

文章编号: 1005-8451 (2016) 01-0017-03

铁路资金管理系统中利息计算算法的设计与实现

吴 宁

(中国铁路信息技术中心, 北京 100844)

摘 要: 针对现有铁路资金管理筹投资系统中利息计算不准确的问题, 本文在利息手工计算方法的基础上, 优化并详细介绍了涵盖各种利息计算场景的一套综合算法, 主要包括算法原理、算法设计和实现。经过大量真实财务数据的验证测试, 新的利息结付功能模块能够得到准确的计息数据。

关键词: 筹资; 算法; 利息

中图分类号: U29 : TP39 **文献标识码:** A

Interest computing algorithm in Railway Capital Management System

WU Ning

(China Railway Information Technology Center, Beijing 100844, China)

Abstract: Aimed at the incorrect problem for the interest computing in the present Railway Capital Management System, this article put forward a comprehensive interest computing algorithm based on manual interest computing manner which could include nearly all kinds of interest computing occasions, given detail description on the principle, the design and implementation of the algorithm. After being tested and verified by large amounts of factual accountant data, the new interest computing function module could be used to get absolutely accurate interest payment result.

Key words: fund raise; algorithm; interest

随着铁路建设发展, 各类在建、运营项目的资金管理变得日益重要。具体来讲, 资金管理主要包括对外筹资/投资、对内资金调剂、资金结算等活动。其中, 筹/投资管理主要包括资金提款、资金还款及利息结付3个方面。由于算法漏洞, 当前筹投资系统中的利息结付模块始终存在计算结果不准确的问题, 为了从繁重的人工计息劳动中解放出来, 业务部门迫切需要对利息结付模块进行重构。

1 需求分析

1.1 业务需求分析

对铁路项目资金筹集业务的管理是以对筹资合同的签订和履行为管理标的, 依据筹资合同约定的日期和金额, 执行实际的本金提款和本金还款业务; 同时, 资金筹集部门需在合同期内每个约定的结息日期, 针对实际持有本金进行计息、付息。本文所述利息计息算法即是完成在特定结息日期, 每个筹

资合同应付的当期利息数额的计算。

1.2 算法要素分析

准确、便捷地获得利息计算结果, 需要从利息计算关键要素及其关联入手。

当期应结利息 = 结算区间 × 本金 × 利率

结息区间: 其取值完全取决于筹资合同指定的结息日期或方式, 通常有按季结息、按年结息等, 结息区间与当期应结利息可视为简单的正比例关系。

本金: 当期应结利息除了与结息区间首日的历史提款余额有关, 还与结息区间期间是否发生提还款及其金额有关。同时, 提还款具体发生的日期也会影响当期应结利息的计算结果。

利率: 利率的取值首先取决于央行在合同期内有效的基准利率以及筹资合同内约定的利息计算选用何种基准利率, 如3年定期、5年定期等; 其次取决于在央行基准利率发生调整时, 合同约定的利率调整方式, 如随央行调整, 还是在固定的一个或几个日期调整; 最后, 对于浮动利率计息, 还要考虑其具体浮动方式, 如按百分比浮动, 还是按点数浮动。

收稿日期: 2015-04-23

作者简介: 吴 宁, 工程师。

由此可见，如果按照影响利息计算结果的各关键要素进行排列组合的话，将会有多达数百种利息计算场景和对应算法。在实际的项目开发中，这会造成大量冗余代码和项目延期，为日后的项目维护升级也带来隐患。

2 算法设计

2.1 算法原理

利息算法通过将结息区间层层细分，直至利息计算中的各关键要素为定量，再据此给出涵盖全部利息场景的综合算法，算法设计原理如图 1 所示。

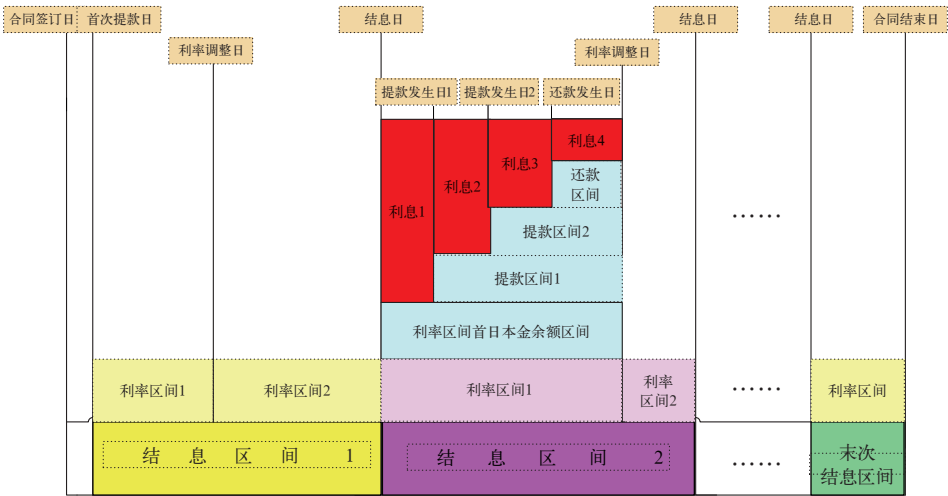


图1 利息计算算法设计原理图

结息区间由两个日期，即区间两端的日期确定，该日期由合同约定，一般具有约定俗成的规律性。如合同约定按季结息，则每年有 4 次结息，结息区间分别为每年的 12 月 21 日~3 月 20 日、3 月 21 日~6 月 20 日、6 月 21 日~9 月 20 日、9 月 21 日~12 月 20 日；在确定的结息区间内，利率未必确定，如果央行利率发生调整，还要依据合同约定的调整方式和日期，进一步将结息区间细分为利率区间，在利率区间内利率是固定不变的；但在利率确定的区间如果发生提还款业务，则计息本金金额会发生变化，因此需依据提还款日期将利率区间进一步细分为提款区间和还款区间，在这些提 / 还款区间及首日本金余额区间内，诸如本金、利率和区间起止日期，这些基本的利息计算要素都是确定已知的。

2.2 算法比较

本文提出对任一结息区间计算利息优化算法：

$$\begin{aligned} \text{结息区间 } i \text{ 利息} &= \sum_{i=1}^n \text{利率区间 } i \text{ 利息} \\ \text{利率区间 } i \text{ 利息} &= \text{区间首日本金余额区间利息} + \\ &\sum_{i=1}^n \text{提款区间 } i \text{ 利息} - \sum_{j=1}^n \text{还款区间 } j \text{ 利息} \end{aligned}$$

传统手工计算任一结息区间利息的算法：

$$\begin{aligned} \text{结息区间 } i \text{ 利息} &= \sum_{i=1}^n \text{利率区间 } i \text{ 利息} \\ \text{利率区间 } i \text{ 利息} &= \sum_{k=1}^n \text{利息块 } k \end{aligned}$$

传统算法以提款 / 还款发生的日期作为划分每个利息块 k 分界，含头不含尾。在计算每个利息块 k 时：(1) 都要通过利率区间首日本金余额加 / 减本利息块 k 首日及首日之前发生的所有提款 / 还款金额，以获得该利息块 k 的计息本金金额，这就需要频繁访问

数据库内的提还款数据，或者开辟缓冲内存保存本利息块首末日期间发生的提还款数据，用以该利息块 k 计算利息时循环调用；(2) 利息块 k 的计息起止日期也要通过频繁循环访问数据库内提款 / 还款日期来获得。即在计算每个利息块时，本金金额和起止日期都要重新计算、更新。

相对于传统算法每个利息块 k 利息的计算都要频繁计算更新利息要素参数相反，本文所提算法，每个计息区间的要素参数可通过对数据库内基本数据直接提取应用，无需二次转化计算和逻辑判断控制，就可进行利息计算。例如：所有计息区间的末日相同，直接采用利率区间的末日，即利率调整日；计息的首日则分别直接采用利率区间首日、提款日或还款日。所有计息区间的本金金额的计算，除首日本金余额区间的本金余额需要提取库内区间首日之前的提还款数据计算获得，其他计息区间的本金金额直接采用提款、还款金额即可。作为基本的利息计算要素，这些都是筹投资合同数据库内已有的基本数据。连同利率调整方式、一年天数等其他利息计算要素，都以筹资合同号为关键字保存在数据库内。直接提取库内已有数据，极大地简化了算法的逻辑难度。

2.3 算法设计与实现难点

2.3.1 算法设计

按照利息计算中计息区间的细分过程,本文算法设计了5个功能模块,逐层调用下层模块并传递已经确定的利息计算要素参数,底层模块借以计算最小提/还款区间的应结利息并对这些计息区间的应结利息累加求和,获得对应利率区间的应结利息,进一步对利率区间的应结利息累加求和,即可获得当期计息区间的应结利息,向顶层模块返回利息计算结果。利息计算算法流程如2所示。

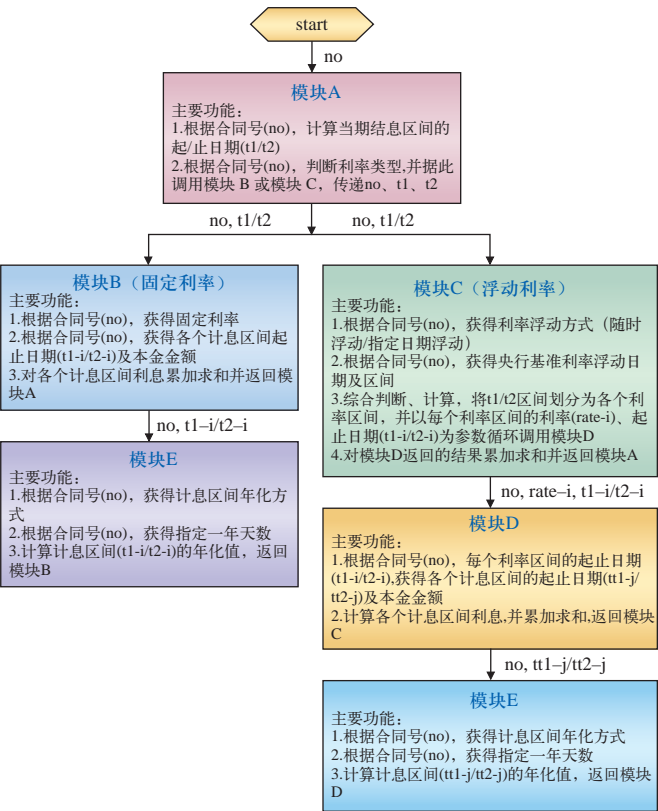


图2 利息计算算法流程图

2.3.2 算法实现难点

计息区间年化的算法：每个计息区间由首末日期确定,要根据年利率计算利息,还要将该区间折算成“年”。目前,业务上普遍有两种年化方法：积数计息法和逐笔计息法。积数计息相对简单,计息区间年化值=实际包含天数/合同约定的一年天数;逐笔计息采用业内约定俗成的满月按月的规则进行年化,例如,计息区间始末日期为3月30日和6月20日,则满月为3月30日~4月29日~5月29日,共计2个月,天数为5月30日~6月20日,共计22天。因此,年化值=2/12+22/约定一年天数。

由于在实际业务中,存在计息区间的起止日期

并非完全具有规律性、每月实际天数有差异以及计息区间跨年等多种可能,因此在区间年化的算法实现上要全面考虑各种特殊情况。另外,依据本文利息算法,每个计息区间从有效本金发生日开始,截止日相同,再根据发生金额是提款,还是还款相应进行利息累加或累减,即先进再退;而传统算法的每个计息区间则是从本金变化日开始,截止日为下一个本金变化日,而后进行利息累加,即逐步前进。本文算法在简化的同时,也会造成年化值计算结果与传统手工算法的年化值结果发生偏差。因此,在区间年化算法上也要考虑这种情况,予以更正。

2.4 算法实现

为了与既有资金管理系统更好地集成,本文算法在J2EE平台上采用Struts开发框架实现,开发环境采用Myeclipse 8.5+JDK 1.6+Java。前台采用B/S模式访问,用户只需要输入合同号,算法即可利用该借款合同在库内已有的要素数据,计算并返回当期应结利息。数据库采用Oracle 10g,在现有数据库基础上,新建相关数据表,并维护央行基准利率等初始数据,为算法的运行建立基本的数据环境。

3 结束语

项目发布前,针对业务部门提供的数百个真实筹资合同的财务数据,进行了全面的测试。合同涵盖了固定利率合同、浮动利率合同中各种典型的利息计算场景,重点测试了浮动利率合同中利率随央行实时调整和利率在每年中指定的若干个日期调整两种情况。本文算法的利息计算结果与实际财务付息数据完全一致,满足了实际应用的要求。相关细节功能还需要在投入使用的过程中进一步细化完善。

参考文献:

[1] 程杰.大话设计模式[M].北京:清华大学出版社,2009.
[2] 杨春晖.系统架构设计师[M].北京:清华大学出版社,2012.
[3] 石连运.新编企业财务管理[M].济南:山东大学出版社,1996.
[4] 铁道部资金清算中心.资金管理系统操作说明书[S].北京:铁道部资金清算中心,2006.

责任编辑 徐侃春