

# Financial Instruments Toolbox™ Release Notes



# MATLAB®



## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
1 Apple Hill Drive  
Natick, MA 01760-2098

### *Financial Instruments Toolbox™ Release Notes*

© COPYRIGHT 2012–2022 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## R2022a

<b>Instrument Pricing: Model trinomial trees with finpricer</b> .....	1-2
<b>Equity Instruments: Calculate rates and pricing for commodity futures, equity index futures, and FX futures</b> .....	1-2
<b>Interest-Rate Instrument: Compute prices for Treasury bond futures</b> ...	1-2
<b>Interest-Rate Instrument: Calibrate option pricing model in the Live Editor</b> .....	1-2
<b>Deep Learning Example: Price complex options using Deep Learning Toolbox</b> .....	1-2
<b>Numerix C++ interface</b> .....	1-2

## R2021b

<b>Interest-Rate Instruments: SABR-Libor market model</b> .....	2-2
<b>Equity Instruments: Price cliquet options</b> .....	2-2
<b>Monte Carlo Simulation: Price interest-rate instruments</b> .....	2-2
<b>Equity Instrument: Partial lookback options</b> .....	2-2
<b>Instrument Pricing: fininstrument supports vector of instruments</b> .....	2-2
<b>Interest-Rate Instrument: Enhanced swap and bond curve construction</b> .....	2-4
<b>Interest-Rate Instrument: LIBOR transition to ARR securities</b> .....	2-4
<b>Reinforcement Learning Example: Price and hedge instruments using Reinforcement Learning Toolbox</b> .....	2-4

## R2021a

<b>Equity Tree Models: Price vanilla, lookback, barrier, or Asian instruments using Cox-Ross-Rubinstein (CRR), Equal Probability (EQP), and Leisen-Reimer (LR) lattice models</b> .....	3-2
<b>Hybrid Instruments: Price convertible bonds with Crank-Nicolson method</b> .....	3-2
<b>Inflation Instruments: Build inflation curves and price inflation bonds and swaps</b> .....	3-2
<b>Bachelier Model: Price vanilla, spread, and binary instruments</b> .....	3-2
<b>derivtool removal</b> .....	3-2

## R2020b

<b>Pricing and Valuation: Variance swaps</b> .....	4-2
<b>Vanna-Volga Method: Price vanilla, barrier, double barrier, touch, and double touch instruments</b> .....	4-2
<b>Touch, Double Touch, and Binary Instruments: Price instruments using Black-Scholes or Asset Monte Carlo methods</b> .....	4-2
<b>Double-Barrier Instrument: Price instrument using finite difference or Ikeda-Kunitomo methods</b> .....	4-2
<b>Call and Put Options: Price callable and puttable float bonds, float bond options, caps, floors, and swaptions by using instruments</b> .....	4-2
<b>Finite Difference Method: Price vanilla instruments with Bermudan exercise</b> .....	4-2
<b>Monte Carlo Simulation: Price equity, FX, and commodity instruments</b> .....	4-2

## R2020a

<b>Pricing and Valuation: Price various types of financial instruments individually or as a portfolio using new object-oriented framework</b> ...	5-2
Curve Objects .....	5-2

## R2019b

<b>Interest-Rate Instruments: Compute prices and option-adjusted spread of amortizing bonds with embedded options using interest-rate tree models</b> .....	<b>6-2</b>
<b>Interest-Rate Instruments: Calculate probability of exercise options on embedded bonds and bonds</b> .....	<b>6-2</b>
<b>Equity and Energy Instruments: Compute prices and sensitivities of one-touch and double one-touch options using the Black-Scholes option pricing model</b> .....	<b>6-2</b>

## R2019a

<b>Normal SABR Model: Support the computation of Normal (Bachelier) volatility for cap or floor volatility stripping</b> .....	<b>7-2</b>
<b>Finite-Difference Methods: Compute prices and sensitivities of double barrier options using the Alternating Direction Implicit (ADI) and Crank-Nicolson methods</b> .....	<b>7-2</b>
<b>Monte Carlo Simulation: Compute prices and sensitivities of double barrier options using the Black-Scholes option pricing model</b> .....	<b>7-2</b>
<b>Vanilla European Options: Compute prices and sensitivities using the Bates and Merton<sup>76</sup> models with finite differences</b> .....	<b>7-2</b>

## R2018b

<b>Normal SABR Model: Compute implied Normal (Bachelier) volatility and sensitivity by the SABR model</b> .....	<b>8-2</b>
<b>Finite-Difference Methods: Calculate option prices by the Heston and local volatility models using the Alternating Direction Implicit (ADI) and Crank-Nicolson methods</b> .....	<b>8-2</b>
<b>Implied Volatility: Improve performance of the <code>impvbybls</code> and <code>impvbyblk</code> functions when using the Jäckel 2016 method</b> .....	<b>8-2</b>

## R2018a

<b>Vanilla European Options: Compute prices and sensitivities using Heston, Bates, and Merton<sup>76</sup> models with FFT and numerical integration . . . .</b>	<b>9-2</b>
<b>Cox-Ingersoll-Ross Lattice Trees: Calculate prices and sensitivities of bonds, caps, floors, swaps, and options . . . . .</b>	<b>9-2</b>
<b>Asian Options: Use Haug, Haug, Margrabe model for discrete arithmetic fixed Asian options and Turnbull-Wakeman model for continuous arithmetic fixed options . . . . .</b>	<b>9-3</b>

## R2017b

<b>Interest-Rate Instruments: Price swaptions with resettable legs and different basis conventions using Black, Normal, and lattice (tree-based) models . . . . .</b>	<b>10-2</b>
<b>Interest-Rate Instruments: Use Hull-White calibration routines for Shifted Black and Normal models . . . . .</b>	<b>10-2</b>

## R2017a

<b>Interest-Rate Instruments: Price interest rate options with negative rates using normal volatility model and shifted SABR model . . . . .</b>	<b>11-2</b>
<b>Equity Instruments: Price American vanilla options using Barone-Adesi and Whaley model . . . . .</b>	<b>11-2</b>
<b>bkcaplet and bkfloorlet removal . . . . .</b>	<b>11-2</b>

## R2016b

<b>Equity Instruments: Price barrier options with closed form, Crank-Nicolson method, and Monte Carlo simulation . . . . .</b>	<b>12-2</b>
<b>Equity Instruments: Price European options with finite differences method . . . . .</b>	<b>12-2</b>
<b>Hybrid Instruments: Price convertible bonds with a default risk and recovery rate using standard and implied trinomial trees . . . . .</b>	<b>12-2</b>

<b>Numerix CAIL Engine: Access the Numerix Engine directly from MATLAB using an updated API</b> .....	12-2
<b>Functions moved from Financial Instruments Toolbox to Financial Toolbox</b> .....	12-2
<b>help fininstdemos removal</b> .....	12-3
<b>bkcaplet and bkfloorlet removal</b> .....	12-3

## R2016a

<b>Cap and Floor Instruments: Volatility stripping</b> .....	13-2
<b>Swap Instruments: Pricing cross-currency, fixed-fixed, and float-float swaps</b> .....	13-2
<b>cbprice removal</b> .....	13-2

## R2015b

<b>Hybrid Instruments: Price convertible bonds using standard and implied trinomial trees</b> .....	14-2
<b>Equity Instruments: Price equity derivatives using a standard trinomial tree</b> .....	14-2
<b>Simple Interest Convention: Calculate zero curves, forward curves, discount curves, rates, and bootstrapping using simple interest</b> . . . .	14-2
<b>cbprice removal</b> .....	14-3

## R2015a

<b>Price convertible bonds using CRR and EQP lattice models</b> .....	15-2
<b>Collateral-level computation from credit exposure simulations</b> .....	15-2
<b>Wrong-way risk example</b> .....	15-2

<b>Pricing functionality for forward options</b> .....	<b>16-2</b>
<b>Amortizing caps and floors pricing using lattice models</b> .....	<b>16-2</b>
<b>Power price simulation example</b> .....	<b>16-2</b>
<b>Hull-White single-factor model calibration using volatility surface</b> ....	<b>16-2</b>
<b>SABR option greeks computation</b> .....	<b>16-2</b>
<b>Amortizing caps and floors pricing using closed form solutions (Black or Linear Gaussian two-factor models)</b> .....	<b>16-2</b>
<b>Asian options pricing example</b> .....	<b>16-2</b>
<b>Numerix CrossAsset Integration Layer (CAIL) 11 example</b> .....	<b>16-3</b>

<b>Dual curve construction</b> .....	<b>17-2</b>
<b>Dual curve pricing of caps, floors, and swaptions using the Black model</b> .....	<b>17-2</b>
<b>Dual curve pricing of swaps and floating-rate notes</b> .....	<b>17-2</b>
<b>Monte Carlo and analytical pricing of lookback options</b> .....	<b>17-2</b>
<b>Implied Black volatility computation for the SABR stochastic volatility model</b> .....	<b>17-2</b>
<b>User-specified simulation paths for Longstaff-Schwartz pricing functions</b> .....	<b>17-3</b>
<b>creditexposures function to compute credit exposures from mark-to-market OTC contract values</b> .....	<b>17-3</b>
<b>exposureprofiles function to derive credit exposure profiles from credit exposures</b> .....	<b>17-3</b>
<b>Enhanced pricing functions for instruments and portfolios with cash flows between tree levels</b> .....	<b>17-3</b>
<b>Swing option pricing example</b> .....	<b>17-3</b>



<b>Support for Numerix CrossAsset Integration Layer (CAIL) API</b> . . . . .	<b>18-2</b>
<b>Kirk's approximation and Bjerksund-Stensland closed-form pricing models for spread options</b> . . . . .	<b>18-2</b>
<b>Finite difference and Monte Carlo simulation pricing for American spread options</b> . . . . .	<b>18-2</b>
<b>Levy and Kemna-Vorst closed-form pricing and Monte Carlo simulation pricing for Asian options</b> . . . . .	<b>18-2</b>
<b>Additional CDS option pricing functionality for index swaptions</b> . . . . .	<b>18-3</b>
<b>Pricing functions for vanilla options using Monte Carlo simulation</b> . . .	<b>18-3</b>
<b>Hedging strategies using spread options example</b> . . . . .	<b>18-3</b>
<b>Pricing European and American spread options example</b> . . . . .	<b>18-3</b>
<b>First-to-default (FTD) swaps example</b> . . . . .	<b>18-3</b>
<b>New function for risky present value of a basis point</b> . . . . .	<b>18-3</b>
<b>optimoptions support</b> . . . . .	<b>18-4</b>
<b>Functions moved from Financial Instruments Toolbox to Financial Toolbox</b> . . . . .	<b>18-4</b>

<b>Pricing functions for options on floating-rate notes (FRNs)</b> . . . . .	<b>19-2</b>
<b>Pricing functions for FRNs with embedded options</b> . . . . .	<b>19-2</b>
<b>Performance enhancements in implied volatility calculations</b> . . . . .	<b>19-2</b>
<b>Calibration and Monte Carlo simulation of single-factor and multifactor interest-rate models, including Hull-White, Linear Gaussian, and LIBOR Market Models</b> . . . . .	<b>19-2</b>

<b>Merge of Fixed-Income Toolbox and Financial Derivatives Toolbox to Financial Instruments Toolbox</b> .....	<b>20-2</b>
<b>Cap and floor floating-rate note pricing using trees</b> .....	<b>20-2</b>
<b>Forward-swap pricing using trees or term structure</b> .....	<b>20-2</b>
<b>Functions for fitting and extracting calibrated parameters from IRFunctionCurve objects</b> .....	<b>20-2</b>
<b>LIBOR market model example</b> .....	<b>20-3</b>
<b>Counterparty credit risk example</b> .....	<b>20-3</b>
<b>Conversion of error and warning message identifiers</b> .....	<b>20-3</b>

# R2022a

---

**Version: 3.4**

**New Features**

**Bug Fixes**

## **Instrument Pricing: Model trinomial trees with finpricer**

The `AssetTree` pricer supports the name-value argument option for 'PricingMethod' using a "StandardTrinomial" lattice tree model. You can use the "StandardTrinomial" lattice tree model option to price Vanilla, Barrier, Lookback, and Asian instruments.

## **Equity Instruments: Calculate rates and pricing for commodity futures, equity index futures, and FX futures**

The `CommodityFuture`, `FXFuture`, and `EquityIndexFuture` instruments use a `ratecurve` object and a `Future` pricing method. You can compute the fair delivery price for an underlying asset using `fairdelivery` and compute the cash settlement using `cashsettle`.

## **Interest-Rate Instrument: Compute prices for Treasury bond futures**

The `BondFuture` instrument supports futures and forwards using a `ratecurve` object and a `Future` pricing method. You can compute the fair delivery price for an underlying asset using `fairdelivery` and compute the cash settlement using `cashsettle`. The "Select Cheapest-to-Deliver Bond Using `BondFuture` Instrument" example demonstrates the workflow for selecting the cheapest-to-deliver (CTD) bond for Treasury bond future contracts using a `BondFuture` instrument.

## **Interest-Rate Instrument: Calibrate option pricing model in the Live Editor**

The **Calibrate Pricing Model** live task supports the workflow to calibrate an option pricing using a Heston, Bates, or Merton model for an equity, foreign exchange (FX), or commodity call or put option.

## **Deep Learning Example: Price complex options using Deep Learning Toolbox**

The "Use Deep Learning to Approximate Barrier Option Prices with Heston Model" example demonstrates how to use Deep Learning Toolbox™ to train a network and obtain predictions on barrier option prices with a Heston model.

## **Numerix C++ interface**

The `numerixCrossAsset` object supports a C++ SDK API on Windows®.

# R2021b

---

**Version: 3.3**

**New Features**

**Bug Fixes**

## Interest-Rate Instruments: SABR-Libor market model

The `SABRBraceGatarekMusiel` model supports a SABR-Libor market model.

## Equity Instruments: Price cliquet options

Create and compute the price of a `Cliquet` instrument using a `BlackScholes`, `Merton`, `Bates`, or `Heston` model and an `AssetMonteCarlo` or `Rubinstein` pricing method.

## Monte Carlo Simulation: Price interest-rate instruments

The following interest-rate instruments support Monte Carlo simulation:

- `FixedBond`
- `FloatBond`
- `Cap`
- `Floor`
- `Swap`
- `Swaption`
- `FixedBondOption`
- `FloatBondOption`
- `OptionEmbeddedFixedBond`
- `OptionEmbeddedFloatBond`

The following models support Monte Carlo simulation:

- `HullWhite`
- `BraceGatarekMusiel`
- `SABRBraceGatarekMusiel`
- `LinearGaussian2F`

The `IRMonteCarlo` pricer supports Monte Carlo simulation.

## Equity Instrument: Partial lookback options

You can create and price a `PartialLookback` instrument using a `BlackScholes`, `Merton`, `Heston`, or `Bates` model and `HeynenKat` or `AssetMonteCarlo` pricer.

## Instrument Pricing: `fininstrument` supports vector of instruments

You can use `fininstrument` to create a vector of instruments for each of the following instrument types:

For interest-rate instruments:

- `Deposit`
- `FRA`

- 
- FixedBond
  - FixedBondOption
  - FloatBond
  - FloatBondOption
  - OptionEmbeddedFixedBond
  - OptionEmbeddedFloatBond
  - Swap
  - Cap
  - Floor
  - Swaption
  - STIRFuture
  - OISFuture
  - OvernightIndexedSwap

For inflation instruments:

- InflationBond
- YearYearInflationSwap
- ZeroCouponInflationSwap
- ConvertibleBond

For equity, commodity, FX, or commodity instruments:

- Vanilla
- Lookback
- PartialLookback
- Barrier
- DoubleBarrier
- Asian
- Spread
- Touch
- DoubleTouch
- Binary
- Cliquet
- VarianceSwap
- ConvertibleBond

For credit derivative instruments:

- CDS
- CDSOption

## **Interest-Rate Instrument: Enhanced swap and bond curve construction**

You can create `STIRFuture`, `OISFuture`, and `OvernightIndexedSwap` instruments and then bootstrap a `ratecurve` object for one or more of these instruments using `irbootstrap`. In addition, you can price one or more of these instruments using the `Discount` pricing method.

## **Interest-Rate Instrument: LIBOR transition to ARR securities**

You can create an `OISFuture` instrument for alternative reference rate (ARR) securities that are compliant with standards from the International Organization of Securities Commissions (IOSCO) like SOFR, EONIA SONIA, SARON, and TONAR futures. For more information on the LIBOR transition to an ARR, see `Compute LIBOR Fallback`.

## **Reinforcement Learning Example: Price and hedge instruments using Reinforcement Learning Toolbox**

The `Hedging an Option Using Reinforcement Learning Toolbox` example demonstrates an approach to learn an optimal option hedging policy that outperforms the traditional Black-Scholes-Merton (BSM) model approach.



# R2021a

---

**Version: 3.2**

**New Features**

**Bug Fixes**

**Version History**

## **Equity Tree Models: Price vanilla, lookback, barrier, or Asian instruments using Cox-Ross-Rubinstein (CRR), Equal Probability (EQP), and Leisen-Reimer (LR) lattice models**

Cox-Ross-Rubinstein (CRR), Equal Probability (EQP), and Leisen-Reimer (LR) lattice models support pricing for Vanilla, Lookback, Barrier, or Asian instruments using a BlackScholes model and an AssetTree pricer.

## **Hybrid Instruments: Price convertible bonds with Crank-Nicolson method**

You can create a ConvertibleBond instrument to use with a BlackScholes model and FiniteDifference pricer.

## **Inflation Instruments: Build inflation curves and price inflation bonds and swaps**

You can create an inflationcurve object for use with InflationBond, YearYearInflationSwap, or ZeroCouponInflationSwap instrument objects. You can price the inflation instrument objects using an Inflation pricer. Also, you can use inflationbuild to build an inflation curve from zero coupon inflation swap (ZCIS) rates and compute inflation index values using indexvalues.

## **Bachelier Model: Price vanilla, spread, and binary instruments**

Create a Bachelier model object for a Vanilla, Spread, or Binary instrument object and price the instrument using an AssetMonteCarlo pricer.

## **derivtool removal**

The derivtool demonstration is removed in this release. To create an HJM tree, use hjmtimespec, hjmtree, and hjmvolspec. To display the HJM tree, use treeviewer.

# R2020b

---

**Version: 3.1**

**New Features**

**Bug Fixes**

## **Pricing and Valuation: Variance swaps**

You can use `VarianceSwap` to create a `VarianceSwap` instrument object. You can price a `VarianceSwap` instrument using the `ReplicatingVarianceSwap` and `Heston` pricers.

## **Vanna-Volga Method: Price vanilla, barrier, double barrier, touch, and double touch instruments**

You can use the `VannaVolga` pricing method with a `BlackScholes` model object for a `Vanilla`, `Barrier`, `DoubleBarrier`, `Touch`, or `DoubleTouch` instrument object.

## **Touch, Double Touch, and Binary Instruments: Price instruments using Black-Scholes or Asset Monte Carlo methods**

You can create instrument objects for `Binary`, `Touch`, and `DoubleTouch` instruments. You can price these instruments using a `BlackScholes` or `AssetMonteCarlo` pricing method.

## **Double-Barrier Instrument: Price instrument using finite difference or Ikeda-Kunitomo methods**

You can create a `DoubleBarrier` instrument object and price the `DoubleBarrier` instrument using the `FiniteDifference` or `IkedaKunitomo` pricing methods.

## **Call and Put Options: Price callable and puttable float bonds, float bond options, caps, floors, and swaptions by using instruments**

You can use call or put options with `FloatBondOption`, `OptionEmbeddedFloatBond`, `Cap`, `Floor`, and `Swaption` instruments. You can price these instruments using an `IRTree` pricing method.

## **Finite Difference Method: Price vanilla instruments with Bermudan exercise**

You can create a `Bermudan` exercise for a `Vanilla` instrument object. You can price this instrument using a `FiniteDifference` pricing method.

## **Monte Carlo Simulation: Price equity, FX, and commodity instruments**

The following equity instruments support Monte Carlo simulation:

- `Vanilla`
- `Lookback`
- `Barrier`
- `Asian`
- `Spread`
- `DoubleBarrier`
- `Binary`

- 
- Touch
  - DoubleTouch

The following models support Monte Carlo simulation:

- Bates
- Merton
- Heston
- BlackScholes

The AssetMonteCarlo pricer supports Monte Carlo simulation.



# R2020a

---

**Version: 3.0**

**New Features**

**Bug Fixes**

## **Pricing and Valuation: Price various types of financial instruments individually or as a portfolio using new object-oriented framework**

The object-based framework supports a workflow for creating instruments, models, and pricer objects to price financial instruments. Using these objects, you can price interest-rate, equity, or credit derivative instruments. The object-based workflow is an alternative to pricing financial instruments using functions. Working with modular objects for instruments, models, and pricers, you can easily reuse these objects to compare instrument prices for different models and pricing engines.

The object-oriented framework supports the following capabilities:

- “Curve Objects” on page 5-2
- “Interest-Rate Instrument Objects” on page 5-3
- “Equity, Commodity, or FX Instrument Objects” on page 5-3
- “Credit Derivative Instrument Objects” on page 5-4
- “Portfolio Object” on page 5-4

For more information on the workflow for creating an instrument object, a model object, and a pricer object, see *Get Started with Workflows Using Object-Based Framework for Pricing Financial Instruments*.

For more information on the available instruments, models, and pricing methods, see *Interest-Rate Instruments with Associated Models and Pricers*.

For more information on the mapping of Financial Instruments Toolbox functions to interest-rate instrument objects, see *Mapping Financial Instruments Toolbox Functions for Interest-Rate Instruments*.

### **Curve Objects**

New objects (`ratecurve`, `parametercurve`, and `defprobcurve`) for creating curve objects. New associated functions enable you to analyze these curve objects.

For interest-rate curves using a `ratecurve` object:

- `forwardrates`
- `zerorates`
- `discountfactors`
- `irbootstrap`

For financial curves using a `parametercurve` object:

- `discountfactors`
- `zerorates`
- `forwardrates`
- `fitNelsonSiegel`
- `fitSvensson`

For default probability curves using a `defprobcurve` object:



- 
- survprobs
  - hazardrates
  - defprobstrip

### **Interest-Rate Instrument Objects**

New instrument, model, and pricer objects are available for pricing interest-rate instruments.

These instrument objects support interest-rate instruments:

- Deposit
- FixedBond
- FloatBond
- FixedBondOption
- OptionEmbeddedFixedBond
- Cap
- Floor
- Swap
- Swaption
- FRA

These model objects support interest-rate instruments:

- Black.
- HullWhite
- BlackKarasinski
- Normal.
- Sabr

These pricer objects support interest-rate instruments:

- IRTree
- Black
- HullWhite
- Normal
- Sabr
- Discount

### **Equity, Commodity, or FX Instrument Objects**

New instrument, model, and pricer objects are available for pricing equity, commodity, or foreign exchange (FX) instruments.

These instrument objects support equity, commodity, or FX instruments:

- Vanilla
- Lookback

- Barrier
- Asian
- Spread

These model objects support equity, commodity, or FX instruments:

- BlackScholes.
- Heston.
- Bates.
- Dupire.
- Merton.

These pricer objects support equity, commodity, or FX instruments:

- Levy.
- KemnaVorst.
- TurnbullWakeman.
- BlackScholes.
- ConzeViswanathan.
- GoldmanSosinGatto.
- RollGeskeWhaley.
- Kirk.
- BjerksundStensland.
- FiniteDifference.
- FFT.
- NumericalIntegration.

#### **Credit Derivative Instrument Objects**

New instrument, model, and pricer objects are available for pricing credit derivative instruments.

These instrument objects support credit derivative instruments:

- CDS
- CDSOption

This model object supports credit derivative instruments:

- CDSBlack

These pricer objects support credit derivative instruments:

- Credit
- CDSBlack

#### **Portfolio Object**

The new `finportfolio` object and associated functions enable you to manage and price a portfolio of instrument objects:

- 
- `finportfolio`
  - `pricePortfolio`
  - `addInstrument`
  - `removeInstrument`
  - `setPricer`



# R2019b

---

**Version: 2.10**

**New Features**

**Bug Fixes**

## **Interest-Rate Instruments: Compute prices and option-adjusted spread of amortizing bonds with embedded options using interest-rate tree models**

These functions now provide support for amortizing bonds with an embedded option specified with the 'BondType' name-value pair argument and a variable Face schedule.

- `optembndbybdt`
- `optembndbyhw`
- `optembndbybk`
- `optembndbyhjm`
- `optembndbycir`
- `oasbybdt`
- `oasbyhw`
- `oasbybk`
- `oasbyhjm`
- `oasbycir`

## **Interest-Rate Instruments: Calculate probability of exercise options on embedded bonds and bonds**

These functions now return the probability of exercise options on embedded bonds and additional exercise information in new `PriceTree` outputs:

- `optembndbybdt`
- `optembndbyhw`
- `optembndbybk`
- `optembndbyhjm`
- `optembndbycir`

These functions now return option exercise information on bonds in new `PriceTree` outputs:

- `optbndbybdt`
- `optbndbyhw`
- `optbndbybk`
- `optbndbyhjm`
- `optbndbycir`

## **Equity and Energy Instruments: Compute prices and sensitivities of one-touch and double one-touch options using the Black-Scholes option pricing model**

You can now compute prices and sensitivities for one-touch, no-touch, double one-touch, and double no-touch options in the foreign exchange market. Touch options (also known as binary barrier options

---

or American digitals) are path-dependent options in which the existence and payment of the options depends on the movement of the underlying asset through the option life.

To calculate the price and sensitivities of one-touch and no-touch options, use `touchbybls` and `touchsensbybls`.

To calculate the price and sensitivities of double one-touch and double no-touch options, use `dbltouchbybls` and `dbltouchsensbybls`.





# R2019a

---

**Version: 2.9**

**New Features**

**Bug Fixes**

## **Normal SABR Model: Support the computation of Normal (Bachelier) volatility for cap or floor volatility stripping**

The following functions have the new additional name-value pair argument 'Model' which enables you to specify a 'normal' (Bachelier) volatility for cap or floor volatility stripping:

- capvolstrip
- floorvolstrip

## **Finite-Difference Methods: Compute prices and sensitivities of double barrier options using the Alternating Direction Implicit (ADI) and Crank-Nicolson methods**

The following functions support finite-difference calculations for pricing double barrier options:

- dblbarrierbyfd
- dblbarriersensbyfd

## **Monte Carlo Simulation: Compute prices and sensitivities of double barrier options using the Black-Scholes option pricing model**

The following functions support the double barrier options and sensitivities using the Black-Scholes option pricing model:

- dblbarrierbybls
- dblbarriersensbybls

## **Vanilla European Options: Compute prices and sensitivities using the Bates and Merton76 models with finite differences**

The following functions support computing prices and sensitivities for vanilla European options with the Bates and Merton76 models using finite differences:

- optByBatesFD
- optSensByBatesFD
- optByMertonFD
- optBySensMertonFD

# R2018b

---

**Version: 2.8**

**New Features**

**Bug Fixes**

## **Normal SABR Model: Compute implied Normal (Bachelier) volatility and sensitivity by the SABR model**

The following functions support the computation of the implied Normal (Bachelier) volatility and sensitivities using the SABR model:

- `normalvolbysabr`
- `optsensbysabr`

## **Finite-Difference Methods: Calculate option prices by the Heston and local volatility models using the Alternating Direction Implicit (ADI) and Crank-Nicolson methods**

The following new functions support finite-difference calculations for option prices:

- `optByLocalVolFD`
- `optSensByLocalVolFD`
- `optByHestonFD`
- `optSensByHestonFD`

## **Implied Volatility: Improve performance of the `impvbybls` and `impvbyblk` functions when using the Jäckel 2016 method**

Improve performance of `impvbybls` and `impvbyblk` by using a new name-value pair argument for 'Method' with the value of 'search' or 'jackel2016'. For computing implied volatility, the default value is 'jackel2016'.

# R2018a

---

**Version: 2.7**

**New Features**

**Bug Fixes**

## **Vanilla European Options: Compute prices and sensitivities using Heston, Bates, and Merton76 models with FFT and numerical integration**

The following functions support computing prices and sensitivities for vanilla European options with the Heston, Bates, and Merton76 models using fast Fourier transform (FFT) or fractional Fourier transform (FRFT) and numerical integration:

- `optByHestonFFT`
- `optSensByHestonFFT`
- `optByHestonNI`
- `optSensByHestonNI`
- `optByBatesFFT`
- `optSensByBatesFFT`
- `optByBatesNI`
- `optSensByBatesNI`
- `optByMertonFFT`
- `optSensByMertonFFT`
- `optByMertonNI`
- `optSensByMertonNI`

## **Cox-Ingersoll-Ross Lattice Trees: Calculate prices and sensitivities of bonds, caps, floors, swaps, and options**

The following functions support Cox-Ingersoll-Ross (CIR) lattice trees:

- `cirtree`
- `cirprice`
- `cirsens`
- `bondbycir`
- `capbycir`
- `cfbycir`
- `cirtimespec`
- `cirvolspec`
- `fixedbycir`
- `floatbycir`
- `floorbycir`
- `oasbycir`
- `optbndbycir`
- `optembndbycir`
- `optfloatbycir`
- `optemfloatbycir`

- 
- rangefloatbycir
  - swapbycir
  - swaptionbycir

**Asian Options: Use Haug, Haug, Margrabe model for discrete arithmetic fixed Asian options and Turnbull-Wakeman model for continuous arithmetic fixed options**

Support for Haug, Haug, Margrabe model for discrete arithmetic fixed Asian options using the following functions:

- asianbyhbm
- asiainsensbyhbm

Support for Turnbull-Wakeman model for continuous arithmetic fixed Asian options using the following functions:

- asianbytw
- asiainsensbytw





# R2017b

---

**Version: 2.6**

**New Features**

**Bug Fixes**

## **Interest-Rate Instruments: Price swaptions with resettable legs and different basis conventions using Black, Normal, and lattice (tree-based) models**

The following functions support different swaption leg reset and basis conventions for lattice, Black, and Normal models:

- `instswaption`
- `swaptionbyblk`
- `swaptionbynormal`
- `swaptionbybdt`
- `swaptionbyhw`
- `swaptionbybk`
- `swaptionbyhjm`

## **Interest-Rate Instruments: Use Hull-White calibration routines for Shifted Black and Normal models**

Calibration routines for the following functions support the Shifted Black model and the Normal (Bachelier) model (which can handle negative interest rates):

- `hwcalbycap`
- `hwcalbyfloor`

# R2017a

---

**Version: 2.5**

**New Features**

**Bug Fixes**

**Version History**

## Interest-Rate Instruments: Price interest rate options with negative rates using normal volatility model and shifted SABR model

The following functions support the normal volatility model (Bachelier model) for interest-rate options to handle negative rates:

- `swaptionbynormal`
- `capbynormal`
- `floorbynormal`

The following functions provide an optional `Shift` argument to support the shifted Black model and the shifted SABR model for interest-rate options to handle negative rates:

- `blackvolbysabr` (Shifted SABR)
- `optsensbysabr` (Shifted SABR)
- `swaptionbyblk` (Shifted Black)
- `capbyblk` (Shifted Black)
- `floorbyblk` (Shifted Black)
- `capvolstrip` (Shifted Black)
- `floorvolstrip` (Shifted Black)

## Equity Instruments: Price American vanilla options using Barone-Adesi and Whaley model

Support for American vanilla options using the Barone-Adesi and Whaley model:

- `optstockbybaw`
- `optstocksensbybaw`
- `impvbybaw`

## bkcaplet and bkfloorlet removal

`bkcaplet` and `bkfloorlet` will be removed in a future release. Use `capbyblk` and `floorbyblk` instead.

## Version History

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
<code>bkcaplet</code> and <code>bkfloorlet</code>	Errors	<code>capbyblk</code> or <code>floorbyblk</code>	Replace all instances of <code>bkcaplet</code> and <code>bkfloorlet</code> with <code>capbyblk</code> or <code>floorbyblk</code> .

# R2016b

---

**Version: 2.4**

**New Features**

**Bug Fixes**

**Version History**

## **Equity Instruments: Price barrier options with closed form, Crank-Nicolson method, and Monte Carlo simulation**

The following functions support barrier options for equity or energy derivatives:

- `barrierbyfd` calculates barrier option prices using finite difference method.
- `barriersensbyfd` calculates barrier option prices and sensitivities using finite difference method.
- `barrierbybls` calculates prices for a European barrier option using Black-Scholes option pricing model.
- `barriersensbybls` calculates prices and sensitivities for a European barrier option using Black-Scholes option pricing model.
- `barrierbyls` calculates barrier option prices using Longstaff-Schwartz model.
- `barriersensbyls` calculates barrier option prices and sensitivities using Longstaff-Schwartz model.

## **Equity Instruments: Price European options with finite differences method**

The following functions support vanilla options for equity derivatives:

- `optstockbyfd` calculates vanilla option prices using finite differences method.
- `optstocksensbyfd` calculates vanilla option prices and sensitivities using finite differences method.

## **Hybrid Instruments: Price convertible bonds with a default risk and recovery rate using standard and implied trinomial trees**

Support for default for risk and recovery rate using `cbondbycrr`, `cbondbyeqp`, `cbondbystt`, and `cbondbyitt`. To specify a convertible bond risk and recovery rate, use the name-value pair arguments for `DefaultProbability` and `RecoveryRate`.

## **Numerix CAIL Engine: Access the Numerix Engine directly from MATLAB using an updated API**

Support for an updated Numerix® CAIL API, using the `numerixCrossAsset` object and the associated methods for: `applicationCall`, `applicationData`, `applicationMatrix`, `getData`, and `close`.

## **Functions moved from Financial Instruments Toolbox to Financial Toolbox**

The following functions are moved from Financial Instruments Toolbox to Financial Toolbox™:

- `cdsbootstrap` calculates barrier option prices using finite difference method.
- `cdsprice` calculates barrier option prices and sensitivities using finite difference method.

- `cdspread` calculates price for a European barrier options using Black-Scholes option pricing model.
- `cdrpv01` calculates price and sensitivities for a European barrier options using Black-Scholes option pricing model.
- `creditexposures` computes credit exposures from contract values.
- `exposureprofiles` compute exposure profiles from credit exposures.

## help fininstdemos removal

The `help fininstdemos` command is removed in this release. Use the `demo` command instead.

### Version History

Command Name	What Happens When You Use This Command	Use This Command Instead	Compatibility Considerations
help fininstdemos	Errors	demo 'toolbox' 'financial instruments'	Replace all instances of help fininstdemos with demo 'toolbox' 'financial instruments'.

## bkcaplet and bkfloorlet removal

`bkcaplet` and `bkfloorlet` will be removed in a future release. Use `capbyblk` and `floorbyblk` instead.

### Version History

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
bkcaplet and bkfloorlet	Warns	capbyblk or floorbyblk	Replace all instances of bkcaplet and bkfloorlet with capbyblk or floorbyblk.





# R2016a

---

**Version: 2.3**

**New Features**

**Bug Fixes**

**Version History**

## Cap and Floor Instruments: Volatility stripping

The following functions support cap and volatility stripping for interest-rate instruments:

- `capvolstrip` strips caplet volatilities from flat cap volatilities.
- `floorvolstrip` strips floorlet volatilities from flat floor volatilities.

## Swap Instruments: Pricing cross-currency, fixed-fixed, and float-float swaps

Support for pricing cross-currency swaps using `swapbyzero`. Support for fixed-fixed and float-float swaps is added to `instswap`, `swapbybdt`, `swapbyhjm`, `swapbybdt`, `swapbybk`, `swapbyzero`.

## cbprice removal

`cbprice` is removed in this release. Use `cbondbycrr`, `cbondbyeqp`, `cbondbyitt`, or `cbondbystt` instead.

## Version History

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
<code>cbprice</code>	Errors	<code>cbondbycrr</code> , <code>cbondbyeqp</code> , <code>cbondbyitt</code> , or <code>cbondbystt</code>	Replace all instances of <code>cbprice</code> with <code>cbondbycrr</code> , <code>cbondbyeqp</code> , <code>cbondbyitt</code> , or <code>cbondbystt</code> .

# R2015b

---

**Version: 2.2**

**New Features**

**Bug Fixes**

**Version History**

## Hybrid Instruments: Price convertible bonds using standard and implied trinomial trees

The following functions support building a standard trinomial tree:

- `stttree`
- `stttimespec`

`cbondbystt` supports pricing a convertible bond using an STTTree.

ITTree now supports pricing convertible bonds with `cbondbyitt`.

## Equity Instruments: Price equity derivatives using a standard trinomial tree

STTTree supports pricing for equity derivatives.

The following modified functions support the following equity instruments:

- Asian options — `asianbystt`
- Lookback options — `lookbackbystt`
- Barrier options — `barrierbystt`
- Vanilla options — `optstockbystt`
- Compound options — `compoundbystt`
- `sttprice`
- `sttsens`

## Simple Interest Convention: Calculate zero curves, forward curves, discount curves, rates, and bootstrapping using simple interest

Simple interest support for the following functions.

- `rate2disc` — Support added for `Compounding = 0` for simple interest where there is no compounding.
- `disc2rate` — Support added for `Compounding = 0` for simple interest where there is no compounding.
- `ratetimes` — Support added for `Compounding = 0` for simple interest where there is no compounding.
- `IRDataCurve` — Support added for `Compounding = 0` for simple interest for “zero” and “discount” curve types only (not supported for “forward” curves) where there is no compounding.
- `getForwardRates` — Support added for `Compounding = 0` for simple interest where there is no compounding.
- `getZeroRates` — Support added for `Compounding = 0` for simple interest where there is no compounding.
- `bootstrap` — Support added for `Compounding = 0` for simple interest for “zero” and “discount” curve types only (not supported for “forward” curves) where there is no compounding.

- 
- `intenvset` — Support added for `Compounding = 0` for simple interest where there is no compounding.

## cbprice removal

`cbprice` will be removed in a future release. Use `cbondbycrr`, `cbondbyeqp`, `cbondbyitt`, or `cbondbystt` instead.

## Version History

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
<code>cbprice</code>	Warns	<code>cbondbycrr</code> , <code>cbondbyeqp</code> , <code>cbondbyitt</code> , or <code>cbondbystt</code>	Replace all instances of <code>cbprice</code> with <code>cbondbycrr</code> , <code>cbondbyeqp</code> , <code>cbondbyitt</code> , or <code>cbondbystt</code> .



# R2015a

---

**Version: 2.1**

**New Features**

**Bug Fixes**

## Price convertible bonds using CRR and EQP lattice models

`cbondbycrr` and `cbondbyeqp` calculate the price of convertible bonds using the Tsiveriotis and Fernandes model. `instcbond` is the constructor for the CBond instrument type

The following modified functions support the new convertible bond (CBond) instrument:

- `instadd`
- `instdisp`
- `crrprice`
- `eqpprice`
- `crrsens`
- `eqpsens`

## Collateral-level computation from credit exposure simulations

`creditexposures` is enhanced to support computing exposures for counterparties under collateral agreements.

## Wrong-way risk example

The example for modeling wrong-way risk for counterparty credit risk using a Gaussian copula is available as `Example_WrongWayRisk.m` at `\finist\fininstdemos`. For more information, see [Wrong Way Risk with Copulas](#).



# R2014b

---

**Version: 2.0**

**New Features**

**Bug Fixes**

## Pricing functionality for forward options

Support is provided for pricing forward options using a modified Black approximation model with `optstockbyblk` and `optstocksensbyblk` using a new name-value pair argument for `ForwardMaturity`, which is the maturity date of the forward contract.

## Amortizing caps and floors pricing using lattice models

Support is provided for name-value pair argument, `Principal`, to pass the schedule to compute the price for amortizing caps (`capbybdt`, `capbybk`, `capbyhjm`, and `capbyhw`) and floors (`floorbybdt`, `floorbybk`, `floorbyhjm`, and `floorbyhw`). In addition, `instcap` and `instfloor` are enhanced to support the creation of cap and floor instruments with amortizing caps and floors.

## Power price simulation example

The example for simulating power price and mean reverting jump diffusion is available as `SimulateElectricityPricesExample.m` at `\fininst\fininstdemos`. For more information, see [Simulating Electricity Prices with Mean-Reversion and Jump-Diffusion](#).

## Hull-White single-factor model calibration using volatility surface

`hwcalbycap` and `hwcalbyfloor` support a new syntax.

```
[Alpha, Sigma, OptimOut] = hwcalbycap(RateSpec, MarketStrike, MarketMaturity, MarketVolatility, I
```

```
[Alpha, Sigma, OptimOut] = hwcalbyfloor(RateSpec, MarketStrike, MarketMaturity, MarketVolatility
```

The `Strike`, `Settle`, and `Maturity` input arguments are no longer required input arguments. By omitting these input arguments, you can use the `MarketStrike`, `MarketMaturity`, and `MarketVolatility` input arguments calibrate the HW model using the entire cap or floor surface.

## SABR option greeks computation

Support is provided for `Delta`, `Vega`, `ModifiedDelta`, and `ModifiedVega` sensitivities for the SABR stochastic model using `optsensbyabr`.

## Amortizing caps and floors pricing using closed form solutions (Black or Linear Gaussian two-factor models)

For the Black model, support is available for an enhanced name-value pair argument, `Principal`, to pass the schedule to compute the price for amortizing caps (`capbyblk`) and floors (`floorbyblk`). For the Linear Gaussian two-factor model, support is available for an enhanced name-value pair argument, `Notional`, to pass the schedule to compute the price for amortizing caps (`capbylg2f`) and floors (`floorbylg2f`).

## Asian options pricing example

The example comparing multiple approaches to pricing Asian options is available as `AsianOptionExample.m` at `\fininst\fininstdemos`. For more information, see [Pricing Asian Options](#).

---

## **Numerix CrossAsset Integration Layer (CAIL) 11 example**

The example for Numerix CAIL 11 support is available at `\fininst\fininstdemos\numerixfileprocessor.m`.



# R2014a

---

**Version: 1.3**

**New Features**

**Bug Fixes**

## Dual curve construction

Support for bootstrapping an interest rate curve using a different curve for discounting the cash flows with the following enhancements:

- `bootstrap` accepts a new optional input argument for `DiscountCurve`.
- `bootstrap` accepts a new bootstrapping instrument type called FRA for a forward rate agreement instrument.

For more information on using `bootstrap` for dual curve construction, see the example: Dual Curve Bootstrapping.

## Dual curve pricing of caps, floors, and swaptions using the Black model

`capbyblk`, `floorbyblk`, and `swaptionbyblk` accept an optional input argument for `ProjectionCurve`.

## Dual curve pricing of swaps and floating-rate notes

`swapbyzero` and `floatbyzero` have new examples to demonstrate pricing a swap and a floating-rate note with two curves.

## Monte Carlo and analytical pricing of lookback options

Support for lookback options using closed-form solutions or Monte Carlo simulations.

Function	Purpose
<code>lookbackbycvgsg</code>	Calculate prices of European lookback fixed- and floating-strike options using the Conze-Viswanathan and Goldman-Sosin-Gatto models.
<code>lookbacksensbycvgsg</code>	Calculate prices and sensitivities of European fixed- and floating-strike lookback options using the Conze-Viswanathan and Goldman-Sosin-Gatto models.
<code>lookbackbyls</code>	Calculate prices of lookback fixed- and floating-strike options using the Longstaff-Schwartz model.
<code>lookbacksensbyls</code>	Calculate prices and sensitivities of lookback fixed- and floating-strike options using the Longstaff-Schwartz model.

## Implied Black volatility computation for the SABR stochastic volatility model

Support for `blackvolbysabr` to calibrate the SABR model parameters and to compute SABR implied Black volatilities.

---

## **User-specified simulation paths for Longstaff-Schwartz pricing functions**

Support for `optpricebysim` to calculate the price and sensitivities of European or American call or put options based on simulation results of the underlying asset. For American options, the Longstaff-Schwartz least squares method is used to calculate the early exercise premium.

## **creditexposures function to compute credit exposures from mark-to-market OTC contract values**

Support for computing credit exposures as a part of a counterparty credit risk workflow. For more information, see `creditexposures`.

## **exposureprofiles function to derive credit exposure profiles from credit exposures**

Support for computing various credit exposure profiles, including potential future exposure and expected exposure. For more information, see `exposureprofiles`.

## **Enhanced pricing functions for instruments and portfolios with cash flows between tree levels**

The pricing algorithms for vanilla stock options have been enhanced to support `ExerciseDates` between tree levels. While `ExerciseDates` previously allowed only values that coincided with tree dates, the new pricing algorithm allows arbitrary `ExerciseDates` between the tree valuation date and tree maturity. For more information see the Bermuda option examples in `optstockbycrr`, `optstockbyeqp`, and `optstockbyitt`.

## **Swing option pricing example**

New example for Pricing Swing Options using the Longstaff-Schwartz Method.





# R2013b

---

**Version: 1.2**

**New Features**

**Version History**

## Support for Numerix CrossAsset Integration Layer (CAIL) API

Support for accessing Numerix instruments and risk models.

Class	Purpose
numerix	Create a numerix object to set up the Numerix CrossAsset Integration Layer (CAIL) environment.

  

Method	Purpose
numerix.parseResults	Converts Numerix CAIL data to MATLAB® data types.

## Kirk's approximation and Bjerksund-Stensland closed-form pricing models for spread options

Support pricing and sensitivity of spread options for the energy market using closed-form solutions.

Function	Purpose
spreadbykirk	Price European spread options using the Kirk pricing model.
spreadsensbykirk	Calculate European spread option prices and sensitivities using the Kirk pricing model.
spreadbybjs	Price European spread options using the Bjerksund-Stensland pricing model.
spreadsensbybjs	Calculate European spread option prices and sensitivities using the Bjerksund-Stensland pricing model.

## Finite difference and Monte Carlo simulation pricing for American spread options

Support pricing and sensitivity of spread options for the energy market using Monte Carlo simulation.

Function	Purpose
spreadbyfd	Price European or American spread options using the Alternate Direction Implicit (ADI) finite difference method.
spreadsensbyfd	Calculate price and sensitivities of European or American spread options using the Alternate Direction Implicit (ADI) finite difference method.
spreadbyls	Price European or American spread options using Monte Carlo simulations.
spreadsensbyls	Calculate price and sensitivities for European or American spread options using Monte Carlo simulations.

## Levy and Kemna-Vorst closed-form pricing and Monte Carlo simulation pricing for Asian options

Support pricing and sensitivity of Asian options for the energy market using Monte Carlo simulation and closed-form solutions.

Function	Purpose
asianbyls	Price European or American Asian options using the Longstaff-Schwartz model.
asiansensbyls	Calculate prices and sensitivities of European or American Asian options using the Longstaff-Schwartz model.
asianbykv	Price European geometric Asian options using the Kemna-Vorst model.
asiansensbykv	Calculate prices and sensitivities of European geometric Asian options using the Kemna-Vorst model.
asianbylevy	Price European arithmetic Asian options using the Levy model.
asiansensbylevy	Calculate prices and sensitivities of European arithmetic Asian options using the Levy model.

### **Additional CDS option pricing functionality for index swaptions**

New example for Pricing a CDS Index Option.

### **Pricing functions for vanilla options using Monte Carlo simulation**

Support pricing and sensitivity of vanilla options for the energy market using Monte Carlo simulation.

Function	Purpose
optstockbyls	Price European, Bermudan, or American vanilla options using the Longstaff-Schwartz model.
optstocksensbyls	Calculate European, Bermudan, or American vanilla option prices and sensitivities using the Longstaff-Schwartz model.

### **Hedging strategies using spread options example**

New example for Hedging Strategies Using Spread Options.

### **Pricing European and American spread options example**

New example for Pricing European and American Spread Options.

### **First-to-default (FTD) swaps example**

New example for First-to-Default Swaps.

### **New function for risky present value of a basis point**

cdsrpv01 computes risky present value of a basis point (RPV01) for a credit default swap (CDS) and conforms to the industry standards (ISDA CDS Standard Model).

## Version History

Compared with the previous version of Financial Instruments Toolbox, there are minor changes in the values computed by `cdsbootstrap`, `cdsspread`, `cdsprice`, and `cdsoptprice` when the starting dates do not fall on a payment date. The affected output arguments are as follows:

- `cdsbootstrap`: ProbData, HazData
- `cdsspread`: Spread
- `cdsprice`: Price
- `cdsoptprice`: Payer, Receiver

While the magnitudes of the value changes are very small, they might affect users who depend on exact matches to previous values. These changes are caused by the modification of the way risky present value of a basis point (RPV01) is computed and these changes were made to better reflect the industry practice of paying CDS premiums only on specific payment dates.

## optimoptions support

`optimoptions` support for `IRFitOptions`, `fitFunction` method, `hwcalbycap`, and `hwcalbyfloor`.

## Functions moved from Financial Instruments Toolbox to Financial Toolbox

The following functions are moved from Financial Instruments Toolbox to Financial Toolbox:

- `cdai`
- `cdprice`
- `cdyield`
- `tbilldisc2yield`
- `tbillprice`
- `tbillrepo`
- `tbillval01`
- `tbillyield`
- `tbillyield2disc`

# R2013a

---

**Version: 1.1**

**New Features**

## Pricing functions for options on floating-rate notes (FRNs)

Support for pricing a floating-rate note instrument with an option using tree models.

Function	Purpose
optfloatbybdt	Price an option for a floating-rate note using a Black-Derman-Toy interest-rate tree.
optfloatbyhjm	Price an option for a floating-rate note using a Heath-Jarrow-Morton interest-rate tree.
optfloatbyhw	Price an option for a floating-rate note using a Hull-White interest-rate tree.
optfloatbybk	Price an option for a floating-rate note using a Black-Karasinski interest-rate tree.
instoptfloat	Define the option instrument for a floating-rate note.

## Pricing functions for FRNs with embedded options

Support for pricing a floating-rate note instrument with an embedded option using tree models.

Function	Purpose
optemfloatbybdt	Price an embedded option for a floating-rate note using a Black-Derman-Toy interest-rate tree.
optemfloatbybk	Price an embedded option for a floating-rate note using a Black-Karasinski interest-rate tree.
optemfloatbyhjm	Price an embedded option for a floating-rate note using a Heath-Jarrow-Morton interest-rate tree.
optemfloatbyhw	Price an embedded option for a floating-rate note using a Hull-White interest-rate tree.
instoptemfloat	Define the floating-rate note with an embedded option instrument.

## Performance enhancements in implied volatility calculations

Improved performance for calculating implied volatility when using `impvbybjs` and `impvbyrgw`.

## Calibration and Monte Carlo simulation of single-factor and multifactor interest-rate models, including Hull-White, Linear Gaussian, and LIBOR Market Models

Support for pricing interest-rate instruments for caps, floors, and swaptions using Monte Carlo simulation with Hull-White, Shifted Gaussian, and LIBOR Market Models. There are three new classes, three new methods, and four new functions.

Class	Purpose
HullWhite1F	Create a Hull-White one-factor model.
LinearGaussian2F	Create a two-factor additive Gaussian interest-rate model.

<b>Class</b>	<b>Purpose</b>
LiborMarketModel	Create a LIBOR Market Model.

  

<b>Method</b>	<b>Purpose</b>
HullWhite1F.simTermStructs	Simulate term structures for a Hull-White one-factor model.
LinearGaussian2F.simTermStructs	Simulate term structures for a two-factor additive Gaussian interest-rate model.
LiborMarketModel.simTermStructs	Simulate term structures for a LIBOR Market Model.

  

<b>Function</b>	<b>Purpose</b>
capbylg2f	Price caps using a Linear Gaussian two-factor model.
floorbylg2f	Price floors using a Linear Gaussian two-factor model.
swaptionbylg2f	Price European swaptions using a Linear Gaussian two-factor model.
blackvolbyrebonato	Compute the Black volatility for a LIBOR Market Model using the Rebonato formula.





# R2012b

---

**Version: 1.0**

**New Features**

**Version History**

## Merge of Fixed-Income Toolbox and Financial Derivatives Toolbox to Financial Instruments Toolbox

Fixed-Income Toolbox™ and Financial Derivatives Toolbox™ are merged into the new product Financial Instruments Toolbox.

### Cap and floor floating-rate note pricing using trees

Support for pricing capped, collared, and floored floating-rate notes using the CapRate and FloorRate arguments.

Function	Purpose
floatbybdt	Price a capped floating-rate note using a Black-Derman-Toy interest-rate tree.
floatbyhjm	Price a capped floating-rate note using a Heath-Jarrow-Morton interest-rate tree.
floatbyhw	Price a capped floating-rate note using a Hull-White interest-rate tree.
floatbybk	Price a capped floating-rate note using a Black-Karasinski interest-rate tree.
instfloat	Create a capped floating-rate note instrument.
instadd	Add capped floating-rate note instruments to a portfolio.

### Forward-swap pricing using trees or term structure

Support for interest-rate forward swaps using the new StartDate argument to define the future date for the swap instrument.

Function	Purpose
swapbyzero	Price a bond using a set of zero curves.
swapbybdt	Price a forward swap using a Black-Derman-Toy interest-rate tree.
swapbyhjm	Price a forward swap using a Heath-Jarrow-Morton interest-rate tree.
swapbyhw	Price a forward swap using a Hull-White interest-rate tree.
swapbybk	Price a forward swap using a Black-Karasinski interest-rate tree.
instswap	Create a forward swap instrument.
instadd	Add forward swap instruments to a portfolio.

### Functions for fitting and extracting calibrated parameters from IRFunctionCurve objects

New enhancements for IRFunctionCurve object, including the ability to get calibrated parameters, the ability to specify linear inequality parameter constraints, and support for curve type in fitSmoothingSpline to be forward, zero, and discount.

---

## LIBOR market model example

New example for mortgage prepayment that uses a LIBOR market model to generate interest-rate evolutions. For more information, see Prepayment Modeling with a Two Factor Hull White Model and a LIBOR Market Model.

## Counterparty credit risk example

New example for computing the unilateral Credit Value (Valuation) Adjustment (CVA) for a bank holding a portfolio of vanilla interest-rate swaps with several counterparties. For more information, see Counterparty Credit Risk and CVA.

## Conversion of error and warning message identifiers

For R2012b, error and warning message identifiers have changed in Financial Instruments Toolbox.

## Version History

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a `try/catch` statement and performs an action based on a specific error identifier.

For example, because Fixed-Income Toolbox and Financial Derivatives Toolbox merged to become Financial Instruments Toolbox, the `finfixed` and `finderiv` message identifiers have changed to `fininst`. If your code checks for `finfixed` or `finderiv` message identifiers, you must update it to check for `fininst` instead.

To determine the identifier for an error, run the following command just after you see the error:

```
exception = MException.last;  
MSGID = exception.identifier;
```

To determine the identifier for a warning, run the following command just after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

