.5)	Less sensitive (1.5)
window	Recent trend only (3)	Longer history (12)
sustained_months	Quick trigger (1)	Confirmed only (3)

> **Tip**
>
> When regime detection triggers, the fit results will show regime_start_idx > 0, indicating the month index where the new regime began. The forecast is based only on data from that point forward.

Recency Weighting

In addition to regime detection, PyForecast applies exponential decay weighting to favor recent production data. This helps the model track current trends.

```
fitting:
  recency_half_life: 12.0  # Half-life in months
```

With a 12-month half-life, data from 12 months ago has half the weight of current data. Lower values weight recent data more aggressively.

6. Configuration

PyForecast uses YAML configuration files for reproducible, version-controlled analysis settings. Generate a template with all options:

```
pyforecast init -o pyforecast.yaml
```

Complete Configuration Reference

```
# Per-product decline curve parameters
oil:
  b_min: 0.01      # Minimum b-factor
  b_max: 1.5       # Maximum b-factor
  dmin: 0.06       # Terminal decline (annual)

gas:
  b_min: 0.01
  b_max: 1.5
  dmin: 0.06

water:
  b_min: 0.01
  b_max: 1.5
  dmin: 0.06

# Regime change detection
regime:
  threshold: 1.0          # Min increase to trigger (1.0 = 100%)
  window: 6               # Trend fitting window (months)
  sustained_months: 2     # Confirmation period

# General fitting parameters
fitting:
  recency_half_life: 12.0  # Weighting half-life (months)
  min_points: 6            # Minimum data points required

# Output options
output:
  products: [oil, gas, water]  # Products to forecast
  plots: true                   # Generate well plots
  batch_plot: true              # Generate overlay plot
  format: ac_economic           # Export format

# Data validation thresholds
validation:
  max_oil_rate: 50000      # Max rate before warning
  max_gas_rate: 500000
  outlier_sigma: 3.0       # Outlier detection threshold
  min_r_squared: 0.5       # Minimum fit quality
```

7. Data Validation

PyForecast includes comprehensive data validation to catch quality issues before they impact your forecasts. Validation runs automatically during batch processing.

Validation Categories

Category	Checks
DATA_FORMAT	Negative values, extreme rates, date issues
DATA_QUALITY	Gaps, outliers, shut-ins, low variability
FITTING_PREREQ	Sufficient data, declining trend
FITTING_RESULT	R-squared, b-factor bounds, decline rate

Key Error Codes

Code	Severity	Description
IV001	ERROR	Negative production values
IV002	WARNING	Values exceed configured maximum
DQ001	WARNING	Data gaps detected
DQ002	WARNING	Outliers detected
FP001	ERROR	Insufficient data points
FP002	WARNING	Increasing trend (not declining)
FR001	WARNING	Poor fit quality (low R-squared)
FR004	WARNING	B-factor at upper bound

After processing, check validation_report.txt in the output directory for detailed issue descriptions and resolution guidance.

8. Output Formats

ARIES AC Economic Format

The default export format (ac_economic) is compatible with ARIES economic evaluation software. It includes monthly forecast volumes for import into AC_ECONOMIC sections.

```
pyforecast process data.csv -o output/ --format ac_economic
```

ARIES AC Forecast Format

Alternative format for direct forecast import with decline parameters.

```
pyforecast process data.csv -o output/ --format ac_forecast
```

Output Directory Structure

```
output/
  ac_economic.csv        # Forecast export file
  validation_report.txt  # Data quality issues
  errors.txt             # Processing errors (if any)
  plots/
    WELL001_oil.html     # Interactive well plots
    WELL001_gas.html
    batch_oil.html       # Multi-well overlay plot
    ...
```

Interactive Plots

PyForecast generates interactive HTML plots using Plotly. Features include:

- Semi-log scale (standard for decline analysis)
- Historical production with fitted curve overlay
- 30-year forecast projection
- Zoom, pan, and hover for data values
- Regime change indicator (if detected)

9. Command Reference

pyforecast process

Process production data and generate forecasts.

```
pyforecast process <files> [options]

Arguments:
  files              Input CSV/Excel file(s)

Options:
  -o, --output       Output directory (default: output)
  -c, --config       YAML configuration file
  -p, --product      Products to forecast (oil, gas, water)
  -w, --workers      Parallel workers (default: auto)
  --no-plots         Skip individual well plots
  --no-batch-plot    Skip multi-well overlay plot
  --format           Export format (ac_economic, ac_forecast)
```

pyforecast plot

Analyze and plot a single well.

```
pyforecast plot <file> [options]

Options:
  --well-id          Well identifier to plot
  -p, --product      Product to plot (default: oil)
  -o, --output       Save plot to file (default: show in browser)
  -c, --config       YAML configuration file
```

pyforecast init

Generate a default configuration file.

```
pyforecast init [options]

Options:
  -o, --output       Output file path (default: pyforecast.yaml)
```

pyforecast info

Display information about a production data file.

```
pyforecast info <file>

Shows:
  - Detected format (Enverus, ARIES)
  - Number of wells
  - Date range
  - Available columns
```

10. Programmatic Usage

Loading Data

```
from pyforecast.data.base import load_wells

wells = load_wells("production.csv")
for well in wells:
    print(f"{well.well_id}: {well.production.n_months} months")
```

Fitting a Single Well

```
from pyforecast.core.fitting import DeclineFitter, FittingConfig

# Configure fitting parameters
config = FittingConfig(
    b_min=0.01,
    b_max=1.5,
    dmin_annual=0.06,
    recency_half_life=12.0,
)

# Fit decline curve
fitter = DeclineFitter(config)
result = fitter.fit(
    t=well.production.time_months,
    q=well.production.oil
)

# Access results
print(f"qi: {result.model.qi:.1f} bbl/mo")
print(f"Di: {result.model.di * 12:.1%}/year")
print(f"b:  {result.model.b:.3f}")
print(f"R2: {result.r_squared:.3f}")
```

Using Configuration Files

```
from pyforecast.config import PyForecastConfig
from pyforecast.core.fitting import FittingConfig

# Load configuration
config = PyForecastConfig.from_yaml("config.yaml")

# Get per-product fitting config
oil_config = FittingConfig.from_pyforecast_config(config, "oil")
gas_config = FittingConfig.from_pyforecast_config(config, "gas")
```

Batch Processing

```
from pyforecast.batch.processor import BatchProcessor, BatchConfig
from pyforecast.config import PyForecastConfig

config = PyForecastConfig.from_yaml("config.yaml")

batch_config = BatchConfig(
    products=["oil", "gas"],
    pyforecast_config=config,
    output_dir="output",
)

processor = BatchProcessor(batch_config)
result = processor.run(["data1.csv", "data2.csv"])
```

```
print(f"Successful: {result.successful}")
print(f"Failed: {result.failed}")
```

Appendix A: Parameter Quick Reference

Fitting Parameters

Parameter	Default	Description
b_min	0.01	Minimum b-factor (near-exponential)
b_max	1.5	Maximum b-factor (super-harmonic)
dmin	0.06	Terminal decline rate (annual)
recency_half_life	12.0	Weighting half-life (months)
min_points	6	Minimum data points required

Regime Detection Parameters

Parameter	Default	Description
threshold	1.0	Min increase to trigger (fraction)
window	6	Trend fitting window (months)
sustained_months	2	Confirmation period (months)

Validation Parameters

Parameter	Default	Description
max_oil_rate	50,000	Max oil rate warning (bbl/mo)
max_gas_rate	500,000	Max gas rate warning (mcf/mo)
outlier_sigma	3.0	Outlier detection threshold
min_r_squared	0.5	Minimum fit quality
gap_threshold	2	Gap detection (months)

Typical B-Factor Ranges

Formation Type	Oil b Range	Gas b Range
Conventional	0.3 - 0.8	0.4 - 0.9
Tight / Unconventional	0.8 - 1.5	1.0 - 2.0
Shale	1.0 - 2.0	1.2 - 2.5