计算机科学
COMPUTER SCIENCE

基于 Coq 的逆矩阵运算的形式化

沈楠, 陈钢

引用本文

沈楠, 陈钢. 基于 Coq 的逆矩阵运算的形式化[J]. 计算机科学, 2023, 50(6A): 220400108-7.

SHEN Nan, CHEN Gang. Formalization of Inverse Matrix Operation Based on Coq[J]. Computer Science, 2023, 50(6A): 220400108-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220400108

基于 Coq 的逆矩阵运算的形式化

沈　楠　　陈　钢
南京航空航天大学计算机学院　南京 210000
(2257776063@qq.com)

摘　要　矩阵是一种在计算机科学中应用广泛的数据结构,其运算正确性具有重要意义。矩阵求逆在矩阵形式化工作当中缺乏合理且实用的形式化工作。其原因在于,工程中现有的两种常见求逆方法的形式化均存在难点。第一种是基于伴随矩阵求解方法,难点在于无法形式化地表示 $n*n$ 矩阵的子矩阵,导致构建余子式组成的矩阵十分困难,因此难以实现伴随矩阵求解逆矩阵形式化;第二种称作高斯约旦初等变换求解法,难点在于构造初等矩阵及其操作函数。若使用 Coq 归纳结构设计操作函数,即采用行优先填充二维表的思想,将舍弃列维度对二维表的描述信息,使得操作函数分支过多,需要设计复杂的归纳结构,导致后续形式化验证无法进行。文中提出了基于记录的矩阵函数构建法,使用行列两种维度同时描述矩阵,使得构造并证明初等矩阵成为可能,在此基础上实现了在 Coq 系统中基于高斯约旦消元法的矩阵求逆的形式化工作。以一种代价更小且时间复杂度更低的方式,实现了首个形式化验证下的软件逆矩阵函数库。

关键词:形式化验证;形式化工程数学;逆矩阵形式化;Coq;软件安全
中图法分类号　TP311

Formalization of Inverse Matrix Operation Based on Coq

SHEN Nan and CHEN Gang
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

Abstract　Matrix is a data structure widely used in computer science, and its correctness of operation is of great significance. In matrix formalization, there is no reasonable and practical formalization work. The reason lies in the difficulty in formalizing the two common inverse methods in engineering. The first method is based on the adjoint matrix solution method. The difficulty lies in that the submatrices of $n*n$ matrix cannot be formally expressed, which makes it very difficult to construct the matrix composed of cosubformulas. Therefore, it is difficult to achieve the formalization of adjoint matrix inverse solution. The second method is called Gauss-Jordan elementary transformation method, the difficulty lies in the construction of elementary matrix and its operation function. If Coq is used to design the operation function of inductive structure, that is, the idea of filling two-dimensional table with rows first is adopted, the description information of two-dimensional table from column dimension will be discarded, so that the operation function branches too much and complex inductive structure needs to be designed, which leads to the failure of subsequent formal verification. In this paper, a record-based matrix function construction method is proposed, which describes the matrix in both column and column dimensions, making it possible to construct and prove the elementary matrix. On this basis, the formalization of matrix inversion in Coq system based on gaussian-Jordan elimination method is realized. And we implement the first software matrix inversion function library under formal verification in a way with lower cost and time complexity.

Keywords　Formal verification, Formal engineering mathematics, Inverse matrix formalization, Coq, Software security

1　引言

矩阵是线性代数中的主要研究对象与研究工具,相较于矩阵加法、转置、乘法等矩阵计算方式,矩阵求逆是矩阵运算中最复杂的一种计算。近年来,矩阵求逆问题出现在诸多前沿领域中。在机器学习领域,用矩阵逆方法求解二维线性回归问题。在信息安全领域,字符串经过代码子表转换后生成矩阵,利用矩阵的可逆过程,实现了信息的加密解密,并在此基础上发展出了众多优秀的信息加密技术。在飞行控制、火箭制导等安全攸关领域,卡尔曼滤波算法作为一种在仿真程序中的核心算法,需借助矩阵求逆,从而实现卡尔曼增益的更新。随着矩阵算法规模的扩大,矩阵的软件算法安全性日益重要。

在工程数学领域,通常非形式化方法会导致模型以及公式描述的偏差[1]。在安全攸关领域,传统的软件测试方法通常无法完全排除程序中的错误,特别是在航空航天、核电控制、智能制造、列车控制和医疗器械等安全攸关系统中,微小错误可能导致灾难性后果。形式化方法[2]具备精确描述数学模型以及公式推导的能力,以数学推导的方式验证软件符合设计的原则从而解决工程数学的安全问题。目前,许多定理证明器都已经建立起基础理论知识库,如实数、微积分、矩阵

通信作者:陈钢(gangchensh@qq.com)

等，并验证其中的数学证明，这些研究工作为形式化工程数学的发展奠定了基础。定理证明器大体上可以分成两类，一类是基于自动证明的技术，另一类是人机交互式的证明技术[3]。前者的优势是自动化，缺点是只能证明一部分程序性质，随着问题的逐渐复杂，其弊端也逐渐显露出来。后者的缺点是需要人工辅助，优点是能够更加全面完整地证明程序性质。当前，基于 Coq 定理证明器的交互式证明技术[4]是形式化方法的代表性成果之一，该技术已经被广泛应用于数学定理证明和软硬件系统的安全性验证，以此来确保安全攸关系统的正确构造及可靠性。

在矩阵求逆形式化工作当中，目前有少量相关工作，主要集中在数学性质的验证，且不存在具备软件移植性的矩阵形式化库。原因在于，现有的两种在工程中常见的矩阵求逆方法均存在形式化困难。在基于伴随矩阵求解方法中，很难形式化地表示 $n*n$ 矩阵的子矩阵，导致矩阵余子式组成的矩阵的形式化非常复杂和困难[5]，无法轻易构建伴随矩阵求解逆矩阵。在高斯约旦消元法中，主流的矩阵形式化表示方法基于嵌套列表，对向量和矩阵的操作本质上是对其列表或嵌套列表的操作，即采用行优先填充二维表的思想，舍弃了列维度对二维表的描述信息。

嵌套列表在描述特殊矩阵二维表时，每出现一行需要 k 个非零值的情况，都需要设计相应的一维列表操作函数，并验证相关性质。这种方式不具备多态特性，复用性差，且函数构造子过多，难以形式化地表示初等变换矩阵以及矩阵求逆操作，导致后续性质证明无法进行。

为了解决上述问题，本文总结分析了已有矩阵形式化工作的利弊，提出了基于记录的矩阵函数构建法。克服了初等矩阵以及逆矩阵操作函数的形式化问题，从而系统地给出了初等矩阵的定义以及逆矩阵的形式化证明。同时继承了 Coq 中基于记录的矩阵形式化方法的良好软件移植性[6]，实现了首个形式化 Ocaml 矩阵函数库。

本文第 2 节总结分析了定理证明器中矩阵形式化的相关工作，特别是嵌套列表形式化难点的详细讨论；第 3 节提出了基于 Record 矩阵的函数描述法，实现初等变换矩阵的形式化；第 4 节提出了基于高斯约旦消元法的逆矩阵形式化方法；第 5 节介绍了求逆过程函数的正确性，逆矩阵定理的证明过程；最后总结全文。

2 相关工作

在高阶定理证明器中，各个定理证明器社区已经存在关于矩阵理论形式化的研究工作。目前，针对定长数据结构的形式化技术大体上有 3 种。

(1)函数表示法描述矩阵。首都师范大学施智平团队使用该方法在 HOL4 当中开展了比较完善的矩阵形式化工作[7-9]。该方法用函数来代表矩阵，具有良好的通用性，关于矩阵数学性质的证明简单易懂。Coq 中的 mathcomp 库沿用了使用函数表示矩阵的思路，把矩阵定义成从两个有限集到 A 的函数[10-11]，即表示为 $Fin->Fin->Type$ 类型。也就是说，它的输入集合变成了高度和宽度下标取值的集合。

这种方式以函数而不是实际数据结构的方式描述矩阵，同矩阵在程序语言中的实现方式距离较远，难以从这样的表示方式中直接抽取出程序语言可处理的矩阵表示以及矩阵代码。这样一种表示方式适合把矩阵看成是一个数学概念，并完成矩阵的数学性质证明，而不是把矩阵看成是一种软件中的数据结构，所完成的形式化工作是数学概念、算法和性质的形式化，而不是矩阵软件的形式化。

(2)采用依赖类型进行描述，Nicolas Magaud 提出了基于依赖类型的 list 的矩阵形式化方法，但只给出了关于向量的一些简单性质的证明，并没有给出其他基础性质的证明。Coq. Vectors 标准库提供了定长数组的实现，可在此基础上实现数组的数组，即矩阵。这种技术同样能够定义出完整的矩阵类型并用于矩阵函数的类型检查。此外，它比基于函数的矩阵定义更接近于程序语言中的矩阵数据结构。然而，由于定义的复杂性，这一技术所产生的形式化描述可读性较差而且证明难度很大，当处理更为复杂的二维矩阵问题时将面临较大的困难[12]。另外，从基于依赖类型的矩阵函数抽取出的列表需要在每一层包含该层的表长，因此在表达上冗余度大，占用空间多，计算时间过长。

Coquelicot 库采用基于依赖类型的对偶类型构建矩阵类型，但矩阵的相关性质较少[13]。

(3)基于记录的矩阵表示法[6]，Coq 提供了 Record 命令来将多个对象聚合为单个对象的数据结构。

Record 中的每一个域，都作为描述该数据结构的一种属性，具有良好的可读性以及便利的代码抽取特性。Ma 等[6]借助 Record 类型，构建了 Record 矩阵类型的定义。Ma 等在此基础上，首次实现了基于 Coq 的分块矩阵运算的形式化[14]。该方法易于理解并且方便使用，与其他矩阵形式化方法相比，记录类型更容易抽取出合理的 Ocaml 代码，因此形式验证的性质等同于抽取的软件代码的性质。

```
Record Mat(A:Set)(m n:nat):Set:=mkMat
{
  mat:List(List A);
  mat_height:height mat=m;
  mat_width:width mat n
}
```

基于记录的矩阵表示法中，Mat 类型分别使用 3 个域来描述二维矩阵，第一是存储元素的二维表，由嵌套 List 构成，其他两个分别是这个二维表高和宽的证明，即指定列表的长度以及嵌套列表的长度[6]。在这个定义中，A 表示矩阵内部元素的类型，是一种多态性质的变量，m 和 n 分别描述矩阵的行数以及列数，mat_height 用于保存一个关于二维表 mat 的高度为 m 的证明，mat_width 则用于保存一个关于二维表 mat 宽度为 n 的证明。

List 是元素类型相同、长度可变的序列，由于列表的元素也可以是列表，因此用嵌套列表的方式可以实现二维甚至多维嵌套列表。Coq 中采用 Inductive 关键字表示归纳类型。在归纳类型定义中，每个构造函数可以接受任意数量的参数。

```
Inductive List A:Type:=
  |nil:List A|cons:A->List A->List A
```

但嵌套列表在描述复杂矩阵二维表时会导致表示困难，

最核心的问题在于，该方案采用了行优先填充二维表的思想，舍弃了列维度对二维表的描述信息，造成函数分支数量骤增。即每出现一行需要 k 个非零的情况，需要设计相应的一维列表操作函数，并验证相关性质，不具备复用性。同时，嵌套列表在描述特殊矩阵二维表时会导致证明困难。Record 矩阵性质证明采用 induction 机制，对每种出现的分支分别进行展开证明，证明的分支数量随着展开呈现指数型的上升。因此构造矩阵时复杂的归纳结构随着列表维度以及性质复杂度的上升而上升，证明代码量也将以指数级的速度增长，导致后续矩阵性质证明困难。

据我们所知，高阶定理证明器的逆矩阵形式化仅有少量研究工作：首都师范大学施智平团队实现了在 HOL 中基于伴随矩阵方法的逆矩阵形式化[5]，该工作构造了特殊的 $n*n$ 矩阵来代替 $(n-1)*(n-1)$ 子矩阵，以便计算其余子矩阵，从而使形式上构造一个伴随矩阵成为可能。但这种矩阵算法计算复杂度高，达到 $O(N^4)$，不适合作为大矩阵的求逆计算，特别是在飞行控制等具有海量数据的领域。在 ssreflect 库[10-11]中也有矩阵求逆的定义，其方法同样是基于伴随矩阵方法的逆矩阵形式化。这种实现被 3 层包装(Matrix/Finfun/Tuple)隐藏，定义了较为复杂的结构，且矩阵类型被视为是抽象的数学概念，无法进行实际应用。

本文期望找到一种矩阵表示方案，一方面要求同矩阵的软件实现有更紧密的关联，另一方面要方便地验证矩阵性质。因此本文提出 Record 矩阵函数描述法，结合了两种典型方案的特性，构建实用矩阵形式化的基础库。我们在 Ma 等工作的基础上增加了更多的矩阵性质证明，特别是初等矩阵定理的证明；在此基础上实现逆矩阵形式化，证明相关性质。

3　Record 矩阵函数描述法

本文提出了基于 Record 矩阵的函数描述法，*Mfill* 函数使用描述函数 f 填充二维列表中的每个元素。描述函数 f 类型为 $nat->nat->A$，两个参数表示下标为 i 和 j，填充值域在 A 下的值。

```
Definition Mfill(m n:nat)(f:nat-> nat-> A):=
let dl:=generate_matrix f m n m in
mkMat A m n dl(dlist_height f m n)(dlist_width f m n).
```

mkMat 负责接收二维列表 *dl*，以及二维列表高度宽度证明 dlist_height，dlist_width。核心为二维列表构造函数 generate_matrix。该函数负责根据 f 的具体定义产生 $m*n$ 大小的二维列表 *dl*，用作 mkMat 中的二维列表参数。

Coq 采用关键字 Fixpoint 表示递归函数。函数 generate_matrix 利用参数 i，控制当前递归的矩阵行数，依次组合相应的一维列表，得到最终的二维列表。

```
Fixpoint generate_matrix(f:nat-> nat-> A)(m n i:
nat):=
  match i with
  | 0=> nil
  | S i'=>
    generate_row f n n(m-i)::generate_matrix f m n i'
  end
```

赋值工作最终由 generate_row 完成，Coq 使用::符号表示连接 List 中的头元素以及后面的 List 内容，元素值通过描述函数 f 得到。

```
Fixpoint generate_row(f:nat-> nat-> A)(j n i:nat):=
  match j with
  | 0=> nil
  | S j'=>
    fi(n-j)::generate_row f j' n i
  end
```

同时通过引理 dlist_height，dlist_width 证明所有由该函数创建的二维表高度、宽度均符合要求，免去了 Record 原有反复证明矩阵高度宽度的步骤，进一步压缩代码量。以证明产生的二维列表 *dl* 高度为例，即 *length*(*generate_matrix f m n m*)=*m*。

```
Lemma dlist_o_height_help:forall (f:nat -> nat -> A) (m n i:nat),
  i<=m ->
  length (generate_matrix f m n i) = i.
Proof.
  intros.
  induction i.
  -auto.
  -simpl. f_equal. apply IHi. auto. omega.
Qed.

Lemma dlist_o_height:forall (f:nat -> nat -> A) (m n:nat),
    length (generate_matrix f m n m) = m.
Proof.
  intros.
  apply dlist_o_height_help with (i:=m).
  auto.
Qed.
```

图 1　二维列表高度证明

Fig. 1　Two dimensional list height proof

值得一提的是，Coq 中使用 Lemma 关键字表示定理的证明，在直接证明 length(generate_matrix f m n m)=m 时存在困难，因此需要利用 Coq 中优秀的归纳特性，额外定义定理。利用

```
length(generate_matrix f m n i)=i->
length(generate_matrix f m n(i+1))=i+1
```

进行推导，完成对于任何 $i<=m$，都使得 *length*(*generate_matrix f m n i*)=*i* 成立的证明。这在 Coq 证明过程中是一种有效且实用的解决方案。

通过建立描述函数($f:nat->nat->A$)和 *generate_matrix* 函数产生的二维列表之间的联系，基于 Record 矩阵的函数描述法合理利用了行参数 i、列参数 j 对二维列表的描述信息，解决行递归带来的复杂矩阵表示问题；同时以多态的思想解决了复用性差的问题，使得构造复杂矩阵成为可能。

在线性代数中，任何将单位矩阵做一次初等变换所得到的矩阵，被称为初等矩阵。本文利用 Record 矩阵的函数描述法定义了 3 种初等矩阵函数。

(1)初等倍乘矩阵：将单位矩阵第 i 行(或列)乘 c，得到矩阵 $E_i(c)$。

```
Definition multi_mult(x:nat)(c:A):
    @Mfill nn(fun i j=>
```

(if beq_nat i j
then(if beq_nat i x then c else M. One)
else M. Zero)).

初等倍乘矩阵函数,参数为描述函数(*fun i j* =>(if *beq_nat i j* then(if *beq_nat i x* then *c* else *M. One*) else *M. Zero*)),以及行数 x 和值 c。其中 *beq_nat* 表示自然数域内的相等函数,*M. One* 和 *M. Zero* 为值域 A 下的 1 和 0,*beq_nat i j* 的作用为判断当前元素是否位于矩阵对角线,若相等则位于对角线,反之则不是。*beq_nat i x* 的作用为判断当前元素是否位于矩阵的第 x 行,若上述判断均为真,那么该元素赋值为 c。若 n 取 3,x 取 1,则能得到形如 $\begin{bmatrix}1&0&0\\0&c&0\\0&0&1\end{bmatrix}$ 的初等倍乘矩阵。

(2)初等倍加矩阵:将单位矩阵第 i 行乘 c 加到第 j 行,或将第 j 列乘 c 加到第 i 列,得到矩阵 $\boldsymbol{E}_{ij}(c)$。

Definition multi_add(i j:nat)(c:A):=
MI M+(single_val x y c)

初等倍加矩阵函数,通过单位矩阵 **MI** 和单值矩阵(*single_val x y c* 函数表示第 x 行第 y 列值为 c,其余元素均为 0)相加方式得到,$M+$ 表示矩阵相加。如图 2 所示,表示 3 * 3 矩阵下 *multi_add* 2 1 *c* 的结果。

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}+\begin{bmatrix}0&0&0\\0&0&0\\0&c&0\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&0\\0&c&1\end{bmatrix}$$

图 2 初等对换矩阵实现

Fig. 2 Elementary commutation matrix implementation

(3)初等对换矩阵:将单位矩阵第 i,j 行(或列)对换,得到矩阵 $\boldsymbol{E}_{ij}$;

Definition swap(i j:nat):
MI
M−(single_val x x M. One)
M−(single_val y y M. One)
M+(single_val x y M. One)
M+(single_val y x M. One).

初等对换矩阵函数,实现思路为单位矩阵减去要交换行的对角线元素,再对相应位置赋值 1,$M-$ 表示矩阵相减。如图 3 所示,表示 3 * 3 矩阵下得到交换第 1,2 行的对换矩阵的思路。

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}+\begin{bmatrix}-1&0&0\\0&0&0\\0&0&0\end{bmatrix}+\begin{bmatrix}0&0&0\\0&-1&0\\0&0&0\end{bmatrix}+\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix}+\begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix}=\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}$$

图 3 初等对换矩阵实现

Fig. 3 Elementary commutation matrix implementation

为方便求逆,下文中定义的 ***multi_mult_n***,***multi_add_n***,***swap_n***,分别表示维度是 $n*n$ 的 3 种初等变换矩阵。

总体来说,这种方法解决了矩阵表示困难的问题,给其他复杂矩阵的构造提供了可能性,具备良好的扩展性。以描述函数 f 表示矩阵的方式替代 Fixpoint 归纳结构,解决了在证明时遇到复杂矩阵难以展开归纳结构的问题[14]。

4 逆矩阵形式化

4.1 高斯约旦消元法

逆矩阵存在多种求解方法,最常见为基于伴随矩阵的求逆方法,采用构建行列式,伴随矩阵的定义。该方法易于在命令式语言中表示,是实际应用中最常见的方法,但在高阶定理证明器当中,难点在于无法形式化地表示 $n*n$ 矩阵的子矩阵,导致构建余子式组成的矩阵十分困难[5];需要额外构建行列式定义,证明行列式相关性质,导致引理数量变得庞大,增加了证明的成本。

另一种常见方法叫做初等变换法求逆,又名高斯约旦消元法求逆。矩阵 $\boldsymbol{A}$ 经过有限次初等行变换变为单位矩阵,则单位矩阵经过同样的初等行变换变为 $\boldsymbol{A}^{-1}$。相比上述方法的时间复杂度为 $O(N^4)$,该方法的计算复杂度为 $O(N^3)$,降低了求逆计算量。

$$\boldsymbol{P}_k\cdots\boldsymbol{P}_3\boldsymbol{P}_2\boldsymbol{P}_1(\boldsymbol{A}|\boldsymbol{I})=(\boldsymbol{I}|\boldsymbol{A}^{-1}) \tag{1}$$

在高阶定理证明器当中,初等变换矩阵 $\boldsymbol{P}_i$ 具有后进先出的栈性质,逻辑清晰。但由于初等变换存在多种求解可能性,需要确定一个合理且固定的变换顺序,作为求解的方法。本文采用先行阶梯化后求解的思路。

4.2 行阶梯矩阵形式化

对于任一个 $m*n$ 型矩阵 $\boldsymbol{A}=(a_{ij})_{m*n}$,如果它的第一列元素不全为零,则可以对它作若干次初等行变换,将其变换为行简化阶梯型矩阵 $\boldsymbol{U}$,初等行变换所对应的初等矩阵依次记为 $\boldsymbol{P}_i(1\leqslant i\leqslant k)$。

$$\boldsymbol{P}_k\cdots\boldsymbol{P}_3\boldsymbol{P}_2\boldsymbol{P}_1\boldsymbol{A}=\boldsymbol{U} \tag{2}$$

行阶梯函数 *row_echelon_form* 即矩阵行简化阶梯型函数,输入参数分别为 $n*n$ 大小的方阵 **MA** 和控制递归深度的参数 i。该函数分为 3 个步骤:1)找出当前列的主元(第一个值非 0 的元素)所在行,交换到第 cur 行;2)得到当前行对角线元素(**MA**&[cur,cur]),当前行整体倍乘它的倒数(M. inv 取倒数);3)随后通过倍加矩阵,每行依次消元。函数最终返回一个含有两个 $n*n$ 大小的方阵的对偶,第一个方阵存储变换矩阵($\boldsymbol{P}_k\cdots\boldsymbol{P}_3\boldsymbol{P}_2\boldsymbol{P}_1$),第二个方阵存储结果,即行简化阶梯型矩 $\boldsymbol{U}$。

其中,M^* 表示矩阵相乘,$\boldsymbol{A}$&[i,j]表示取矩阵 $\boldsymbol{A}$ 中第 i 行第 j 列的元素值。

行阶梯函数 row_echelon_form 依赖于多个递归函数,其中 *first_none_zero* 作用为得到当前列的第一个值非 0 的元素所在的行数,返回值为 r。

```
Fixpoint row_echelon_form(MA:Mat A n n)(i:nat):=
  match i with
  | 0=>(MI,MA)
  | S i'=>
  let cur:=(n−i) in
  (* step 1 *)
  let r:=first_none_zero MA i cur in
  let P1:=(swap_n cur r) in
  let A0:=P1 M* MA in
  (* step 2 *)
  let P2:=(multi_mult_n cur(M. inv A0&[cur,cur]))
  in
  let A1:=P2 M* A0 in
```

```
(* step 3 *)
let(P3, A2) := minus_down A1 cur i' in
let(P4, A3) := row_echelon_form A2 i' in
(P4 M* P3 M* P2 M* P1, A3)
end.
```

利用 *swap_n* 初等对换矩阵，将当前列的主元（第一个值非 0 的元素）所在行元素，与 cur 行元素互换，如图 4 所示。

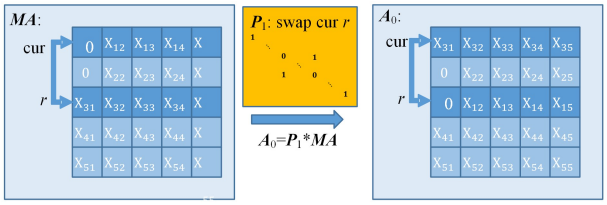


图 4　*swap_n* 示意图

Fig. 4　*swap_n* diagram

为方便后续化简，需接着使用 *multi_mult_n* 初等倍乘矩阵，将当前行首个元素乘以逆元 *M. inv* $\mathbf{A}_0$ &[*cur*, *cur*]，进行化简，如图 5 所示。

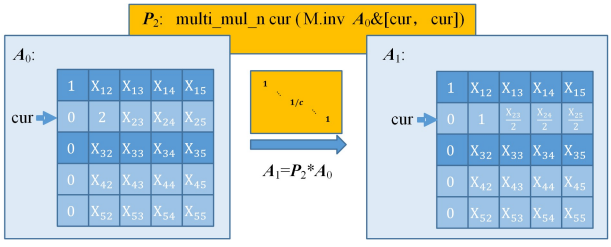


图 5　*multi_mult_n* 示意图

Fig. 5　*multi_mult_n* diagram

minus_down 为列化简函数，依赖于 multi_add 函数，其作用如图 6 所示，将当前列对角线以下元素进行简化处理，每行基于初等倍加矩阵进行消元。

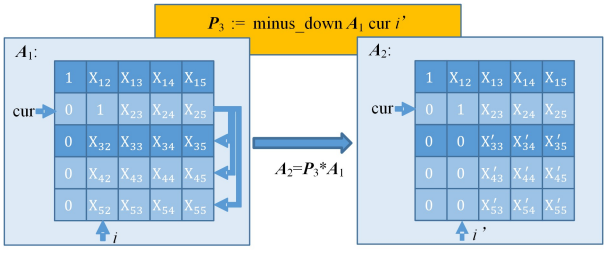


图 6　*minus_down* 示意图

Fig. 6　*minus_down* diagram

在整个过程中，函数 *row_echelon_form* 存储相应初等变换矩阵 $\boldsymbol{P}_i$（$1 \leqslant i \leqslant k$），作为求逆的条件，如图 7 所示。

$$\boldsymbol{A}^{-1} = \boldsymbol{U}^{-1}\boldsymbol{P}_k \cdots P_3 P_2 P_1 \tag{3}$$

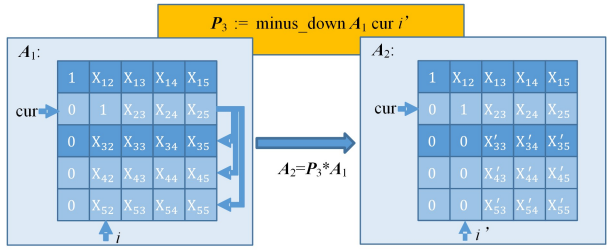


图 7　***row_echelon_form*** 示意图

Fig. 7　Diagram of ***row_echelon_form***

4.3　求逆函数形式化

对于任何一个可逆行阶梯矩阵 $\boldsymbol{U}$，都可以作若干次初等行变换将其化为单位矩阵 $\mathbf{I}$，即存在初等矩阵 $\boldsymbol{P}_i'$（$1 \leqslant i \leqslant k$）使得：

$$\boldsymbol{P}_k' \cdots \boldsymbol{P}_3' \boldsymbol{P}_2' \boldsymbol{P}_1' \boldsymbol{U} = \mathbf{I} \tag{4}$$

即可视作：

$$\boldsymbol{U}^{-1} = \boldsymbol{P}_k' \cdots \boldsymbol{P}_3' \boldsymbol{P}_2' \boldsymbol{P}_1' \tag{5}$$

同理，*fst_to_I* 函数描述上述公式，输入参数为 $n * n$ 大小的行阶梯方阵 ***MA*** 和控制递归深度的参数 i。其中，*minus_up* 作用类似于 *minus_down*，将当前列对角线以上元素进行消元化简。函数返回两个 $n * n$ 的矩阵对偶，第一个矩阵表示变换矩阵的乘积 $\boldsymbol{P}_k, \boldsymbol{P}_{k-1}, \cdots, \boldsymbol{P}_1$，第二个矩阵存储结果，即单位矩阵 $\mathbf{I}$。

```
Fixpoint fst_to_I(MA:Mat A n n)(i:nat):=
  match i with
   O=>(MI,MA)
   S i'=>
   (P1, A0):=minus_up(MA) i' i'
   let(P2, E):=fst_to_I A0 i' in
   (P2 M* P1, E)
  end
```

根据式(3)和式(5)，矩阵求逆由上述两个步骤合并得到。*row_echelon_form* 对偶的第一个元素，即 $\boldsymbol{P} = \boldsymbol{P}_k \cdots \boldsymbol{P}_3 \boldsymbol{P}_2 \boldsymbol{P}_1$，*fst_to_I* 对偶的第一个元素 $\boldsymbol{P}'' = \boldsymbol{P}_k' \cdots \boldsymbol{P}_3' \boldsymbol{P}_2' \boldsymbol{P}_1'$，相乘得到逆矩阵 $\boldsymbol{A}^{-1} = \boldsymbol{P}''\boldsymbol{P}$，如图 8 所示，完成矩阵求逆的形式化。

```
Definition Inversion(MA:Mat A n n):
let(P,A1):=row_echelon_form MA n in
let(P',A2):=fst_to_I A1 n in
  P' * P
end
```

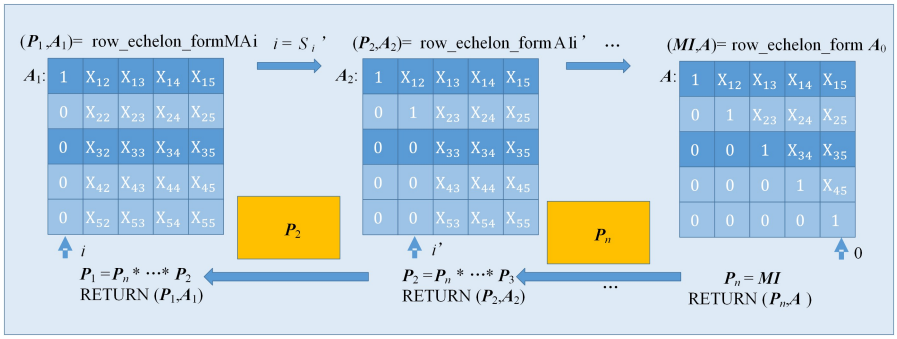


图 8　Inversion 示意图

Fig. 8　Diagram of Inversion

特别地，本次研究采取 Module Type 的方式定义矩阵元素的性质，以抽象值域 A 代替实际数字类型，使得所有矩阵函数具有多态性质，提取出了形式化整数矩阵、实数矩阵等基础库，可直接应用于 Ocaml 代码抽取。

```
Module Type MType
...
End MType
Module Matrix(M:MType)
```

5 逆矩阵证明

5.1 矩阵类型等价

在文献[6]中，矩阵类型的等价，采用 Coq 标准库中“*eq*”函数的方式定义矩阵间的等价关系，即莱布尼兹等式。

Axiom M_eq:forall(A:Set)(m n:nat)(m_1 m_2:Mat A m n),mat A m n m_1=mat A m n m_2<-> m_1=m_2

由于 Record 类型的等价，要求其中的每一个属性相等，即要求二维表的高度以及宽度证明引理相等。且该公理(Axiom)存在漏洞，前后条件并非充分必要关系，二维表相等无法得到 Record 内二维表的高度以及宽度证明引理相等。我们知道，对于两个矩阵，只需要满足二维列表相等即可认为两个矩阵相等。而对于各自行、列数的证明，不必要求相等。为了解决上述公理产生的问题，我们提出了新的矩阵等价判断函数 *Meq*，定义如下：

```
Definition Meq(m1 m2:Mat A m n):=
  for all i j,i < m-> j < n->
m1 &[i,j]=m2 &[i,j]
```

该定义用来解决 Record 类型的相等要求，且能够利用下标访问的便利性，完成文献[6]中无法证明的性质(见第 5.2 节)。

5.2 矩阵与逆矩阵形式化性质证明

本文证明的矩阵函数性质，可看作是对文献[6]已有性质的补充。补充证明性质包括但不限于：1)单位矩阵转置不变性质；2)左乘相同矩阵性质；3)右乘相同矩阵性质。证明逆矩阵性质有：1)五大逆矩阵性质；2)矩阵乘法消元法求解逆矩阵；3)转置仍可逆性质；4)矩阵两次求逆相等性质等。

表 1 矩阵性质引理

Table 1 Lemma of properties of matrix

Lemma_name	lemma
trans	
Inv_trans_trans	$((\boldsymbol{A}^{-1})^{\mathrm{T}})^{\mathrm{T}}=\boldsymbol{A}^{-1}$
I_trans_I	$\mathbf{I}^{\mathrm{T}}=\mathbf{I}$
mult	
mult_equal_r	$\boldsymbol{A}=\boldsymbol{B}->\boldsymbol{A}*\boldsymbol{C}=\boldsymbol{B}*\boldsymbol{C}$
mult_equal_l	$\boldsymbol{A}=\boldsymbol{B}->\boldsymbol{C}*\boldsymbol{A}=\boldsymbol{C}*\boldsymbol{B}$
matrix_cancellation	$\boldsymbol{C}^{-1}->\boldsymbol{A}*\boldsymbol{C}=\boldsymbol{B}*\boldsymbol{C}->\boldsymbol{A}=\boldsymbol{B}$
inv	
row_mul_c_inv	$\boldsymbol{A}*\boldsymbol{B}=\mathbf{I}->$ $\boldsymbol{A}*\boldsymbol{C}=\mathbf{I}->\boldsymbol{C}=\boldsymbol{B}$
AB_inv	$\boldsymbol{B}^{-1}*\boldsymbol{A}^{-1}*\boldsymbol{A}*\boldsymbol{B}=\mathbf{I}$
trans_mult_inv_correct	$\boldsymbol{A}^{\mathrm{T}}*(\boldsymbol{A}^{-1})^{\mathrm{T}}=\mathbf{I}$
inv_inv_correct	$(\boldsymbol{A}^{-1})^{-1}=\boldsymbol{A}$
row_mul_c_inv	$(k\boldsymbol{A})^{-1}=k^{-1}*\boldsymbol{A}^{-1}$

以单位矩阵转置不变性质 I_trans_I 为例，借助新等价定义，以类似 destruct(j?=i)的方式分解矩阵，避开了行优先填充二维表的思想，利用列维度对二维表的描述信息，证明引理正确性。如图 9 所示，证明转置仍可逆性质$\boldsymbol{A}^{\mathrm{T}}*(\boldsymbol{A}^{-1})^{\mathrm{T}}=\mathbf{I}$。

```
Lemma I_trans_I:
    Mtrans I M= I.
Proof.
    unfold I.
    intros.
    unfold Meq.
    intros.
    rewrite Mtrans_help; auto.
    (* x=j时，即n=j，表示对角线 *)
    rewrite ! Mfill_help; auto.
    destruct (j =? i) eqn:eq. destruct (i =? j) eqn:eq1. auto.
    apply Nat.eqb_eq in eq.
    apply Nat.eqb_neq in eq1. omega.
    destruct (i =? j) eqn:eq1. auto.
    apply Nat.eqb_neq in eq.
    apply Nat.eqb_eq in eq1. omega.
    auto.
Qed.
```

图 9 单位矩阵转置不变性质的证明过程

Fig. 9 Proving process of identity matrix transpose invariance

如图 10 所示，依据单位矩阵转置不变性质，证明转置仍可逆性质。

```
Theorem Inversion_trans_mult_inv_correct:
  forall (MA A_inv A_inv_inv:Mat A n n),
    Inversion MA = Some A_inv ->
    Mtrans MA M* Mtrans A_inv M= MI.
  Proof.
    intros.
    rewrite <-matrix_trans_mul.
    assert (A_inv M* MA M= MI).
    {
      apply Inversion_correct in H.
      assumption.
    }
    unfold MI.
    rewrite <-I_trans_I.
    apply Mtrans_mor.
    auto.
Qed.
```

图 10 转置仍可逆性质的证明过程

Fig. 10 Proving process of reversibility of transpose

从上述例子可以发现新矩阵等价定义对引理证明的简化，并且该等价定义亦然能够作用于更复杂的矩阵性质证明，以及逆矩阵性质证明，它能够很大程度上化简逆矩阵函数性质证明过程，减少证明工作。

结束语 本文分析并对比了现有高阶定理证明器中的矩阵形式化工作，相较而言，利用函数表示法的矩阵形式化方法是一种较成熟的形式化方式，可以方便直观地表示复杂矩阵，具备表示复杂矩阵的可能性，但缺乏实际应用能力以及软件移植性；利用 Record 类型的矩阵形式化方法易于理解并且方便使用，记录类型更容易抽取出合理的 Ocaml 代码，具备良好的工程数学基础。本文提出基于 Record 的矩阵函数表示法，完成初等矩阵的形式化，首次实现了高斯约旦消元法求解逆矩阵的形式化，并实现了首个形式化验证下的逆矩阵软件

函数库。相比现有的逆矩阵形式化工作，以代价更小且时间复杂度更低的方式，完成了逆矩阵形式化函数库的构建，并在实际应用中进行了初步探索。之后将继续完善相关引理，将函数库整合入飞行控制系统的形式化工作，是我们未来的研究重点。

参 考 文 献

[1] CHEN G，QIU Z Y，SONG X Y，et al. A Report from Head Start：Formalizing Engineering Mathematics[J]. Communications of China Computer Society，2017：92-93.

[2] ALMEIDA JB，FRADE M J，PINTO J S，et al. An overview of formal methods tools and techniques[C]// Rigorous Software Development. London：Springer，2011：15-44.

[3] HATEL P H，MOREAU L. Formalizing the safety of Java，the Java virtual machine，and Java card[J]. ACM Computing Surveys(CSUR)，2001，33(4)：15-44.

[4] BERTOT Y，CASTERAN P. Interactive Theorem Proving and Program Development[M]// Interactive theorem proving and program development. Coq'Art：The Calculus of Inductive Constructions. Springer，2004.

[5] LI L M，SHI Z P，GUAN Y，et al. Formalizing the Matrix Inversion Based on the Adjugate Matrix in HOL4[C]// 8th International Conference on Intelligent Information Processing(IIP). 2014：178-186.

[6] MA Z W，CHEN G. Matrix Formalization Based on Coq Record[J]. Computer Science，2019，46(7)：139-145.

[7] SHI Z P，LIU Z K，GUAN Y，et al. Formalization of Function Matrix Theory in HOL[J]. Journal of Applied Mathematics，2014，SI(1)：1-10.

[8] KANG X N，SHI Z P，YE S W，et al. Formalization of Matrix Transformation Theory in HOL4[J]. Computer Simulation，2014(3)：1-16.

[9] YANG X M，GUAN Y，SHI Z P，et al. Higher-order Logic Formalization of Function Matrix and its Calculus[J]. Computer Science，2016(11)：24-29.

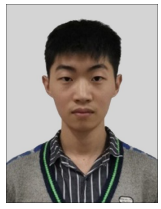
[10] BIHA S O. Finite groups representation theory with Coq[C]// 8th International Conference on Mathematical Knowledge Management. Berlin：Spriger，2009：438-452.

[11] HERAS J，POZA M，DÉNÈS M. Incidence Simplicial Matrices Formalized in Coq/SSReflect[M]. Spriner_verlag，2011：30-44.

[12] DENES M，BERTOT Y. Experiments with computable matrices in the Coq system[R]. France：INRIA，2011：2-27.

[13] BOLDO S，LELAY C，MELQUIOND G. Coquelicot：A User-Friendly Library of Real Analysis for Coq[J]. Mathematics in Computer Science，2015，9(1)：41-62.

[14] MA Y Y，MA Z W，CHEN G. Formalization of Operations of Block Matrix Based on Coq[J]. Ruan Jian Xue Bao/Journal of Software，2021，32(6)：1882-1909.



SHEN Nan，born in 1998，postgraduate. His main research interests include formal methods and so on.

CHEN Gang，born in 1958，Ph.D，professor，is a distinguished member of China Computer Federation. His main research interests include formal methods and so on.