面向中小机构的文件管理系统的设计与实现

郭彩云 王会进 龙 舜 黎欣健
(暨南大学信息科学技术学院 广东 广州 510632)

摘 要 针对目前中小机构对文件管理的需求,分析当前网盘存在的问题,设计研发一款操作便捷、管理有效的轻量级文件管理系统。该系统采用标签技术对文件进行分类和检索,通过虚拟文件层和 SHA-1 算法实现了文件去重和资源回收,利用分布式的版本控制思想有效地解决了多用户情况下文件版本控制问题,保证了用户访问数据的一致性,实现了网络文件存储、文档协同修改、多用户共享文件等功能。实际应用情况表明,系统能够有效地管理和共享文件,实现多用户协同工作,对重复文件能够自动检测和删除,大大提高了存储资源和网络资源的利用率,降低了资源管理复杂度。

关键词 文件管理系统 标签 版本控制 虚拟文件层

中图分类号 TP317.1 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2016.09.017

DESIGN AND IMPLEMENTATION OF DOCUMENT MANAGEMENT SYSTEM FOR SMALL AND MEDIUM-SIZED INSTITUTIONS

Guo Caiyun Wang Huijin Long Shun Li Xinjian
(*College of Information Science and Technology, Jinan University, Guangzhou 510632, Guangdong, China*)

Abstract Targeted at the requirement of medium and small-sized institutions on documents management at present, we analyse current problems of network disks, and present in this paper a light-weighted document management system which is convenient in operation and effective in management. The system classifies and retrieves documents with tagging technique, and achieves files deduplication and resources recovery through virtual file layer and SHA-1 algorithm. It uses the idea of distributed version control to effectively solve the problem of document version control among multiple users, guarantees the consistency of users data access, and realises the functions of network files storage, document collaborative amendment, and multi-user document sharing, etc. Actual application situations of the system demonstrate that it can effectively manage and share files, achieves multi-user cooperative work, and can automatically detect and delete duplicate files, these greatly increase the utilisation of storage and network resources, and reduce the complexity of resource management as well.

Keywords Document management system Tag Version control Virtual file layer

0 引 言

信息技术的发展不仅推动了操作流程的自动化,也使机构和个人积累了大量的电子文档。宽带网络的普及使得文件的下载和分享变得快速而简便,但随之而来的问题是文件数量以指数级增长,文件重复率显著增加。据 IDC 报告显示[1,2],2005 年到 2020 年期间,数据总量将从 130 EB 增长到 40 000 EB。如何有效地管理这些海量数据,保护数据资产的安全和可靠,以及如何快速、便捷的实现数据转移、存储和获取等,已经成为当前阶段急需解决的现实问题[3]。

对中小型机构来说,文件管理是其信息管理的核心部分,低效的文件管理将会导致工作效率低下,存储资源浪费,大大削减其竞争力。目前,许多机构对文件管理不再局限于文件的存储与分类查询,对文档的及时更新共享和协同修改合并也提出了更多更高的要求,对于数据存储的实时性、安全性、可靠性和移动性要求也越来越高。传统的文件管理系统已经不能很好地适应用户的需求。

随着云计算云存储的发展,已经有很多网盘系统实现了文件的存储和管理[4-6]。Dropbox[7] 是 Dropbox 公司提供的在线存储服务,用户可以存储并共享文件和文件夹。Google Drive[8] 是 Google 推出的在线同步存储服务,结合了文档在线编辑功能。OneDrive[9] 是微软推出的网络硬盘及云服务,允许用户通过 Microsoft Account 来限制不同用户访问文件的权限。这些服务在一定程度上可以帮助机构和个人管理文件,实现文件存储、同步和分享等功能,但不能很好地支持多用户对同一文件的修改与合并,实现协同工作。而且出于对隐私数据的安全性考虑,很多组织机构和个人也难以接受将一些机密的文件存储在公共云存储服务中。此外,从经济角度考虑,网盘价格昂贵,对中小机构并非合适的选择。

为此,本文在综合分析中小机构需求和现有网盘技术的基础上,开发了一款轻量级的、高效实用的网络文件管理系统。该系统利用标签对文件进行多维度分类,使文件检索更加便捷高

收稿日期:2015-05-29。广州市科技计划项目(2013Y2-00066)。
郭彩云,硕士生,主研领域:网络技术与数据库。**王会进**,教授。**龙舜**,副教授。**黎欣健**,本科。

效;设计的轻量级虚拟文件层实现了文件的存储和排重,大大降低了文件的冗余程度,节省了存储空间;借鉴 Git 分布式的版本控制思想解决了在多用户文件共享中的版本冲突、版本更新等版本控制问题。

1　总体设计

系统整体框架如图 1 所示,该系统以基础网络存储为核心,包含三个主要的功能模块,分别为标签模块、用户模块、共享模块。

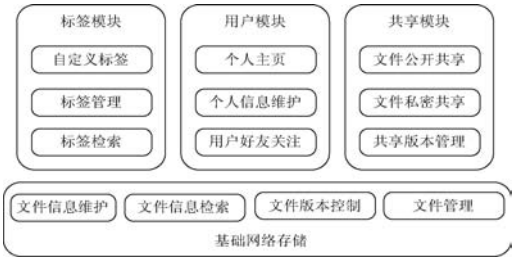

图 1　系统整体框架

(1) 基础网络存储模块用于提供基础的文件操作,提供全文检索功能,还负责维护文件的历史版本,允许查询、删除或回滚到某个历史版本。

(2) 标签模块允许用户为文件添加任意数量的自定义或系统推荐的标签。系统会根据标签使用频次和标签标注的文件数量进行加权排序得出标签显示列表。系统提供标签检索文件的功能。

(3) 用户模块提供用户个人信息的基本维护和对外公开访问的个人页面。用户可以查看公开共享的文件信息及下载次数,还提供关注的功能。关注某用户后可以收到该用户新增共享文件的通知,用户互相关注可以实现以小组形式的文件共享与版本维护,实现灵活的共享。

(4) 共享模块提供用户公开共享和私密共享两种方式。前者由系统为用户创建公开共享链接并发送至用户选择的社交网络中;后者用户可为共享添加密码。在共享版本管理中,用户可以指定文件的分享对象,并设定是否接受更新推送,接受共享的用户可以对共享文件做任意的修改,并使用系统中的更新推送功能把版本更新推送给原文件的分享者,而分享者可以查看该版本更新,并选择是否接受新版本覆盖自己的当前版本。

该系统以个人和小团队为服务对象、面向中小组织机构,致力于对大量文件的存储管理进行优化,通过对流程和技术的改进,解决了多用户对于文件分享以及版本维护等问题。与现有网盘系统相比,具有如下优势:

(1) 能有效地支持对文件的多分类管理,从而实现文件多分支检索。

(2) 支持多人共同维护文件版本,从而实现多人协同工作,提高工作效率。

(3) 不仅可以把文件公开分享到社交网络,还实现了小组内指定访问权限式的共享。

2　关键技术

2.1　标签分类思想

传统的文件分类使用单一的维度对资源进行分类,文件访问路径是唯一确定的,整个目录是一个树型结构。如果想要一个文件同属于多个类别,那么就需要拷贝该资源到相应的目录下,但这样造成了存储资源的浪费。

为了解决上述问题,受 Gmail 邮箱系统标签式邮件管理和国内外关于标签的最新研究思想的启发,将标签式分类思想引入了文件管理系统中。通过标签对资源进行分类,每个资源可以添加任意数量的标签,每个标签也能被用在任意数量的资源上,资源和标签的关联是多对多的关系。文件访问路径不再唯一,用户可以通过多条路径找到自己想要的资源。例如可以给某电影资源加上“电影”、“爱情”、“喜剧”等标签。用户就可以通过该电影的全部标签的任意非空子集作为条件检索到它。这种标签分类的思想可以满足对同一个资源多维度分类的需求,而无需额外创建资源的拷贝占用不必要的空间。

2.2　虚拟文件层的设计

传统的计算机文件系统利用文件和树形目录的抽象逻辑概念来代替硬盘和光盘等物理设备的数据块概念,用户不必关心数据实际存储地址,只需记住它所属的目录和文件名即可。

然而,这一做法在实际应用中表现出明显的不足。例如:用户可以给同一个文件起完全不同的文件名称,这会导致一个文件出现多个副本,造成存储资源的浪费和文件检索项过多而引发的查询效率低下。有研究表明[10,11],系统所保存的数据存在大量冗余,并随着时间的推移,冗余程度会变得越来越严重,系统可能要额外花费超过 10 倍的存储空间和管理成本。因此,减少冗余文件是降低系统存储和管理成本的必经途径。

为了解决上述问题,我们提出并实现了一种轻量级的虚拟文件层(如图 2 所示),它包括文件信息层、文件索引层和文件系统层。MD5 和 SHA-1 是两种常用的根据文件内容产生哈希值进行指纹识别以检测重复文件的方法[12]。对于 MD5 的 128 位哈希值,在数据量为 10^{18} B 情况下,发生单次冲突的概率为 10^{-9}。而对于 160 位的 SHA-1 哈希值,发生单次冲突的概率为 7×10^{-18}。两种方法不仅抗冲突性较好,而且在普通硬件条件下计算速度非常快,加州大学研究表明[13]采用 SHA-1 及 MD5 生成哈希索引的处理速度分别为 83 MB/s 和 227 MB/s。

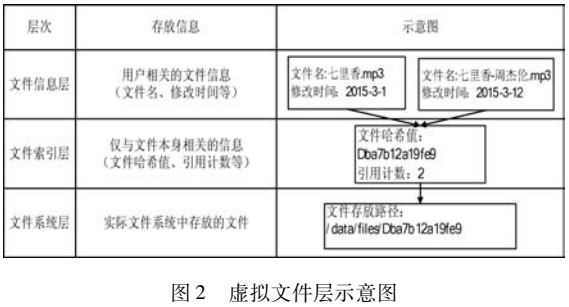

图 2　虚拟文件层示意图

本系统选用 SHA-1 生成文件的哈希值,然后将文件索引与所有已存储的哈希值进行比较,如果检测到相同的索引值,则仅将该文件信息指向重复文件的索引而不进行实际存储,否则存储新的文件。该方法计算速度快,重复数据删除吞吐量高。

在上传文件过程中,本系统首先会计算该文件的 SHA-1 值,并在索引层中检索该 SHA-1 值。如果没有找到,则新文件会上传至服务器中,引用计数加 1,并在文件信息层中添加一条属于该用户的文件条目记录,这条记录包含文件的上传时间、文件名和对应的文件哈希值等内容。如果找到,就直接将文件的引用计数加 1,并为上传用户添加新的文件条目记录,但指向原来已经存在的文件。具体如流程 1 所述:

流程 1 用户上传文件

```
Function UploadFile( HttpPostedFile uploadfile,User user1 ){
//使用 SHA-1 计算文件的哈希值
Hashvalue = SHA-1( uploadfile) ;
//在索引层中检索是否已存在该哈希值
If ( Check( Hashvalue) == false) {                    //false 表示不存在
    int count = 0;                                    //引用计数
    Add uploadfile to file server;                    //将文件上传至服务器中
    Add Hashvalue to index layer;                     //将哈希值加入到索引层中
//为文件上传者增加一条文件条目记录
    Create a record for the user1 ;
    count ++ ;                                        //引用计数加 1
}else{                                                //true 表示文件已经存在,不再上传
//为文件上传者增加一条文件条目记录
    Create a record for the user1 ;
    count ++ ;                                        //引用计数加 1
  }
}
```

当用户执行删除文件操作时,系统首先会检测用户是否有权限。如果有,则执行删除操作,如果没有,就提醒用户无权删除该文件。在删除文件过程中,由于其他用户也可能指向该文件,系统不应直接把底层对应的文件删除。因此,系统首先会计算该文件的引用计数,如果引用计数不为零表明仍有文件信息层记录引用该底层文件。执行删除操作时,就将引用计数减 1,并删除该用户指向该文件的文件条目;当引用计数为零时,就表示已经没有引用记录,即可进行资源回收。具体流程如下所述:

流程 2 用户删除文件

```
Function DeleteFile( HttpDeleteFile deletefile,User user2 ){
    //检测用户是否有删除文件的权限
    If( Check( deletefile,user2) == true) {    //true 表示用户有权限
        If( deletefile. count > 0 ) {          //文件引用计数大于 0
        Deletefile. count -- ;                 //引用计数减 1
        //删除 user2 的文件条目记录
        Delete the record of user2;
    }
    If( deletefile. count == 0 ) {             //文件无用户引用
    //存储资源回收,删除索引层中该文件的哈希值
    Delete deletefile,recycling storage resource;
    Delete deletefile. Hashvalue from index layer;
    }
}else{
    printf("无权删除该文件");        //提示用户无权限删除文件
  }
}
```

2.3 版本控制

版本控制 VC(Version Control) [14-16] 是用于追踪文件状态变化的软件管理方法,常见于软件开发、数据同步等应用环境中。目前,分布式版本控制系统被广泛运用于软件开发中,它不仅有利于源文件的更新和共享,而且客户端可以完整备份服务器中的数据,当服务器发生故障时,也可以从其他节点恢复数据。

系统借鉴 Git[17] 的分布式版本控制,实现文件追踪和多用户协同工作,具体流程如图 3 所示。

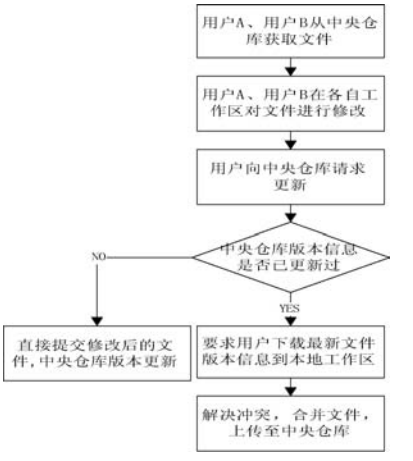

图 3 文件版本控制流程图

它的基本思想是将文件管理员的工作区作为中央仓库,允许具有下载和修改文件权限的用户获取文件,并在自己的工作区修改文件。用户可将修改后的文件提交给中央仓库。中央仓库将以栈的形式保存每个用户提交后的文件版本信息。

在提交修改文件的过程中,如果用户基于中央仓库的文件版本没有更新,用户可以直接提交修改后的文件。如果有更新,那么用户就需要将其他用户的修改合并到自己的修改文件中,并产生一个新的版本上传至中央仓库。

2.4 共享文件版本维护

共享文件在有了多个版本后会带来管理上的困难。其中一个问题是允许用户共享的是文件的当前版本还是所有历史版本。观察目前已有网盘系统,共享文件大多是创建共享时文件的当前版本。接收共享的用户一般直接下载文件,或将文件转存到自己的网盘空间再做修改,但是却无法跟原有用户或原文件做互动。有部分网盘系统也有简单的私密群组共享功能,利用简单的添加访问密码的方式来实现。但总的来说,并不能很好地处理多用户和多个文件历史版本时的文件版本维护问题。

为解决这一问题,本系统借鉴了 GitHub 在处理多用户协同开发以及代码版本控制时,使用的一种称为"fork-pull-request"的处理方法[18]。当用户 A 在 Github 上共享了一个代码仓库,用户 B 想要对其中的代码进行修改,必须通过"Fork"创建一份完整的代码仓库拷贝到自己的存储空间。当用户 B 对在自己存储空间中的那一份代码拷贝做出修改后,可以向仓库的原作者用户 A 发起"Pull - Request"。用户 A 接收到"Pull - Request"之后可以查看用户 B 做出的改动,用户 A 可以选择是否接收该改动,并且对于文本格式的代码文件,通常可以由系统自动合并修改,合并修改后文件会往后推进一个版本。

受上述做法的启发,在文件共享的过程中,系统的文件版本控制业务流程如图 4 所示。

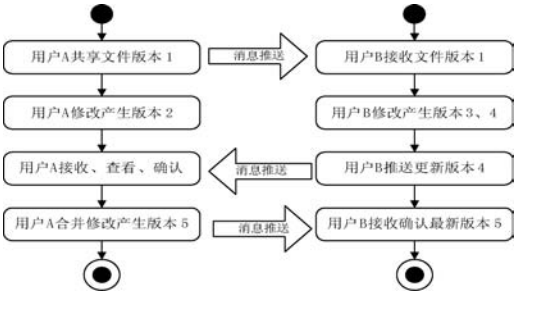

图 4 多用户文件分享中的版本控制流程图

(1) 用户 A 选择共享某个文件的版本 1 给用户 B,系统会以邮件方式发送推送消息通知用户 B 接收文件,用户 B 也可通过登录系统查看用户 A 发送的共享信息;

(2) 用户 A 在共享后对文件进行修改产生了版本 2,而用户 B 将文件版本 1 转存到自己的存储空间后进行修改,产生了版本 3 和版本 4;

(3) 如果用户 B 选择把自己的修改更新推送给原用户 A,系统将会通过推送消息通知用户 A 接收该版本的更新,用户 A 可以查看推送过来的新版本,并确认是否接收该版本;

(4) 用户 A 可以根据接收的用户 B 的修改,将修改合并到版本 2 中,产生新的文件版本 5,此时系统会以消息推送通知用户 B,如果用户 B 确认接收文件的最新版本 5,则两个用户间的文件版本就同步了。

归纳来说,我们允许其他用户创建文件的分支版本到自己的存储空间,修改之后允许用户向原文件提供者发送请求更新,这样就可以较好地处理在多用户文件共享过程中出现的版本冲突、版本更新等版本控制问题。

3　系统实现与测试

基于上述的研究和设计,我们使用 PHP 5.3.5 语言在 Windows 7 Ultimate 环境下实现了该系统,后台数据库使用 Mysql 5.1,应用服务器为 Apache 2.2.25。前台开发框架为 Bootstrap v3.1.1,JQuery 1.10.2,后台开发框架为 ThinkPHP 3.1.3。整体界面如图 5 所示,支持文件上传,标签编辑,文件搜索,用户信息维护等功能,主体部分用来显示文件列表,右击文件列表中的文件,就能弹出与该文件相关的菜单,包括编辑标签、重命名、删除文件、查看和编辑历史版本、分享文件的功能。

图 5　个人文件管理界面

为了衡量系统整体的主观满意度,邀请了 40 个用户对本系统进行了为期一个月的试用和评估,他们对本系统进行了评分,评分等级为 1 ~ 5 分,评分越高表示越满意,从图 6 中可以看出用户对本系统整体评价还是比较高的,具有一定的实用价值。

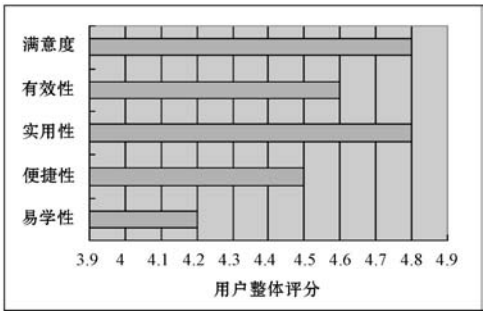

图 6　用户对系统整体评价

4　结　语

本文在基于个人和中小机构对文件存储管理的需求上,研究和分析了现有一些网盘所存在的问题,设计并实现了一款适合中小机构的基于标签的文件存储管理系统。该系统能够满足用户对大量文件管理的需求,可以自由地上传下载文件,能够利用标签对文件进行分类存储,能有效地对文件进行跟踪和版本控制,用户之间可以通过互相共享文件并以小组的方式共同维护文件的版本。

在接下来的工作中,将从以下几个方面继续完善该系统:

(1) 利用分享文件的下载热度发展个性化推荐。系统可以根据用户常用标签、常分享的文件的标签等给用户推荐文件。

(2) 社会化协同标签编辑,可以通过分析其他用户对相似文件添加的标签来做标签推荐,也允许用户邀请其他用户来协同编辑某个文件的标签。

(3) 搭建文件共享型社交网络,利用文件分享来聚集有相近兴趣点的人群,提供更加便捷的用户通信交流及文件互通的方式,加强用户之间的互动。

参　考　文　献

[1] Gantz J, Reinsel D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east[OL]. 2012. http://isilon.org/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.

[2] Gantz J, Reinsel D. The digital universe decade-are you ready? [OL]. 2010. http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf.

[3] 杨扬,贾君君,李为卫. 基于 ASP.NET 的网络文件管理系统开发应用研究[J]. 计算机科学,2011,38(B10):222-224.

[4] 王洋. 浅谈网络硬盘的应用及其优势以及缺点[J]. 智能计算机与应用,2009(4):60-61.

[5] 王妙娅. 基于网盘的网络资源共享模式特点及问题[J]. 图书馆学研究,2013(3):38-41.

[6] 傅颖勋,罗圣美,舒继武. 一种云存储环境下的安全网盘系统[J]. 软件学报,2014(8):1831-1843.

[7] Wikipedia. Dropbox_(service)[EB/OL]. [2015-5-27]. http://en.wikipedia.org/wiki/Dropbox_(service).

[8] Solano E T, Casado N S, Marí J H G. Google Drive[J]. Pilar Sanmartín Pita, 2014:69-79.

[9] Wikipedia. Microsoft_OneDrive[EB/OL]. [2015-4-28]. http://en.wikipedia.org/wiki/Microsoft_OneDrive.

[10] McKnight J, Asaro T, Babineau B. Digital archiving: end-user survey and market forecast 2006-2010[EB/OL]. [2006-03-15]. http://www.enterprisestrategygroup.com/2006/03/.

[11] 敖莉,舒继武,李明强. 重复数据删除技术[J]. 软件学报,2010,21(5):916-929.

[12] 尚颖丹. 面向文件级重复数据删除的稀疏索引技术[D]. 国防科学技术大学,2012.

[13] You L L, Pollack K T, Long D D E. Deep Store: An archival storage system architecture[C]//Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE, 2005:804-815.

[14] 张莲. 基于云存储的云同步系统的设计与实现[D]. 中国科学技术大学,2014.

[15] 阳万安,李彦. 通用版本控制系统的研究和设计[J]. 计算机工程,2008,34(12):283-284.

(下转第 87 页)

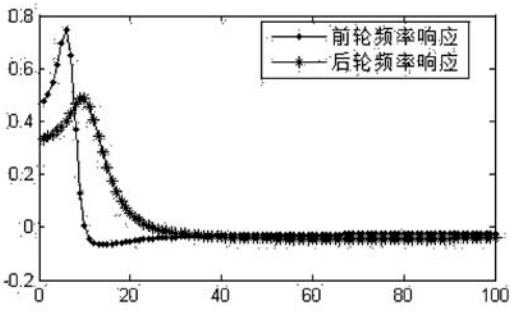

图 4　车身俯仰振动频率响应曲线

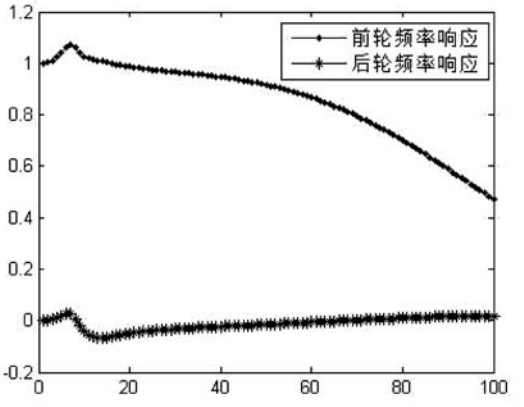

图 5　前轴振动频率响应曲线

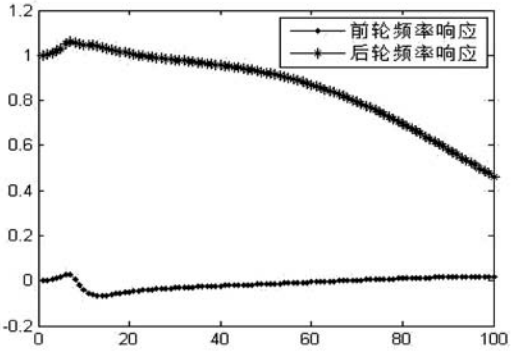

图 6　后轴振动频率响应曲线图

4　结　语

本文利用振动力学理论,建立了无人战车四自由度的振动模型,通过求解运动微分方程,得到战车系统的固有参数。同时为同类其他车载武器的动力响应求解提供了方法的借鉴。

(1) 提供求解系统的固有频率和主振型的数学解法及仿真解法。从结果可以看出,二阶固频对车体俯仰运动的影响最大,因此发射模式或者路面不平所带来的外力激励的频率要尽量避开这四阶固有频率,来避免由于共振所带来的机械损坏以及对射击精度的影响。

(2) 提供求解系统频率响应函数的数学解法及仿真解法。从图 3 和图 4 可以看出,前后轮在激励频率为 7 ~ 9 范围内极易发生共振[8]。前轮的垂直振动和俯仰振动的幅度比后轮幅度稍大;从图 5 和图 6 可以看出,由于前后悬架的型号一致,前轴振动和后轴振动引起的频率响应非常相似。其中,前轴振动对前轮的影响较大,后轴振动对后轮的影响较大。

参　考　文　献

[1] 王良曦,王红岩. 车辆动力学[M]. 北京:国防工业出版社,2008.
[2] 谢官模. 振动力学[M]. 北京:国防工业出版社,2011.
[3] William T Thomson, Marie Dillon Dahleh. Theory of Vibration with Application [M]. 5th Ed. Englewood Cliff: Prentice Hall, 1997.
[4] 全顺喜,王平,赵才有. 车辆多体系统振动方程建立探讨[J]. 振动与冲击,2013,32(11):173 - 177.
[5] 余胜威. MATLAB 车辆工程应用实战[M]. 北京:清华大学出版社,2014.
[6] Singiresu S Rao. 机械振动[M]. 4 版. 李欣业,张明路,译. 北京:清华大学出版社,2009.
[7] 李红波,薛亮,吴渝,等. 展示车辆平顺性的虚拟仿真技术[J]. 计算机工程与设计,2012,33(10):3918 - 3922.
[8] 刘云,钱振东. 路面平整度及车辆振动模型的研究综述[J]. 公路交通科技,2008,25(1):51 - 57.

(上接第 75 页)

[16] 张莲,李京,刘炜清. 云同步系统中采用增量存储的版本控制技术研究[J]. 小型微型计算机系统,2015,36(3):427 - 432.
[17] Loeliger J, Mccullough M. Version Control with Git[J]. Oreilly Media, 2012:365 - 382.
[18] Dabbish L, Stuart C, Tsay J, et al. Social coding in GitHub: transparency and collaboration in an open software repository[C]//In CSCW. 2012:1277 - 1286.

(上接第 83 页)

库进行有限元计算流程管理的思路。该思路将复杂条件下水工有限元计算流程关联起来,以期节省工作时间、提高工作效率。根据编制的管理程序的实际运用情况可以得出:进行复杂条件下水工有限元计算的管理,对于计算的效率与成功率都有显著的改善。

为实现复杂条件下水工有限元计算流程管理程序的编制,除了需要水工有限元计算必要的知识外,还需要一定的 C ++ 语言基础以及对前处理、后处理软件的熟悉。这就要求编写者具有一定的跨学科知识基础。后续的研究工作需要弱化对界面编程语言的要求,并且强化程序的可移植性,以期实现一个管理程序可以集成多种复杂条件下的水工有限元计算流程。

参　考　文　献

[1] 王勖成. 有限单元法[M]. 北京:清华大学出版社,2003.
[2] 彭国伦. 自制带标签版 Fortran 95 程序设计[M]. 北京:中国电力出版社,2002.
[3] 郭祥伟,陈国荣,刘银芳. 基于 VB 与 Fortran 混合编程的混凝土坝温度应力仿真分析软件开发[J]. 水利水电科技进展,2012,32(2):65 - 69.
[4] 黄艾舟,梅绍祖. 流程管理原理及卓越流程建模方法研究[J]. 工业工程与管理,2003(2):46 - 50.
[5] 侯俊杰. 深入浅出 Windows MFC 程序设计[M]. 武汉:华中理工大学出版社,1998.
[6] DavidJ · Kruglinski. VisualC ++ 6.0 技术内幕[M]. 北京:北京希望电子出版社,1999.
[7] 美国 ANSYS 公司. APDL 使用指南[M]. 匹兹堡:ANSYS 公司,2001.
[8] 朱百里,沈珠江. 计算土力学[M]. 上海:上海科学技术出版社,1990.