The Fundamental Financial Equation Class.

By Henrik Vestermark (hve@hvks.com)

User Manual for the Fundamental Financial Class

Overview

The **Financial** class is designed to perform various financial calculations such as determining present value, future value, periodic payments, number of periods, and interest rates. This class also handles compounding and payment frequencies, providing comprehensive tools for financial analysis.

Contents

Introduction to the Financial Class and the Fundamental Financial Equation

The Financial class is a powerful tool designed to perform various financial computations, including calculating present value (PV), future value (FV), periodic payments (PMT), the number of periods (NP), and interest rates (IR). These calculations are based on the Fundamental Financial Equation, which plays a crucial role in finance, underpinning mortgage calculations, investment planning, retirement savings, and more.

The Fundamental Financial Equation

The Fundamental Financial Equation is expressed as:

$$(PV+PMT(1+c\cdot IR)/IR)((1+IR)^{NP}-1)+PV+FV=0$$

This equation links five essential financial variables:

- **Present Value (PV):** The initial amount of money.
- **Future Value (FV):** The money in the future.
- **Periodic Payment (PMT):** Regular payment amount.
- **Number of Periods (NP):** Total number of payment periods.
- **Interest Rate (IR):** Interest rate per period.
- **c:** A constant that is one if interest is added at the beginning of a period and 0 if at the end.

By knowing any four of these variables, you can compute the fifth. This versatility makes the equation a fundamental tool in finance, applicable in various scenarios such as loan amortization, investment growth, and retirement planning.

Constructor

constructor(pv = null, fv = null, pmt = null, np = null, ir = null)

Parameters:

- **pv** (Present Value): Initial amount of money.
- **fv** (Future Value): Amount of money in the future.
- **pmt** (Periodic Payment): Regular payment amount.
- **np** (Number of Periods): Total number of payment periods.
- **ir** (Interest Rate): Interest rate per period.

Usage:

const finance = new Financial(1000, null, 100, 12, 5);

Getters and Setters

- **pv**: Get/Set Present Value.
- **fv**: Get/Set Future Value.
- **pmt**: Get/Set Periodic Payment.
- **np**: Get/Set Number of Periods.
- **ir**: Get/Set Interest Rate.
- **cf**: Get/Set Compounding Frequency.
- **pf**: Get/Set Payment Frequency.
- **beginning**: Get/Set Payment at the Beginning of the Period (boolean).
- **discrete**: Get/Set Discrete Compounding (boolean).

Usage:

finance.pv = 5000; let futureValue = finance.fv;

Methods

checkValidParameters(excludedParam)

Check if the four required parameters are present, excluding the one specified.

Parameters:

- **excludedParam**: The parameter to exclude from the check.

Returns:

- Object with **valid** (boolean) and **error** (string) properties.

Usage:

let check = finance.checkValidParameters('ir'); if (!check.valid) console.log(check.error);

calcEffectiveInterestRate()

Calculates the effective interest rate per period.

Returns:

- Effective interest rate (number).

Usage:

let effectiveRate = finance.calcEffectiveInterestRate();

calcPV()

Calculates and updates the Present Value.

Returns:

- Object with **success** (boolean), **value** (number), and **verbose** (string) properties.

Usage:

let result = finance.calcPV(); if (result.success) console.log(result.value);

calcFV()

Calculates and updates the Future Value.

Returns:

- Object with **success** (boolean), **value** (number), and **verbose** (string) properties.

Usage:

let result = finance.calcFV(); if (result.success) console.log(result.value);

calcPMT()

Calculates and updates the Periodic Payment.

Returns:

- Object with **success** (boolean), **value** (number), and **verbose** (string) properties.

Usage:

let result = finance.calcPMT(); if (result.success) console.log(result.value);

calcNP()

Calculates and updates the Number of Periods.

Returns:

- Object with **success** (boolean), **value** (number), and **verbose** (string) properties.

Usage:

let result = finance.calcNP(); if (result.success) console.log(result.value);

calcIR()

Calculates and updates the Interest Rate.

Returns:

- Object with **success** (boolean), **value** (number), and **verbose** (string) properties.

Usage:

let result = finance.calcIR(); if (result.success) console.log(result.value);

calcPayments()

Calculates the payment schedule and updates the internal state.

Returns:

- Object with **success** (boolean) and **value** (array of payment objects) properties.

Usage:

let result = finance.calcPayments(); if (result.success) console.log(result.value);

getDetails()

Retrieves all financial parameters in a formatted object.

Returns:

- Object with details about the financial parameters.

Usage:

let details = finance.getDetails(); console.log(details);

getPaymentPlanAsString()

Returns the payment plan as a formatted string.

Returns:

- String representation of the payment plan.

Usage:

let paymentPlan = finance.getPaymentPlanAsString(); console.log(paymentPlan);

plotPayments(canvas)

Plots the payment schedule using Plotly.

Parameters:

- **canvas**: The HTML element or canvas to plot on.

Usage:

finance.plotPayments(document.getElementById('paymentCanvas'));

plotFinancialOverview(canvas)

Plots the financial overview using Plotly.

Parameters:

- **canvas**: The HTML element or canvas to plot on.

Usage:

finance.plotFinancialOverview(document.getElementById('overviewCanvas'));

Example Usage

```
// Initialize Financial object const finance = new Financial(1000, 5000, null, 10, 5);
// Calculate Present Value
let pvResult = finance.calcPV();
if (pvResult.success) { console.log('Present Value:', pvResult.value);
console.log('Details:', pvResult.verbose); }
// Get Payment Plan as String
let paymentPlan = finance.getPaymentPlanAsString();
console.log(paymentPlan); // Plot Payments
finance.plotPayments(document.getElementById('paymentCanvas'));
 // Plot Financial Overview
finance.plotFinancialOverview(document.getElementById('overviewCanvas'));
```