

# MathProject

1.2

Generated by Doxygen 1.8.17



<b>1 File Index</b>	<b>1</b>
1.1 File List	1
<b>2 File Documentation</b>	<b>3</b>
2.1 include/SqEq.hh File Reference	3
2.1.1 Macro Definition Documentation	4
2.1.1.1 SSEQ_ERROR	4
2.1.1.2 MAX_ROOT_NUM	4
2.1.1.3 COEFF_NUM	4
2.1.1.4 EPS	4
2.1.2 Enumeration Type Documentation	4
2.1.2.1 ResType	4
2.1.3 Function Documentation	5
2.1.3.1 read_coeffs()	5
2.1.3.2 print_res()	5
2.1.3.3 solve_sseq()	6
2.1.3.4 solve_linear()	6
2.1.3.5 is_equal()	7
2.1.3.6 ret_code()	7
2.1.3.7 unit_testing()	8
2.2 SqEq.hh	8
2.3 lib/SqEq.cc File Reference	9
2.3.1 Detailed Description	10
2.3.2 Function Documentation	10
2.3.2.1 read_coeffs()	10
2.3.2.2 solve_sseq()	11
2.3.2.3 solve_linear()	11
2.3.2.4 is_equal()	12
2.3.2.5 print_res()	12
2.3.2.6 ret_code()	12
2.4 SqEq.cc	13
<b>Index</b>	<b>15</b>



# Chapter 1

## File Index

### 1.1 File List

Here is a list of all files with brief descriptions:

<a href="#">include/SqEq.hh</a> . . . . .	3
<a href="#">lib/SqEq.cc</a>	
File with most important SqEq functions . . . . .	9



## Chapter 2

# File Documentation

### 2.1 include/SqEq.hh File Reference

#### Macros

- #define `SSEQ_ERROR` 1  
*return this from main on error*
- #define `MAX_ROOT_NUM` 2  
*maximal munber of roots*
- #define `COEFF_NUM` 3  
*number of coefficients*
- #define `EPS` 1e-12  
*epsilon*

#### Enumerations

- enum `ResType` {  
    `RT_ERROR`, `RT_INV_COEFF_ERROR`, `RT_NULLPTR_ERROR`, `RT_VALID`,  
    `RT_NO_ROOTS`, `RT_ONE_ROOT`, `RT_TWO_ROOTS`, `RT_INF_ROOTS` }  
*enum with result types of solve functions*

#### Functions

- bool `read_coeffs` (const char \*prompt, double coeffs[`COEFF_NUM`])  
*Reading 3 coefficients from stdin.*
- void `print_res` (int res\_type, const double results[`MAX_ROOT_NUM`])  
*Print squaree equations roots.*
- int `solve_sseq` (const double coeffs[`COEFF_NUM`], double results[`MAX_ROOT_NUM`])  
*Solving square equation.*
- int `solve_linear` (double b, double c, double \*x\_ptr)  
*Solving linear equation.*
- int `is_equal` (double n1, double n2)  
*Compares two double numbers.*
- int `ret_code` (int res\_type)  
*Generate program return code.*
- bool `unit_testing` ()  
*Testing SqEq functions.*

## 2.1.1 Macro Definition Documentation

### 2.1.1.1 SSEQ\_ERROR

```
#define SSEQ_ERROR 1
```

return this from main on error

Definition at line 8 of file [SqEq.hh](#).

### 2.1.1.2 MAX\_ROOT\_NUM

```
#define MAX_ROOT_NUM 2
```

maximal munber of roots

Definition at line 13 of file [SqEq.hh](#).

### 2.1.1.3 COEFF\_NUM

```
#define COEFF_NUM 3
```

number of coefficients

Definition at line 18 of file [SqEq.hh](#).

### 2.1.1.4 EPS

```
#define EPS 1e-12
```

epsilon

Definition at line 23 of file [SqEq.hh](#).

## 2.1.2 Enumeration Type Documentation

### 2.1.2.1 ResType

```
enum ResType
```

enum with result types of solve functions



## Enumerator

RT_ERROR	
RT_INV_COEFF_ERROR	
RT_NULLPTR_ERROR	
RT_VALID	
RT_NO_ROOTS	
RT_ONE_ROOT	
RT_TWO_ROOTS	
RT_INF_ROOTS	

Definition at line 28 of file [SqEq.hh](#).

## 2.1.3 Function Documentation

### 2.1.3.1 read\_coeffs()

```
bool read_coeffs (
    const char * prompt,
    double coeffs[COEFF_NUM] )
```

Reading 3 coefficients from stdin.

## Parameters

in	<i>prompt</i>	message to user
out	<i>coeffs</i>	array for coeffs, length must be $\geq 3$

## Returns

true if all OK  
false on error

Definition at line 13 of file [SqEq.cc](#).

References [COEFF\\_NUM](#).

### 2.1.3.2 print\_res()

```
void print_res (
    int res_type,
    const double results[MAX_ROOT_NUM] )
```

Print squaree equations roots.

**Parameters**

in	<i>res_type</i>	defines error/number of roots
in	<i>results</i>	roots

Definition at line 92 of file [SqEq.cc](#).

References [RT\\_ERROR](#), [RT\\_INF\\_ROOTS](#), [RT\\_INV\\_COEFF\\_ERROR](#), [RT\\_NO\\_ROOTS](#), [RT\\_NULLPTR\\_ERROR](#), [RT\\_ONE\\_ROOT](#), and [RT\\_TWO\\_ROOTS](#).

**2.1.3.3 solve\_sqeq()**

```
int solve_sqeq (
    const double coeffs[COEFF_NUM],
    double results[MAX_ROOT_NUM] )
```

Solving square equation.

**Parameters**

in	<i>coeffs</i>	array with coefficients, length must must be $\geq 3$
out	<i>results</i>	array with roots, length must must be $\geq 2$

**Returns**

result type

Definition at line 20 of file [SqEq.cc](#).

References [is\\_equal\(\)](#), [RT\\_ERROR](#), [RT\\_INV\\_COEFF\\_ERROR](#), [RT\\_NO\\_ROOTS](#), [RT\\_NULLPTR\\_ERROR](#), [RT\\_ONE\\_ROOT](#), [RT\\_TWO\\_ROOTS](#), and [solve\\_linear\(\)](#).

**2.1.3.4 solve\_linear()**

```
int solve_linear (
    double b,
    double c,
    double * x_ptr )
```

Solving linear equation.

**Parameters**

in	<i>b</i>	coeff on x
in	<i>c</i>	free member
out	<i>x_ptr</i>	result pointer

**Returns**

result type

Definition at line 71 of file [SqEq.cc](#).

References [is\\_equal\(\)](#), [RT\\_INF\\_ROOTS](#), [RT\\_NO\\_ROOTS](#), and [RT\\_ONE\\_ROOT](#).

Referenced by [solve\\_sseq\(\)](#).

**2.1.3.5 is\_equal()**

```
int is_equal (
    double n1,
    double n2 )
```

Compares two double numbers.

**Parameters**

in	<i>n1</i>	1st num
in	<i>n2</i>	2nd num

**Returns**

int

Definition at line 87 of file [SqEq.cc](#).

References [EPS](#).

Referenced by [solve\\_linear\(\)](#), and [solve\\_sseq\(\)](#).

**2.1.3.6 ret\_code()**

```
int ret_code (
    int res_type )
```

Generate program return code.

**Parameters**

in	<i>res_type</i>	value returned by solve function
----	-----------------	----------------------------------

**Returns**

int

Definition at line 124 of file [SqEq.cc](#).

References [RT\\_VALID](#), and [SSEQ\\_ERROR](#).

**2.1.3.7 unit\_testing()**

```
bool unit_testing ( )
```

Testing SqEq functions.

**Returns**

true if all tests passed

false if tests not passed

**2.2 SqEq.hh**

```
00001 /**
00002  * @brief return this from main on error
00003  */
00004
00005 #ifndef SSEQ
00006 #define SSEQ
00007
00008 #define SSEQ_ERROR 1
00009
00010 /**
00011  * @brief maximal munber of roots
00012  */
00013 #define MAX_ROOT_NUM 2
00014
00015 /**
00016  * @brief number of coefficients
00017  */
00018 #define COEFF_NUM 3
00019
00020 /**
00021  * @brief epsilon
00022  */
00023 #define EPS 1e-12
00024
00025 /**
00026  * @brief enum with result types of solve functions
00027  */
00028 enum ResType
00029 {
00030     RT_ERROR, // Errors from here
00031     RT_INV_COEFF_ERROR,
00032     RT_NULLPTR_ERROR,
00033
00034     RT_VALID, // Valid codes from here
00035     RT_NO_ROOTS,
00036     RT_ONE_ROOT,
00037     RT_TWO_ROOTS,
00038     RT_INF_ROOTS
00039 };
00040
00041 /**
00042  * @brief Reading 3 coefficients from stdin
00043  *
00044  * @param[in] prompt message to user
00045  * @param[out] coeffs array for coeffs, length must be >= 3
00046  * @return true if all OK
00047  * @return false on error
00048  */
```

```

00049 bool read_coeffs(const char *prompt, double coeffs[COEFF_NUM]);
00050
00051 /**
00052  * @brief Print square equations roots
00053  *
00054  * @param[in] res_type defines error/number of roots
00055  * @param[in] results roots
00056  */
00057 void print_res(int res_type, const double results[MAX_ROOT_NUM]);
00058
00059 /**
00060  * @brief Solving square equation
00061  *
00062  * @param[in] coeffs array with coefficients, length must be >= 3
00063  * @param[out] results array with roots, length must be >= 2
00064  * @return result type
00065  */
00066 int solve_sseq(const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM]);
00067
00068 /**
00069  * @brief Solving linear equation
00070  *
00071  * @param[in] b coeff on x
00072  * @param[in] c free member
00073  * @param[out] x_ptr result pointer
00074  * @return result type
00075  */
00076 int solve_linear(double b, double c, double *x_ptr);
00077
00078 /**
00079  * @brief Compares two double numbers
00080  *
00081  * @param[in] n1 1st num
00082  * @param[in] n2 2nd num
00083  * @return int
00084  */
00085 int is_equal(double n1, double n2);
00086
00087 /**
00088  * @brief Generate program return code
00089  *
00090  * @param[in] res_type value returned by solve function
00091  * @return int
00092  */
00093 int ret_code(int res_type);
00094
00095 /**
00096  * @brief Testing SqEq functions
00097  *
00098  * @return true if all tests passed
00099  * @return false if tests not passed
00100  */
00101 bool unit_testing();
00102
00103
00104 #endif // SQEQ

```

## 2.3 lib/SqEq.cc File Reference

File with most important SqEq functions.

```

#include <assert.h>
#include <math.h>
#include <stdio.h>
#include "SqEq.hh"

```

### Functions

- bool `read_coeffs` (const char \*prompt, double coeffs[COEFF\_NUM])  
*Reading 3 coefficients from stdin.*
- int `solve_sseq` (const double coeffs[COEFF\_NUM], double results[MAX\_ROOT\_NUM])

*Solving square equation.*

- int [solve\\_linear](#) (double b, double c, double \*x\_ptr)

*Solving linear equation.*

- int [is\\_equal](#) (double n1, double n2)

*Compares two double numbers.*

- void [print\\_res](#) (int res\_type, const double results[[MAX\\_ROOT\\_NUM](#)])

*Print square equations roots.*

- int [ret\\_code](#) (int res\_type)

*Generate program return code.*

### 2.3.1 Detailed Description

File with most important SqEq functions.

Author

Tako

Definition in file [SqEq.cc](#).

### 2.3.2 Function Documentation

#### 2.3.2.1 read\_coefs()

```
bool read_coefs (
    const char * prompt,
    double coefs[COEFF\_NUM] )
```

Reading 3 coefficients from stdin.

Parameters

in	<i>prompt</i>	message to user
out	<i>coefs</i>	array for coefs, length must be $\geq 3$

Returns

true if all OK  
false on error

Definition at line 13 of file [SqEq.cc](#).

References [COEFF\\_NUM](#).

### 2.3.2.2 solve\_sqeq()

```
int solve_sqeq (
    const double coeffs[COEFF_NUM],
    double results[MAX_ROOT_NUM] )
```

Solving square equation.

#### Parameters

in	<i>coeffs</i>	array with coefficients, length must must be $\geq 3$
out	<i>results</i>	array with roots, length must must be $\geq 2$

#### Returns

result type

Definition at line 20 of file [SqEq.cc](#).

References [is\\_equal\(\)](#), [RT\\_ERROR](#), [RT\\_INV\\_COEFF\\_ERROR](#), [RT\\_NO\\_ROOTS](#), [RT\\_NULLPTR\\_ERROR](#), [RT\\_ONE\\_ROOT](#), [RT\\_TWO\\_ROOTS](#), and [solve\\_linear\(\)](#).

### 2.3.2.3 solve\_linear()

```
int solve_linear (
    double b,
    double c,
    double * x_ptr )
```

Solving linear equation.

#### Parameters

in	<i>b</i>	coeff on x
in	<i>c</i>	free member
out	<i>x_ptr</i>	result pointer

#### Returns

result type

Definition at line 71 of file [SqEq.cc](#).

References [is\\_equal\(\)](#), [RT\\_INF\\_ROOTS](#), [RT\\_NO\\_ROOTS](#), and [RT\\_ONE\\_ROOT](#).

Referenced by [solve\\_sqeq\(\)](#).

### 2.3.2.4 is\_equal()

```
int is_equal (
    double n1,
    double n2 )
```

Compares two double numbers.

#### Parameters

in	<i>n1</i>	1st num
in	<i>n2</i>	2nd num

#### Returns

int

Definition at line 87 of file [SqEq.cc](#).

References [EPS](#).

Referenced by [solve\\_linear\(\)](#), and [solve\\_sseq\(\)](#).

### 2.3.2.5 print\_res()

```
void print_res (
    int res_type,
    const double results[MAX_ROOT_NUM] )
```

Print squaree equations roots.

#### Parameters

in	<i>res_type</i>	defines error/number of roots
in	<i>results</i>	roots

Definition at line 92 of file [SqEq.cc](#).

References [RT\\_ERROR](#), [RT\\_INF\\_ROOTS](#), [RT\\_INV\\_COEFF\\_ERROR](#), [RT\\_NO\\_ROOTS](#), [RT\\_NULLPTR\\_ERROR](#), [RT\\_ONE\\_ROOT](#), and [RT\\_TWO\\_ROOTS](#).

### 2.3.2.6 ret\_code()

```
int ret_code (
    int res_type )
```

Generate program return code.



## Parameters

in	<i>res_type</i>	value returned by solve function
----	-----------------	----------------------------------

## Returns

int

Definition at line 124 of file SqEq.cc.

References [RT\\_VALID](#), and [SSEQ\\_ERROR](#).

## 2.4 SqEq.cc

```

00001 /**
00002  * @file SqEq.cc
00003  * @author Tako
00004  * @brief File with most important SqEq functions
00005  */
00006
00007 #include <assert.h>
00008 #include <math.h>
00009 #include <stdio.h>
00010
00011 #include "SqEq.hh"
00012
00013 bool read_coeffs(const char *prompt, double coeffs[COEFF_NUM])
00014 {
00015     puts(prompt);
00016     int scanned = scanf("%lf %lf %lf", coeffs, coeffs + 1, coeffs + 2);
00017     return COEFF_NUM == scanned;
00018 }
00019
00020 int solve_sseq(const double coeffs[COEFF_NUM], double results[MAX_ROOT_NUM])
00021 {
00022     if ((nullptr == coeffs) || (nullptr == results))
00023         return RT_NULLPTR_ERROR;
00024
00025     double a = coeffs[0], b = coeffs[1], c = coeffs[2];
00026     if (!(isfinite(a) && isfinite(b) && isfinite(c)))
00027         return RT_INV_COEFF_ERROR;
00028
00029     if (is_equal(a, 0))
00030         return (solve_linear(b, c, results));
00031     else if (is_equal(c, 0))
00032     {
00033         if (is_equal(b, 0))
00034         {
00035             results[0] = 0;
00036             results[1] = NAN;
00037             return RT_ONE_ROOT;
00038         }
00039         else
00040         {
00041             results[0] = 0;
00042             return (solve_linear(a, b, results + 1) + 1);
00043         }
00044     }
00045
00046     else
00047     {
00048         double d = b * b - 4 * a * c;
00049
00050         if (is_equal(d, 0))
00051         {
00052             results[0] = -b / (2 * a);
00053             results[1] = NAN;
00054             return RT_ONE_ROOT;
00055         }
00056         else if (d < 0)
00057         {
00058             return RT_NO_ROOTS;
00059         }
00060         else if (d > 0)
00061         {

```

```

00062     double sqrt_d = sqrt(d);
00063     results[0] = (-b + sqrt_d) / (2 * a);
00064     results[1] = (-b - sqrt_d) / (2 * a);
00065     return RT_TWO_ROOTS;
00066 }
00067 }
00068 return RT_ERROR;
00069 }
00070
00071 int solve_linear(double b, double c, double *x_ptr)
00072 {
00073     if (is_equal(b, 0))
00074         return is_equal(c, 0) ? RT_INF_ROOTS : RT_NO_ROOTS;
00075     else if (is_equal(c, 0))
00076     {
00077         *x_ptr = 0;
00078         return RT_ONE_ROOT;
00079     }
00080     else
00081     {
00082         *x_ptr = -c / b;
00083         return RT_ONE_ROOT;
00084     }
00085 }
00086
00087 int is_equal(double n1, double n2)
00088 {
00089     return (fabs(n1 - n2) < EPS);
00090 }
00091
00092 void print_res(int res_type, const double results[MAX_ROOT_NUM])
00093 {
00094     printf("Answer:\n");
00095     switch (res_type)
00096     {
00097     case RT_NO_ROOTS:
00098         printf("No roots\n");
00099         break;
00100     case RT_ONE_ROOT:
00101         printf("x = %lg\n", results[0]);
00102         break;
00103     case RT_TWO_ROOTS:
00104         printf("x1 = %lg\nx2 = %lg\n", results[0], results[1]);
00105         break;
00106     case RT_INF_ROOTS:
00107         printf("Infinite number of roots\n");
00108         break;
00109     case RT_INV_COEFF_ERROR:
00110         printf("INVALID COEFFICIENT VALUE\n");
00111         break;
00112     case RT_NULLPTR_ERROR:
00113         printf("COEFF OR ROOT ARRAY IS NULLPTR\n");
00114         break;
00115     case RT_ERROR:
00116         printf("ERROR\n");
00117         break;
00118     default:
00119         printf("UNKNOWN ERROR\n");
00120         break;
00121     }
00122 }
00123
00124 int ret_code(int res_type)
00125 {
00126     return (res_type >= RT_VALID) ? 0 : SEQO_ERROR;
00127 }

```

# Index

COEFF\_NUM  
SqEq.hh, 4

EPS  
SqEq.hh, 4

include/SqEq.hh, 3, 8

is\_equal  
SqEq.cc, 11  
SqEq.hh, 7

lib/SqEq.cc, 9, 13

MAX\_ROOT\_NUM  
SqEq.hh, 4

print\_res  
SqEq.cc, 12  
SqEq.hh, 5

read\_coeffs  
SqEq.cc, 10  
SqEq.hh, 5

ResType  
SqEq.hh, 4

ret\_code  
SqEq.cc, 12  
SqEq.hh, 7

RT\_ERROR  
SqEq.hh, 5

RT\_INF\_ROOTS  
SqEq.hh, 5

RT\_INV\_COEFF\_ERROR  
SqEq.hh, 5

RT\_NO\_ROOTS  
SqEq.hh, 5

RT\_NULLPTR\_ERROR  
SqEq.hh, 5

RT\_ONE\_ROOT  
SqEq.hh, 5

RT\_TWO\_ROOTS  
SqEq.hh, 5

RT\_VALID  
SqEq.hh, 5

solve\_linear  
SqEq.cc, 11  
SqEq.hh, 6

solve\_sseq  
SqEq.cc, 10  
SqEq.hh, 6

SqEq.cc  
is\_equal, 11  
print\_res, 12  
read\_coeffs, 10  
ret\_code, 12  
solve\_linear, 11  
solve\_sseq, 10

SqEq.hh  
COEFF\_NUM, 4  
EPS, 4  
is\_equal, 7  
MAX\_ROOT\_NUM, 4  
print\_res, 5  
read\_coeffs, 5  
ResType, 4  
ret\_code, 7  
RT\_ERROR, 5  
RT\_INF\_ROOTS, 5  
RT\_INV\_COEFF\_ERROR, 5  
RT\_NO\_ROOTS, 5  
RT\_NULLPTR\_ERROR, 5  
RT\_ONE\_ROOT, 5  
RT\_TWO\_ROOTS, 5  
RT\_VALID, 5  
solve\_linear, 6  
solve\_sseq, 6  
SSEQ\_ERROR, 4  
unit\_testing, 8

SSEQ\_ERROR  
SqEq.hh, 4

unit\_testing  
SqEq.hh, 8