

Gauss-Jordan Elimination for Matrices Represented as Functions

Tobias Nipkow

December 14, 2021

Abstract

This theory provides a compact formulation of Gauss-Jordan elimination for matrices represented as functions. Its distinctive feature is succinctness. It is not meant for large computations.

1 Gauss-Jordan elimination algorithm

theory *Gauss-Jordan-Elim-Fun*

imports

HOL-Combinatorics.Transposition

begin

Matrices are functions:

type-synonym *'a matrix = nat ⇒ nat ⇒ 'a*

In order to restrict to finite matrices, a matrix is usually combined with one or two natural numbers indicating the maximal row and column of the matrix.

Gauss-Jordan elimination is parameterized with a natural number n . It indicates that the matrix A has n rows and columns. In fact, A is the augmented matrix with $n+1$ columns. Column n is the “right-hand side”, i.e. the constant vector b . The result is the unit matrix augmented with the solution in column n ; see the correctness theorem below.

fun *gauss-jordan* :: (*'a::field*)*matrix ⇒ nat ⇒ ('a)matrix option* **where**

gauss-jordan A $0 = \text{Some}(A)$ |

gauss-jordan A (*Suc* m) =

(*case dropWhile* ($\lambda i. A\ i\ m = 0$) [$0..<Suc\ m$] of

$[] \Rightarrow \text{None}$ |

$p \# - \Rightarrow$

(*let* $Ap' = (\lambda j. A\ p\ j / A\ p\ m)$;

$A' = (\lambda i. \text{if } i=p \text{ then } Ap' \text{ else } (\lambda j. A\ i\ j - A\ i\ m * Ap'\ j))$)

in gauss-jordan (*Fun.swap* $p\ m\ A'$) m))

Some auxiliary functions:

definition *solution* :: ('a::field)matrix \Rightarrow nat \Rightarrow (nat \Rightarrow 'a) \Rightarrow bool **where**
solution A n x = ($\forall i < n. (\sum_{j=0..<n. A\ i\ j * x\ j} = A\ i\ n)$)

definition *unit* :: ('a::field)matrix \Rightarrow nat \Rightarrow nat \Rightarrow bool **where**
unit A m n =
($\forall i\ j::nat. m \leq j \longrightarrow j < n \longrightarrow A\ i\ j = (\text{if } i=j \text{ then } 1 \text{ else } 0)$)

lemma *solution-swap*:

assumes p1 < n p2 < n

shows *solution* (Fun.swap p1 p2 A) n x = *solution* A n x (**is** ?L = ?R)
<proof>

lemma *solution-upd1*:

$c \neq 0 \implies \text{solution } (A(p:= (\lambda j. A\ p\ j / c)))\ n\ x = \text{solution } A\ n\ x$
<proof>

lemma *solution-upd-but1*: $\llbracket ap = A\ p; \forall i\ j. i \neq p \longrightarrow a\ i\ j = A\ i\ j; p < n \rrbracket \implies$
solution ($\lambda i. \text{if } i=p \text{ then } ap \text{ else } (\lambda j. a\ i\ j - c\ i * ap\ j)$) n x =
solution A n x
<proof>

1.1 Correctness

The correctness proof:

lemma *gauss-jordan-lemma*: $m \leq n \implies \text{unit } A\ m\ n \implies \text{gauss-jordan } A\ m = \text{Some } B \implies$
unit B 0 n \wedge *solution* A n ($\lambda j. B\ j\ n$)
<proof>

theorem *gauss-jordan-correct*:

gauss-jordan A n = Some B $\implies \text{solution } A\ n\ (\lambda j. B\ j\ n)$
<proof>

definition *solution2* :: ('a::field)matrix \Rightarrow nat \Rightarrow nat \Rightarrow (nat \Rightarrow 'a) \Rightarrow bool
where *solution2* A m n x = ($\forall i < m. (\sum_{j=0..<m. A\ i\ j * x\ j} = A\ i\ n)$)

definition *usolution* A m n x \longleftrightarrow

solution2 A m n x \wedge ($\forall y. \text{solution2 } A\ m\ n\ y \longrightarrow (\forall j < m. y\ j = x\ j)$)

lemma *non-null-if-pivot*:

assumes *usolution* A m n x **and** q < m **shows** $\exists p < m. A\ p\ q \neq 0$
<proof>

lemma *lem1*:

fixes f :: 'a \Rightarrow 'b::field

shows $(\sum_{x \in A. f\ x * (a * g\ x)}) = a * (\sum_{x \in A. f\ x * g\ x)$

<proof>

```
lemma lem2:  
  fixes f :: 'a ⇒ 'b::field  
  shows  $(\sum x \in A. f\ x * (g\ x * a)) = a * (\sum x \in A. f\ x * g\ x)$   
  <proof>
```

1.2 Complete

```
lemma gauss-jordan-complete:  
   $m \leq n \implies \text{usolution } A\ m\ n\ x \implies \exists B. \text{gauss-jordan } A\ m = \text{Some } B$   
  <proof>
```

Future work: extend the proof to matrix inversion.

```
hide-const (open) unit
```

```
end
```