Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 201214, 10 pages
http://dx.doi.org/10.1155/2014/201214

Research Article

Formalization of Function Matrix Theory in HOL

Zhiping Shi,[1,2] Zhenke Liu,[1] Yong Guan,[1] Shiwei Ye,[3] Jie Zhang,[4] and Hongxing Wei[5]

[1] *Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, Beijing 100048, China*
[2] *Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China*
[3] *College of Information Science and Engineering, Graduate University of Chinese Academy of Sciences, Beijing 100049, China*
[4] *College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China*
[5] *School of Mechanical Engineering and Automation, Beijing University of Aeronautics and Astronautics, Beijing 100191, China*

Correspondence should be addressed to Zhiping Shi; shizhiping@gmail.com

Received 12 January 2014; Accepted 22 April 2014; Published 24 July 2014

Academic Editor: Guiming Luo

Function matrices, in which elements are functions rather than numbers, are widely used in model analysis of dynamic systems such as control systems and robotics. In safety-critical applications, the dynamic systems are required to be analyzed formally and accurately to ensure their correctness and safeness. Higher-order logic (HOL) theorem proving is a promise technique to match the requirement. This paper proposes a higher-order logic formalization of the function vector and the function matrix theories using the HOL theorem prover, including data types, operations, and their properties, and further presents formalization of the differential and integral of function vectors and function matrices. The formalization is implemented as a library in the HOL system. A case study, a formal analysis of differential of quadratic functions, is presented to show the usefulness of the proposed formalization.

1. Introduction

Being operators of linear space transformation, matrices have extended their applications in many science fields such as physics, mechanics, optics, the probability theory, and many engineering fields such as computer graphics, signal processing, and robotics. In the applications, dynamic system modeling requests function matrices, in which elements are functions rather than constants.

Traditionally, function matrix computations are dealt with by numerical analysis algorithms or computer algebra algorithms, yet the absolute precision in the real number field can never be reached because of round-off error, approximate algorithms to address large-scale issues, and so on. On the other hand, analysis of function matrix based models has been carried out with paper and pencil, as is quite tedious and error-prone. A tiny error or inaccuracy, however, may result in failure or even loss of lives in highly sensitive and safety-critical engineering applications. Mechanical theorem proving, on the contrary, is capable of performing precise and scalable analysis.

Mechanical theorem proving has been considered a promising and powerful method of formal proofs in pure mathematics or system analysis and verification [1–5]. Systems or any proof goals need to be modeled formally before they are verified by theorem provers, and theorem provers work based on logic theorem libraries of mathematics. It is indispensable to formalize function matrix theory before formal verifying the systems based on the theory. Real matrices have been formalized in many theorem provers. Nakamura et al. [6] presented the formalization of the matrix theory in Mizar in 2006. The COQ system initiated to provide matrices in recent years [7]. Harrison presented the formalization of Euclidean space in the HOL Light system in 2005 [8]. In Isabelle/HOL [9], some basic matrix theory has been formalized [10, 11]. We have developed the basic matrix theory in HOL theorem prover [12]. However, no formalized function matrix theory has yet been reported in literatures.

HOL is one of the most popular theorem provers and has a lot of successful applications. The newest version of the HOL is named HOL4. This paper introduces the formalization of the function matrix theory in HOL4, including formalization of definitions and properties of function vectors and function matrices as well as their arithmetic operations and differential. This work is jointly based on the realTheory library, limTheory library, and fcpTheory library in HOL4.

The formalization of the function vector is proposed in Section 2, while the formalization of the function matrix occupied Section 3. Section 4 proposes formal definitions and properties of differential and integral of function matrices (vectors). As a case study, formal proof of differential of quadratic functions is shown in Section 5 to demonstrate the usefulness of the formalized theory. Finally, the paper draws the conclusion in Section 6.

2. Formalization of Function Vectors

Formal definitions and properties of function vectors and their operations are proposed in this section.

A vector with functions of a variable x as elements is called a function vector, denoted by

$$\alpha(x) = (\alpha_1(x), \ldots, \alpha_i(x), \ldots, \alpha_n(x)). \tag{1}$$

The difference between a function vector and a real vector is that the elements of the function vector are functions although a real number could also be seen as a constant function. In HOL4, "α -> β" donates a function type with domain α and range β. In this paper the elements of a function vector are real functions, and the data type of functions is denoted by "real -> real."

In HOL4, the library fcpTheory provides an operator "∗∗" to construct multidimensional data types. So, the function vector data type is defined by

```
Hol_type: (real -> real) ∗ ∗ 'n,
```

where "'n" is a type variable denoting the dimension of function vectors, and the actual dimension can be retrieved by the function dimindex (:'n).

According to the data type definition above, for a function vector **v** of this type, the element v_i at position i is denoted by "v %% i" or "v ' i" in HOL4. Based on the definition, the arithmetic operations and their properties of function vectors are formalized below.

Based on the type definition, we present the formal definitions of the arithmetic operations of function vectors and the formal definitions of some special function vectors, such as the base vectors and the zero vectors in HOL4. In this paper, **fv**, **fv**1, and **fv**2 denote function vectors, f and g real functions, and k and l real numbers. To simplify the definitions of arithmetic operations, two mapping functions are given in Definitions 1 and 2, which expose their elements to operations of function vectors.

Definition 1 (fvector_map_def). |- !f fv. fvector_map f fv = FCP i. (\x. f (fv ' i x)).

Definition 2 (fvector_map2_def). |- !f fv1 fv2. fvector_map2 f fv1 fv2 = FCP i. (\x. f (fv1 ' i x) (fv2 ' i x)).

Definition 1 is a unary operation and Definition 2 is a binary operation on function vectors. Based on the two definitions, the arithmetic operations of function vectors could be defined concisely. "~", "+," and "-" are overloaded for negative, addition, and subtraction of function vectors.

Definition 3 (fvector_neg_def). |- $ ~ = fvector_map numeric_negate.

Definition 4 (fvector_add_def). |- $ + = fvector_map2 $+.

Definition 5 (fvector_sub_def). |- $ - = fvector_map2 $-.

Two function vectors can do inner product operation, calculated as the following formula:

$$v(x)u(x) = \sum_{i=0}^{n} v_i(x)u_i(x). \tag{2}$$

In HOL4, "∗∗" is used to represent the inner product operator.

Definition 6 (fvector_dot_def). |- !fv1 fv2. fv1 ∗∗ fv2 = (\x. sum (0, dimindex (:'n)) (\i. fv1 ' i x ∗ fv2 ' i x)).

A function vector can be multiplied by a scalar. Here, we formalize the operations multiplying function vectors by real numbers and real functions on left and on right, respectively.

Definition 7 (fvector_mul_lk_def). |- !k fv. k ∗∗ fv = FCP i. (\x. k ∗ fv ' i x).

Definition 8 (fvector_mul_rk_def). |- !fv k. fv ∗∗ k = FCP i. (\x. fv ' i x ∗ k).

Definition 9 (fvector_mul_lkx_def). |- !kx fv. kx ∗∗ fv = FCP i. (\x. kx x ∗ fv ' i x).

Definition 10 (fvector_mul_rkx_def).

|- !fv kx. fv ∗∗ kx = FCP i.(\x. fv ' i x ∗ kx x),

where kx denotes a real function contrasting k for a real number.

We present the definitions of the zero vectors and the basis vectors, and the elements of these function vectors are the constant functions 0 and 1, respectively.

Definition 11 (fvector_0_def). |- fvector_0 = FCP i. (\x. 0).

Definition 12 (fvector_basis_def). |- !k. fvector_basis k = FCP i. if i = k then (\x. 1) else (\x. 0).

It is useful to compute the value of a function vector at a certain x, as is defined by Definition 13.

Definition 13 (compute_fvector_def). |- !fv x. compute_fvector fv x = FCP i. fv ' i x.

On the basis of the definitions formalized above, we formalize a number of the operations' properties and all the properties are proven to be HOL4 theorems. Most of the properties are of linearity and direct-viewing. We list parts of the properties in Table 1.

Table 1: Part operations' properties of function vectors.

Property name	Formalization
FVECTOR_NEG	|- !fv. ~fv = −1 ** fv
FVECTOR_ADD_MUL_LK	|- !fv1 fv2 k. k ** (fv1 + fv2) = k ** fv1 + k ** fv2
FVECTOR_ADD_RDISTRIB	|- !fv1 fv2 fv3. (fv1 + fv2) ** fv3 = (\x. (fv1 ** fv3) x + (fv2 ** fv3) x)
FVECTOR_ADD_ASSOC	|- !fv1 fv2 fv3. fv1 + fv2 + fv3 = fv1 + (fv2 + fv3)
FVECTOR_SUB_LZERO	|- !fv. fvector_0 − fv = ~fv
FVECTOR_DOT_FBASIS	|- !fv k x. k < dimindex (:'n) ==> (fv ** fvector_basis k = fv ' k)
FVECTOR_DOT_FCP	|- ($FCP fv1 ** fv2 = (\x. sum (0,dimindex (:'n)) (\i. fv1 i x * fv2 ' i x))) ∧ (fv2 ** $FCP fv1 = (\x. sum (0,dimindex (:'n)) (\i. fv2 ' i x * fv1 i x)))
FVECTOR_ADD_INDEX	|- !fv1 fv2 i. i < dimindex (:'n) ==> ((fv1 + fv2) ' i = (\x. fv1 ' i x + fv2 ' i x))
FVECTOR_SUB_INDEX	|- !fv1 fv2 i. i < dimindex (:'n) ==> ((fv1 − fv2) ' i = (\x. fv1 ' i x − fv2 ' i x))
FVECTOR_NEG_NEG	|- !fv. ~~fv = fv
FVECTOR_ADD_MUL_LKX	|- !fv1 fv2 kx. kx ** (fv1 + fv2) = kx ** fv1 + kx ** fv2
FVECTOR_ADD_MUL_RKX	|- !fv1 fv2 kx. (fv1 + fv2) ** kx = fv1 ** kx + fv2 ** kx
FVECTOR_MUL_LRADD	|- !fv k l. (k + l) ** fv = k ** fv + l ** fv
FVECTOR_MUL_RRADD	|- !fv k l. fv ** (k + l) = fv ** k + fv ** l
FVECTOR_MUL_LFADD	|- !fv f g. (\x. f x + g x) ** fv = f ** fv + g ** fv
FVECTOR_ADD_RDISTRIB	|- !fv1 fv2 fv3. (fv1 + fv2) ** fv3 = (\x. (fv1 ** fv3) x + (fv2 ** fv3) x)
FVECTOR_SUB_LDISTRIB	|- !fv1 fv2 fv3. fv1 ** (fv2 − fv3) = (\x. (fv1 ** fv2) x − (fv1 ** fv3) x)
FVECTOR_SUB_RDISTRIB	|- !fv1 fv2 fv3. (fv1 − fv2) ** fv3 = (\x. (fv1 ** fv3) x − (fv2 ** fv3) x)
FVECTOR_DOT_LMUL_K	|- !fv1 fv2 k. (\x. k * (fv1 ** fv2) x) = (k ** fv1) ** fv2
FVECTOR_DOT_LMUL_KX	|- !fv1 fv2 k. (\x. k x * (fv1 ** fv2) x) = (k ** fv1) ** fv2
FVECTOR_MUL_LK_ASSOC	|- !fv k l. k ** l ** fv = (k * l) ** fv
FVECTOR_MUL_LKX_ASSOC	|- !fv f g. f ** g ** fv = (\x. f x * g x) ** fv
FVECTOR_DOT_COMM	|- !fv1 fv2. fv1 ** fv2 = fv2 ** fv1
FVECTOR_EQ	|- !fv1 fv2. (fv1 = fv2) <=> (fv1 − fv2 = fvector_0)
FVECTOR_EQ2	|- !fv1 fv2. (fv1 = fv2) <=> !i. i < dimindex (:'n) ==> (fv1 ' i = fv2 ' i)
FVECTOR_ADD_LID	|- !fv. fvector_0 + fv = fv
FVECTOR_ADD_RID	|- !fv. fv + fvector_0 = fv
FVECTOR_ADD_NEG	|- !fv. fv + ~fv = fvector_0
FVECTOR_ADD_NEG2	|- !fv1 fv2. fv1 + ~fv2 = fv1 − fv2
FVECTOR_SUB_ADD	|- !fv1 fv2. fv1 − fv2 + fv2 = fv1
FVECTOR_MUL_L1	|- !fv. 1 ** fv = fv
FVECTOR_LNEG_UNIQ	|- !fv1 fv2. (fv1 + fv2 = fvector_0) <=> (fv1 = ~fv2)
FVECTOR_RNEG_UNIQ	|- !fv1 fv2. (fv1 + fv2 = fvector_0) <=> (fv2 = ~fv1)
FVECTOR_MULK_COMM	|- !fv k. fv ** k = k ** fv
FVECTOR_MULKX_COMM	|- !fv f. fv ** f = f ** fv
FVECTOR_EXIST_NEG	|- !fv. ?fv'. fv + fv' = fvector_0
FVECTOR_FVECTOR_0_DOT	|- !fv. fvector_0 ** fv = (\x. 0)
COMPUTE_FVEC_MUL_MATRIX	|- !fv A x. compute_fvector fv x ** A = compute_fvector (fv ** A) x
COMPUTE_VEC_MUL_FVEC	|- !fv v x. v ** compute_fvector fv x = (v ** fv) x

3. Function Matrices

Like defining function vectors, "**" is used again to define function matrices. A function matrix takes function vectors with data type "(real -> real) * * 'n" as elements. So, the data type of function matrices is formally defined using "* *" twice as

```
Hol_type: ((real -> real) * * 'n) * * 'm.
```

This defines a function matrix with dimindex (:'n) rows and dimindex (:'m) columns. Similar to function vectors, "A %% i %% j" or "A ' i ' j" refers to the element of the *i*th row and *j*th column of the function matrix *A*.

Based on the type definition, we present formal definitions of the arithmetic operations of function matrices, including negative, addition, subtraction, transposition and multiplication by function matrices, function vectors, and scalars and functions. And the formal definitions of the special function matrices, the identity matrices, and the zero matrixes are presented. In addition, the function matrices' inversion is formally defined. In this paper, fm, fm1, and fm2 symbolize function matrices, f and g functions, and k and l real numbers. Two mapping functions are defined to make formalizations of negative, addition, and subtraction concise, like in the function vectors case.

Definition 14 (fmatrix_map_def). |- !f fm. fmatrix_map f fm = FCP i j. (\x. f (fm ' i ' j x)).

Definition 15 (fmatrix_map2_def). |- !f fm1 fm2. fmatrix_map2 f fm1 fm2 = FCP i j. (\x. f (fm1 ' i ' j x)(fm2 ' i ' j x)).

Definition 16 (fmatrix_neg_def). |- fmatrix_neg = fmatrix_map numeric_negate.

Definition 17 (fmatrix_add_def). |- $+ = fmatrix_map2 $+.

Definition 18 (fmatrix_sub_def). |- $- = fmatrix_map2 $-.

Multiplication of function matrices is based on the multiplication of rows and columns of the function matrices. The functions to retrieve a certain row or column of a function matrix are formalized based on FCP.

Definition 19 (fun_row_def). |- !fm k. fun_row fm k = FCP j. fm ' k ' j.

Definition 20 (fun_column_def). |- !fm k. fun_column fm k = FCP i. fm ' i ' k.

Definition 21 (fmatrix_prod_def). |- !fm1 fm2. fm1 ** fm2 = FCP i j. fun_row fm1 i ** fun_column fm2 j.

The function matrices can be multiplied with different data types including function factors, real numbers, and real functions. The formal definitions are as follows.

Definition 22 (fmatrix_mul_lk_def). |- !k fm. k ** fm = FCP i j. (\x. k * fm ' i ' j x).

Definition 23 (fmatrix_mul_rk_def). |- !fm k. fm ** k = FCP i j. (\x. fm ' i ' j x * k).

Definition 24 (fmatrix_mul_lkx_def). |- !k fm. k ** fm = FCP i j. (\x. k x * fm ' i ' j x).

Definition 25 (fmatrix_mul_rkx_def). |- !fm k. fm ** k = FCP i j. (\x. fm 'i ' j x * k x).

Definition 26 (fvector_fmatrix_prod_def). |- !fv fm. fv ** fm = FCP i. fv ** fun_column fm i.

Definition 27 (fmatrix_fvector_prod_def). |- !fm fv. fm ** fv = FCP i. fun_row fm i ** fv.

As shown below, we present definitions of the identity function matrix, the zero function matrix, transposed matrices, and reversibility of function matrices.

Definition 28 (fmatrix_0_def). |- fmatrix_0 = FCP i j. (\x. 0).

Definition 29 (fmatrix_E_def). |- fmatrix_E = FCP ij. if i = j then (\x. 1) else (\x. 0).

Definition 30 (transp_fmatrix_def). |- !fm. transp_fmatrix fm = FCP i j. fm ' j ' i.

Definition 31 (fmatrix_inv_def). |- !fm. fmatrix_inv fm <=> ?fm'. (fm ** fm' = fmatrix_E) $\wedge$ (fm' ** fm = fmatrix_E).

Computing values of function matrices on a certain x produces real matrices, as is defined by Definition 32.

Definition 32 (compute_fmatrix_def). |- !fm x. compute_fmatrix fm x = FCP i j. fm ' i ' j x.

Function vectors and function matrices can operate with real vectors and real matrices, which are special function vectors and function matrices.

Based on the definitions above, we formalize many linear properties, which are useful in proving new theorems.

Property 1 (TRANSP_FMATRIX_FCP). The relation between the elements of a function matrix and its transpose is as follows:

|- !fm. transp_fmatrix (FCP i j. fm i j) = FCP i j. fm j i.

Property 2 (TRANSP_TRANSP_FMATRIX). Transposing a function matrix twice changes nothing:

|- !fm. transp_fmatrix (transp_fmatrix fm) = fm.

Property 3 (FMATRIX_PROD_FVECTOR). For any function matrix and function vector, denoted by $A(x)$ and $v(x)$, respectively, it is held that

$$A(x)\,v(x) = v(x)\,A^{T}(x) \qquad (3)$$

|- !fm fv. fm ** fv = fv ** transp_fmatrix fm.

Property 4 (FVECTOR_PROD_FMATRIX). Swapping the positions of the function matrix and the function vector, it is held that

$$v(x)\,A(x) = A^{T}(x)\,v(x) \qquad (4)$$

|- !fm fv. fv ** fm = transp_fmatrix fm ** fv.

Property 5 (TRANSP_FMATRIX_PROD). For any function matrix $A(x)$, it is held that

$$\left[A^{T}(x)\,A(x)\right]^{T} = A^{T}(x)\,A(x) \qquad (5)$$

|- !fm. transp_fmatrix (transp_fmatrix fm ** fm) = transp_fmatrix fm ** fm.

Property 6 (TRANSP_FMATRIX_COLUMN). The rows of a function matrix equal the corresponding columns of its transpose:

```
|- !fm i.i < dimindex (:'m) ==> (fun_column (transp_
fmatrix fm) i = fun_row fm i).
```

Property 7 (TRANSP_FMATRIX_ROW). The columns of a function matrix equal the corresponding rows of its transpose:

```
|- !fm i.i < dimindex (:'n) ==> (fun_row (transp_fmatrix
fm) i = fun_column fm i).
```

Now, we present a special property (Property 8). A function matrix, denoted by $A(x)$, can be formed by column function vectors, written as

$$A(x) = \left[v_1(x), \ldots, v_i(x), \ldots, v_n(x)\right]. \tag{6}$$

So, multiplying a function matrix by its transpose can be calculated as

$$A^T(x)\,A(x) = \left[v_i(x)\,v_j(x)\right]_{n\times n}. \tag{7}$$

The property is formalized in Property 8.

Property 8 (FMATRIX_ROW_PROD). |- !fm. transp_fmatrix fm ** fm = FCP i j. fun_column fm i ** fun_column fm j.

Property 9 (FMATRIX_VECTOR_DOT_PROD_FMATRIX). For multiplication of a function matrix and a function vector, it is held that

$$A(x)\,v(x)\,A(x) = v(x)\,A^T(x)\,A(x) \tag{8}$$

```
|- !fm v. (fm ** v) ** fm = v ** transp_fmatrix fm
**fm.
```

Property 9 could be derived by Property 3.

Property 10 (COMPUTE_FVEC_MUL_MATRIX). A function vector multiplies a real matrix:

```
|- !fv A x. compute_fvector fv x ** A = compute_
fvector (fv ** A) x.
```

Property 11 (COMPUTE_VEC_MUL_FVEC). A real vector multiplies a function vector

```
|- !fv v x. v ** compute_fvector fv x = (v ** fv) x.
```

Other properties are listed in Table 2.

In practice, function matrices (and vectors) often operate with real matrices (and vectors), and formalizations of the operations are presented as follows.

Definition 33 (vec_mul_fvector_def). |- !v fv. v ** fv = (\x. sum (0,dimindex (:'n)) (\i. v 'i * fv 'i x)).

Definition 34 (fvector_mul_vec_def). |- !fv v. fv ** v = (\x. sum (0,dimindex (:'n)) (\i. fv 'i x * v 'i)).

Definition 35 (vec_mul_fmatrix_def). |- !v fm. v ** fm = FCP i. v ** fun_column fm i.

Definition 36 (fmatrix_mul_vec_def). |- !fm v. fm ** v = FCP i. fun_row fm i ** v.

Definition 37 (matrix_mul_fvector_def). |- !A fv. A ** fv = FCP i. row A i ** fv.

Definition 38 (fvector_mul_matrix_def). |- !fv A. fv ** A = FCP i. fv ** column A i.

Definition 39 (matrix_mul_fmatrix_def). |- !A fm. A ** fm = FCP i j. row A i ** fun_column fm j.

Definition 40 (fmatrix_mul_matrix_def). |- !fm A. fm ** A = FCP i j. fun_row fm i ** column A j.

4. Formalization of Differential and Integral of Function Matrices

A function vector or function matrix is differentiable or integrable supposing all its elements are differentiable or integrable. This section presents the formalizations of differential and integral of function vectors and function matrices based on that of real functions.

A function vector, denoted by $v(x) = (a_i(x))_n$, is derivable at $x = x_0$ if all its elements $a_i(x)$ $(i = 1, 2, \ldots, n)$ are derivable at $x = x_0$, and the derivative can be written as

$$\begin{aligned} v'(x) = \left.\frac{dv(x)}{dx}\right|_{x=x_0} &= \lim_{\Delta x\to 0}\frac{v(x_0+\Delta x)-v(x_0)}{\Delta x} \\ &= \left(a_1'(x)\ \ a_2'(x)\ \ \cdots\ \ a_n'(x)\right). \end{aligned} \tag{9}$$

A function vector, denoted by $v(x) = (a_i(x))_n$, is integrable in $[t_0, t_1]$ if all its elements $a_i(x)$ $(i = 1, 2, \ldots, n)$ are integrable in $[t_0, t_1]$, and the integral can be written as

$$\int_{t_0}^{t_1} v(x)\,dt = \left(\int_{t_0}^{t_1} a_i(x)dt\right)_n. \tag{10}$$

According to the mathematics descriptions, we formally define the differential and integral of function vectors based on that of real functions. The differential and integral of a real function are denoted by "diffl" and "integral" [12], respectively, in HOL4.

Definition 41 (fvector_diffl). For anyone function vector **fv**, it is the case that the differential of **fv** at x is the real vector **v** if and only if the differential of members of **fv** at x equals the corresponding members of **v** at x. In HOL4, it is said that

```
|- !fv v x. (fv fvector_diffl v) x <=> !i. i < dimindex
(:'n) ==> (fv '  i diffl v ' i) x.
```

Definition 42 (fvector_integral). Calculating the integral of a function vector **fv** in $[a, b]$ is equal to calculating the integral of all members of **fv** in $[a, b]$. In HOL4, it is said that

```
|- !a b fv. fvector_integral (a,b) fv = FCP i. integral
(a,b) (fv '  i).
```

Table 2: Properties of operations of function matrices.

Property name	Formalization
COMPUTE_FMATRIX_MUL_EQ	|- !fm1 fm2 x. compute_fmatrix fm1 x ** compute_fmatrix fm2 x = compute_fmatrix (fm1 ** fm2) x
FMATRIX_ADD_INDEX	|- !fm1 fm2 i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==> ((fm1 + fm2) ' i ' j = (\x. fm1 ' i ' j x + fm2 ' i ' j x))
FMATRIX_SUB_INDEX	|- !fm1 fm2 i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==> ((fm1 − fm2) ' i ' j = (\x. fm1 ' i ' j x − fm2 ' i ' j x))
FMATRIX_ROW_ADD	|- !fm1 fm2 i. i < dimindex (:'m) ==> (fun_row fm1 i + fun_row fm2 i = fun_row (fm1 + fm2) i)
FMATRIX_COLUMN_ADD	|- !fm1 fm2 i. i < dimindex (:'n) ==> (fun_column fm1 i + fun_column fm2 i = fun_column (fm1 + fm2) i)
FMATRIX_ROW_SUB	|- !fm1 fm2 i. i < dimindex (:'m) ==> (fun_row fm1 i − fun_row fm2 i = fun_row (fm1 − fm2) i)
FMATRIX_COLUMN_SUB	|- !fm1 fm2 i. i < dimindex (:'n) ==> (fun_column fm1 i − fun_column fm2 i = fun_column (fm1 − fm2) i)
FMATRIX_NEG	|- !fm. ~fm = −1 ** fm
FMATRIX_NEG_NEG	|- !fm. ~~fm = fm
FMATRIX_MUL_K_EQ	|- !fm k. fm ** k = k ** fm
FMATRIX_MUL_KX_EQ	|- !fm f. fm ** f = f ** fm
FMATRIX_ADD_COMM	|- !fm1 fm1. fm1 + fm2 = fm2 + fm1
FMATRIX_ADD_ASSOC	|- !fm1 fm2 fm3. fm1 + (fm2 + fm3) = fm1 + fm2 + fm3
FMATRIX_ADD_MUL_LK	|- !fm1 fm2 k. k ** (fm1 + fm2) = k ** fm1 + k ** fm2
FMATRIX_ADD_MUL_RK	|- !fm1 fm2 k. (fm1 + fm2) ** k = fm1 ** k + fm2 ** k
FMATRIX_ADD_MUL_LKX	|- !fm1 fm2 kx. kx ** (fm1 + fm2) = kx ** fm1 + kx ** fm2
FMATRIX_ADD_MUL_RKX	|- !fm1 fm2 kx. (fm1 + fm2) ** kx = fm1 ** kx + fm2 ** kx
FMATRIX_ADD_MUL_LFVEC	|- !fm1 fm2 fv. fv ** (fm1 + fm2) = fv ** fm1 + fv ** fm2
FMATRIX_ADD_MUL_RFVEC	|- !fm1 fm2 fv. (fm1 + fm2) ** fv = fm1 ** fv + fm2 ** fv
FMATRIX_SUB_MUL_LFVEC	|- !fm1 fm2 fv. fv ** (fm1 − fm2) = fv ** fm1 − fv ** fm2
FMATRIX_SUB_MUL_RFVEC	|- !fm1 fm2 fv. (fm1 − fm2) ** fv = fm1 ** fv − fm2 ** fv
FMATRIX_MUL_LRADD	|- !fm k l. (k + l) ** fm = k ** fm + l ** fm
FMATRIX_MUL_RRADD	|- !fm k l. fm ** (k + l) = fm ** k + fm ** l
FMATRIX_MUL_LFADD	|- !fm f g. (\x. f x + g x) ** fm = f ** fm + g ** fm
FMATRIX_MUL_RFADD	|- !fm f g. fm ** (\x. f x + g x) = fm ** f + fm ** g
FMATRIX_MUL_RFVADD	|- !fm fv1 fv2. fm ** (fv1 + fv2) = fm ** fv1 + fm ** fv2
FMATRIX_MUL_LFVADD	|- !fm fv1 fv2. (fv1 + fv2) ** fm = fv1 ** fm + fv2 ** fm
FMATRIX_ADD_LDISTRIB	|- !fm1 fm2 fm3. fm1 ** (fm2 + fm3) = fm1 ** fm2 + fm1 ** fm3
FMATRIX_ADD_RDISTRIB	|- !fm1 fm2 fm3. (fm1 + fm2) ** fm3 = fm1 ** fm3 + fm2 ** fm3
FMATRIX_MUL_LMUL_K	|- !fm1 fm2 k. k ** fm1 ** fm2 = (k ** fm1) ** fm2
FMATRIX_MUL_RMUL_K	|- !fm1 fm2 k. k ** fm1 ** fm2 = fm1 ** k ** fm2
FMATRIX_MUL_NEG	|- !fm1 fm2. ~fm1 ** fm2 = fm1 ** ~fm2
FMATRIX_NEG_PROD	|- !fm1 fm2. ~fm1 ** fm2 = ~(fm1 ** fm2)
FMATRIX_MUL_LMUL_KX	|- !fm1 fm2 kx. kx ** fm1 ** fm2 = (kx ** fm1) ** fm2
FMATRIX_MUL_LK_ASSOC	|- !fm k l. k ** l ** fm = (k * l) ** fm
FMATRIX_MUL_LKX_ASSOC	|- !fm f g. f ** g ** fm = (\x. f x * g x) ** fm
FMATRIX_ADD_LID	|- !fm. fmatrix_0 + fm = fm
FMATRIX_ADD_RID	|- !fm. fm + fmatrix_0 = fm
FMATRIX_ADD_NEG	|- !fm. fm + ~fm = fmatrix_0
FMATRIX_ADD_NEG2	|- !fm1 fm2. fm1 + ~fm2 = fm1 − fm2
FMATRIX_SUB_ADD	|- !fm1 fm2. fm1 − fm2 + fm2 = fm1
FMATRIX_SUB_LZERO	|- !fm. fmatrix_0 − fm = ~fm

Table 2: Continued.

Property name	Formalization
FMATRIX_MUL_L1	|- !fm. 1 ** fm = fm
FMATRIX_MULK_COMM	|- !fm k. fm ** k = k ** fm
FMATRIX_MULKX_COMM	|- !fm kx. fm ** kx = kx ** fm
FVECTOR_PROD_FMATRIX	|- !fm fv. fv ** fm = transp_fmatrix fm ** fv
FMATRIX_FVECTOR_0_PROD	|- !fm. fvector_0 ** fm = fvector_0
FMATRIX_ROW_PROD	|- !fm. transp_fmatrix fm ** fm = FCP i j. fun_column fm i ** fun_column fm j
TRANSP_FMATRIX_COLUMN	|- !fm i. i < dimindex (:'m) ==> (fun_column (transp_fmatrix fm) i = fun_row fm i)
TRANSP_FMATRIX_FVECTOR_PROD	|- !fm fv. fm ** fv = fv ** transp_fmatrix fm
TRANSP_FMATRIX_PROD	|- !fm. transp_fmatrix (transp_fmatrix fm ** fm) = transp_fmatrix fm ** fm
TRANSP_FMATRIX_ROW	|- !fm i. i < dimindex (:'n) ==> (fun_row (transp_fmatrix fm) i = fun_column fm i)

Again, the differentiability and integrability of function vectors are formally defined based on those of real functions. The differentiability and integrability of real functions are denoted by “differentiable” and “integrable” respectively, in HOL4.

Definition 43 (fvector_differentiable). For anyone function vector **fv**, it is the case that **fv** is differentiable at x if and only if all the members of **fv** are differentiable at x. In HOL4, it is said that

```
|- !a b fv. fvector_differentiable fv x <=>

        !a b i. a <= b ∧ i < dimindex (:'n) ==> (fv ' i)
        differentiable x.
```

Definition 44 (fvector_integrable). For anyone function vector **fv**, it is the case that **fv** is integrable in $[a, b]$ if and only if all the members of **fv** are integrable in $[a, b]$. In HOL4, it is said that

```
|- !a b fv. fvector_integrable (a,b) fv <=>

        !a b i. a <= b ∧ i < dimindex (:'n) ==> integrable
        (a,b) (fv ' i).
```

A function matrix, denoted by $A(x) = (a_{ij}(x))_{m*n}$, is derivable at $x = x_0$ if its all elements $a_{ij}(x)$ $(i = 1, 2, \dots, m;$ $j = 1, 2, \dots, n)$ are derivable at $x = x_0$, and the derivative can be written as

$$\begin{aligned} A'(x) &= \left.\frac{dA(x)}{dx}\right|_{x=x_0} \\ &= \lim_{\Delta x \to 0} \frac{A(x_0+\Delta x) - A(x_0)}{\Delta x} \\ &= \begin{bmatrix} a'_{11}(x_0) & a'_{12}(x_0) & \cdots & a'_{1n}(x_0) \\ a'_{21}(x_0) & a'_{22}(x_0) & \cdots & a'_{2n}(x_0) \\ \cdots & \cdots & \cdots & \cdots \\ a'_{m1}(x_0) & a'_{m2}(x_0) & \cdots & a'_{mn}(x_0) \end{bmatrix}. \end{aligned} \tag{11}$$

A function matrix, denoted by $A(x) = (a_{ij}(x))_{m*n}$, is integrable in $[t_0, t_1]$ if all its elements $a_{ij}(x)$ $(i = 1, 2, \dots, m;$ $j = 1, 2, \dots, n)$ are integrable in $[t_0, t_1]$, and the integral can be written as

$$\int_{t_0}^{t_1} A(t)\,dt = \left(\int_{t_0}^{t_1} a_{ij}(t)dt\right)_{m\times n}. \tag{12}$$

Similar to function vectors, we formally define the differential and integral of function matrices based on those of real functions as follows.

Definition 45 (fmatrix_diffl). For anyone function matrix fm, it is the case that the differential of fm at x is a matrix A if and only if the differentials of members of fm at x equal the corresponding members of A. In HOL4, it is said that

```
|- !fm A x. (fm fmatrix_diffl A) x <=>

        !i j. i < dimindex (:'m) ∧ j < dimindex (:'n) ==>
        (fm 'i 'j diffl A 'i 'j) x.
```

Definition 46 (fmatrix_integral). Calculating the integral of a function matrix fm in $[a, b]$ is equal to calculating the integral of all members of fm in $[a, b]$. In HOL4, it is said that

```
|- !a b fm. fmatrix_integral (a,b) fm = FCP i j. integral
(a,b) (fm 'i 'j).
```

Definition 47 (fmatrix_differentiable). For anyone function matrix fm, it is the case that fm is differentiable at x if and only if there exists a matrix A which is the differential of fm at x. In HOL4, it is said that

```
|- !fm x. fm fmatrix_differentiable x <=> ?A. (fm
fmatrix_diffl A) x.
```

Definition 48 (fmatrix_ingegrable). For anyone function matrix fm, it is the case that fm is integrable in $[a, b]$ if and only if all the elements of fm are integrable in $[a, b]$. In HOL4, it is said that

```
|- !a b fm. fmatrix_integrable (a,b) fm <=>

        !a b i j. a <= b ∧ i < dimindex (:'m) ∧ j <
        dimindex (:'n) ==>
            integrable (a,b) (fm ' i ' j).
```

Based on the definitions above, we formalize and prove many properties about differential and integral of function matrices. Some of them are presented as follows.

Uniqueness is one of the most important properties for differential. Differential of a function matrix is unique.

Property 12 (FMATRIX_DIFF_UNIQ). |- !fm A B x. (fm fmatrix_diffl A) x ∧ (fm fmatrix_diffl B) x ==> (A = B).

Suppose $A(x) = (a_{ij}(x))_{m*n}$, $B(x) = (b_{ij}(x))_{m*n}$ are differentiable. It is the case that

$$\frac{d}{dx}[A(x) \pm B(x)] = \frac{dA(x)}{dx} \pm \frac{dB(x)}{dx}. \tag{13}$$

The property is formalized in HOL4 as follows.

Property 13 (DIFF_FMATIRX_ADD). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x

```
==>((fm1 + fm2) fmatrix_diffl (A + B)) x.
```

Property 14 (DIFF_FMATIRX_SUB). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x

```
==>((fm1 - fm2) fmatrix_diffl (A - B)) x.
```

Similar to the differential of product of real functions, the differential of inner product of function vectors is defined by

$$\frac{d}{dx}[v_1(x) v_2(x)] = \frac{dv_1(x)}{dx} v_2(x) + v_1(x) \frac{dv_2(x)}{dx}. \tag{14}$$

In HOL4, it is formalized by Property DIFF_FVECTOR_MUL.

Property 15 (DIFF_FVECTOR_MUL). |- !fv1 fv2 V1 V2 x. (fv1 fvector_diffl V1) x ∧ (fv2 fvector_diffl V2)x ==>

```
((fv1 ** fv2) diffl (V1 ** compute_fvector fv2 x +
V2 ** compute_fvector fv1 x)) x.
```

The differential of the product of a function vector and a matrix is defined by

$$\frac{d}{dx}[v(x) A] = \frac{dv(x)}{dx} A. \tag{15}$$

Property 16 (DIFF_FVEC_MUL_MATRIX). |- !A fv v. (fv fvector_diffl v)(x) ==> ((fv ** A) fvector_diffl (v ** A))(x).

Let $k(x)$ be a real function of x, $A(x)$ is a function matrix, and both $k(x)$ and $A(x)$ are differentiable, and then

$$\frac{d}{dx}[k(x) A(x)] = \frac{dk(x)}{dx} A(x) + k(x) \frac{dA(x)}{dx}. \tag{16}$$

Specially, if $k(x)$ regresses to a constant k, then

$$\frac{d}{dx}[kA(x)] = k\frac{dA(x)}{dx}. \tag{17}$$

In HOL4, the above properties are formalized as follows.

Property 17 (DIFF_FMATIRX_MUL_KX). |- !fm A kx k x. (fm fmatrix_diffl A) x ∧ (kx diffl k) x ==>

```
((kx ** fm) fmatrix_diffl (k ** compute_fmatrix
fm x + A ** kx x)) x.
```

Property 18 (DIFF_FMATIRX_MUL_K). |- !fm A k x. (fm fmatrix_diffl A) x ==> ((k ** fm) fmatrix_diffl (k ** A)) x.

Suppose $A(x)$ and $B(x)$ are differentiable, and $A(x)$ and $B(x)$ are multipliable, and then

$$\frac{d}{dx}[A(x) B(x)] = \frac{dA(x)}{dx} B(x) + A(x) \frac{dB(x)}{dx}. \tag{18}$$

Property 19 (DIFF_FMATRIX_MUL). |- !fm1 fm2 A B x. (fm1 fmatrix_diffl A) x ∧ (fm2 fmatrix_diffl B) x ==>

```
((fm1 ** fm2) fmatrix_diffl (compute_fmatrix fm1 x
** B + A ** compute_fmatrix fm2 x)) x.
```

Suppose $A(x)$ is a function matrix, $x = f(t)$ is a real function of t, and $A(x)$ and $f(t)$ are differentiable, and then

$$\frac{d}{dx} A(x) = \frac{dA(x)}{dx} f'(t) = f'(t) \frac{dA(x)}{dx}. \tag{19}$$

Property 20 (DIFF_FMATRIX_CHAIN). |- !fm g A m x. (fm fmatrix_diffl A) (g x) ∧ (g diffl m) x ==>

```
(fmatrix_o fm g fmatrix_diffl (A ** m)) x.
```

That $A(x)$ is a constant matrix is equivalent to that

$$\frac{dA(x)}{dx} = 0. \tag{20}$$

Property 21 (DIFF_CONST_MATRIX). |- !A x. (matrix_to_fun A fmatrix_diffl matrix_0) x.

If $A(x)$ and its inverse are differentiable, then

$$\frac{dA^{-1}(x)}{dx} = -A^{-1}(x) \frac{dA(x)}{dx} A^{-1}(x). \tag{21}$$

Property 22 (FMATRIX_0_INTEGAL). If a function matrix equals the zero function matrix, then the integral of the function matrix is the zero real matrix. In HOL4, it is formalized by

```
|- !fm a b. a <= b ∧ (fm = fmatrix_0) ==> (fmatrix_
integral (a,b) fm = matrix_0).
```

5. Case Study—Differential of Quadratic Functions

For linear control systems, the mathematical models of their performance indicators are quadratic functions of state

```
val DIFF_QUADRATIC = store_thm(“DIFF_QUADRATIC”,
    “!(fv:'n fun_vector) (v:'n vector) (A:('n,'n) matrix) (t:real).
     (fv fvector_diffl v)(t) ∧ (transp A = A) ==>
     ((fv ** A ** fv) diffl
      (v ** A ** (compute_fvector fv t) + (v ** A) ** (compute_fvector fv t)))(t)”,
    REPEAT GEN_TAC THEN
    RW_TAC std_ss [MATRIX_VECTOR] THEN
    ‘!(fv:'n fun_vector) (A:('n,'n) matrix).
     (transp A = A) ==> (fv ** A ** fv = (fv ** A) ** fv)’
     by REWRITE_TAC [] THENL
    [SRW_TAC [fcpLib.FCP_ss] [fvector_mul_matrix_def] THEN
     SRW_TAC [fcpLib.FCP_ss] [fvector_dot_def] THEN
     ABS_TAC THEN
     MATCH_MP_TAC SUM_EQ THEN
     SRW_TAC [][] THEN
     SRW_TAC [fcpLib.FCP_ss] [fvector_mul_vec_def] THEN
     SRW_TAC [fcpLib.FCP_ss] [matrix_mul_fvec_def] THEN
     SRW_TAC [fcpLib.FCP_ss] [vec_mul_fvector_def] THEN
     GEN_REWR_TAC RAND_CONV [REAL_MUL_COMM] THEN
     REWRITE_TAC [GSYM SUM_CMUL] THEN
     MATCH_MP_TAC SUM_EQ THEN
     SRW_TAC [][] THEN
     DISJ2_TAC THEN
     SRW_TAC [fcpLib.FCP_ss] [row_def, column_def] THEN
     NTAC 3(POP_ASSUM MP_TAC) THEN
     SRW_TAC [fcpLib.FCP_ss] [transp_def] THEN
     PROVE_TAC [REAL_MUL_COMM],ALL_TAC] THEN
    ‘!(fv:'n fun_vector) (A:('n,'n) matrix) t:real.
      (compute_fvector fv t) ** A = compute_fvector (fv ** A) t’
      by REWRITE_TAC [COMPUTE_FVEC_MUL_MATRIX] THEN
    ‘!(fv:'n fun_vector) (v:'n vector) t:real.
      v ** (compute_fvector fv t) = (v ** fv) t’
      by REWRITE_TAC [COMPUTE_VEC_MUL_FVEC] THEN
    ‘!(fv:'n fun_vector) (v:'n vector) (A:('n,'n) matrix) (t:real).
      (fv fvector_diffl v)(t) ==> ((fv ** A) fvector_diffl (v ** A))(t)’
      by REWRITE_TAC [DIFF_FVEC_MUL_MATRIX] THEN
    PROVE_TAC [DIFF_FVECTOR_MUL]);
```

Algorithm 1: Formal proof of the quadratic function differential.

and control variables, and the optimal control problem is called the linear quadratic problem [13]. For example, the differential of quadratic functions is involved in analyzing asymptotic stability of the optimal closed-loop systems. In this section, differential of quadratic functions is formalized.

Let $x = x(t) \in R^n$ be a function vector, and $A = A^T \in R^{n\times n}$ a constant matrix, we formally analyze the differential of the quadratic function $x^T Ax$ with respect to $t_\circ$. Based on the properties of differential of function vectors and matrices, we have

$$\begin{aligned}\frac{d}{dt}\left(x^T Ax\right) &= \frac{dx^T}{dt}Ax + x^T\frac{d}{dt}(Ax)\\ &= \frac{dx^T}{dt}Ax + x^T\left(\frac{dA}{dt}x + A\frac{dx}{dt}\right)\\ &= \frac{dx^T}{dt}Ax + x^T A\frac{dx}{dt}.\end{aligned} \tag{22}$$

The formula is formally proved in HOL4 as shown in Algorithm 1. Following the custom of our formalization, **fv** is employed to denote function vector x and real vector **v** to denote the differential of **fv** at x. $x^T Ax = (x^T A)x$ is proved first to transform the original goal into the differential of the inner product of two function vectors, which has been proven in Property DIFF_FVECTOR_MUL. And the differential of $x^T A$ could be dealt with by Property DIFF_FVEC_MUL_MATRIX.

6. Conclusion

Based on high order logic theorem prover HOL4, this paper formalized the data type definitions and operation definitions of function vectors and function matrices and proved lots of operation properties. This paper also presented the definitions of function matrix differential and integral and their properties. All the formalization was implemented as a library in the HOL4 system. The case study of formal proof of quadratic function illustrated the usefulness of the formalized theory.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the International Cooperation Program on Science and Technology (2010DFB10930, 2011DFG13000), the National Natural Science Foundation of China (61070049, 61170304, 61104035, 61373034, and 61303014), the Natural Science Foundation of the City of Beijing (4122017), the S&R Key Program of the Beijing Municipal Education Commission (KZ201210028036), and the Open Project Program of State Key Laboratory of Computer architecture and the Open Project Program of Guangxi Key Laboratory trusted software.

References

[1] C. Kern and M. R. Greenstreet, "Formal verification in hardware design: a survey," *ACM Transactions on Design Automation of Electronic Systems*, vol. 4, no. 2, pp. 123–193, 1999.

[2] W. Wu and X. Gao, "Mathematics mechanization and applications after thirty years," *Frontiers of Computer Science in China*, vol. 1, no. 1, pp. 1–8, 2007.

[3] J. Liu and H. Lin, "Proof system for applied Pi calculus," in *Theoretical Computer Science*, vol. 323, pp. 229–243, Springer, Berlin, Germany, 2010.

[4] Y. Li, W. N. N. Hung, and X. Song, "A novel formalization of symbolic trajectory evaluation semantics in Isabelle/HOL," *Theoretical Computer Science*, vol. 412, no. 25, pp. 2746–2765, 2011.

[5] L. Chang, Z. Shi, T. Gu, and L. Zhao, "A family of dynamic description logics for representing and reasoning about actions," *Journal of Automated Reasoning*, vol. 49, no. 1, pp. 1–52, 2010.

[6] Y. Nakamura, N. Tamura, and W. Chang, "A theory of matrices of real elements," *Formalized Mathematics*, vol. 14, no. 1, pp. 21–28, 2006.

[7] I. Pasca, "Formally verified conditions for regularity of interval matrices," in *Intelligent Computer Mathematics*, vol. 6167 of *Lecture Notes in Computer Science*, pp. 219–233, Springer, Berlin, Germany, 2010.

[8] J. Harrison, "A HOL theory of Euclidean space," in *Theorem Proving in Higher Order Logics*, vol. 3603 of *Lecture Notes in Computer Science*, pp. 114–129, Springer, Berlin, Germany, 2005.

[9] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: a Proof Assistant for Higher-Order Logic*, vol. 2283 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2002.

[10] S. Obua, *Flyspeck II: the basic linear programs [Ph.D. thesis]*, Technische Universität München, Munich, Germany, 2008.

[11] S. Obua, "Proving bounds for real linear programs in Isabelle/HOL," in *Theorem Proving in Higher Order Logics*, vol. 3603 of *Lecture Notes in Computer Science*, pp. 227–244, Springer, Berlin, Germany, 2005.

[12] Z. Shi, W. Gu, X. Li et al., "The gauge integral theory in HOL4," *Journal of Applied Mathematics*, vol. 2013, Article ID 160875, 7 pages, 2013.

[13] H. U. Shou-song, *Principle of Automatic Control*, Science Press, Beijing, China, 2007.