编程求解
一元四次方程

Hanford

2016 年 09 月 27 日

目 录

第 1 章　简介

1.1　项目功能

本项目用来求解一元一次至四次方程。具体包括：

1、将求解一元四次方程的算法封装至动态库 SolveEquationDll.dll 内，供 VB/Excel/C#等语言调用；

2、将求解一元四次方程的算法封装至 COM 组件 SolveEquationCom.dll 内，供脚本语言（vbs 或 js）调用；

3、将求解一元四次方程的算法封装至.NET 组件 SolveEquationNET.dll 内，供.NET 语言（VB.NET/C#……）调用；

4、使用 VC 编写本机程序，实现一元四次方程求解；

5、使用 Excel 调用动态库 SolveEquationDll.dll，实现一元四次方程求解；

6、使用脚本语言（vbs 或 js）编写 html 网页，调用 COM 组件 SolveEquationCom.dll，实现一元四次方程求解；

7、使用 c#编写本机程序，实现一元四次方程求解；

8、使用 vb 编写本机程序，实现一元四次方程求解。

1.2　核心算法

核心算法请参考 doc 目录下的两个文档：

《一元四次方程-16.04.05.pdf》　　公式(28)至(35)用来求解一元四次方程

《一元三次方程-16.04.06.pdf》　　公式(29)至(34)用来求解一元三次方程

第 2 章　VC

2.1　dllSDK

vc\dllSDK 存放着 dllSDK 项目。它是一个基础模块，封装了核心算法。

2.1.1　目录结构

目录结构请见下表：

文件夹	说　　明
bin	存放编译结果文件（SolveEquationDll.dll、SolveEquationDll.lib……）。vc2010-x64-RU 表示编译器为 vc2010，平台为 x64，RU 表示 Release Unicode
inc	本模块暴露给客户程序的接口头文件，用来声明接口函数 客户程序要动态连接本模块，请包含 _Inc.h 客户程序要静态连接本模块，请包含 _Static.h
make-dll	VC 项目文件，编译生成动态库文件 SolveEquationDll.dll 及其导入库文件 SolveEquationDll.lib
make-libD	VC 项目文件，编译生成静态库文件 SolveEquationDllD.lib 使用的 C 运行时库文件是“多线程 DLL”版本
make-libT	VC 项目文件，编译生成静态库文件 SolveEquationDllT.lib 使用的 C 运行时库文件是“多线程”版本
make-libS	VC 项目文件，编译生成静态库文件 SolveEquationDllS.lib 使用的 C 运行时库文件是“单线程”版本
src	存放源代码

2.1.2　接口函数

接口函数在文件 inc\Solve.h 里，其定义如下：

```
SolveEquationDllI  int  __stdcall  SolveEquation(const  double  z[],double  x[]);
```

宏 SolveEquationDllI 的定义如下：

编译生成动态库时，宏 SolveEquationDllI 被定义为__declspec(dllexport)；

编译生成静态库时，宏 SolveEquationDllI 被定义为空；

客户程序包含 inc_Inc.h 后，宏 SolveEquationDllI 被定义为__declspec(dllimport)，即导入本模块的接口函数；

客户程序包含 inc_Static.h 后，宏 SolveEquationDllI 被定义为空，即静态连接本模块的接口函数。

2.1.3　编译

假如使用 vc2015 编译生成动态库，请运行 Visual Studio 2015，打开 make-dll\vc2015\SolveEquationDll.sln。编译即可生成 SolveEquationDll.dll 和 SolveEquationDll.lib。

假如使用 vc2015 编译生成静态库（C 运行时库文件是“多线程”版本），请运行 Visual Studio 2015，打开 make-libT\vc2015\SolveEquationDllT.sln。编译即可生成 SolveEquationDllD.lib。

假如使用 vc2015 编译生成静态库（C 运行时库文件是“多线程 DLL”版本），请运行 Visual Studio 2015，打开 make-libD\vc2015\SolveEquationDllD.sln。编译即可生成 SolveEquationDllT.lib。

请编译生成如上四个文件：SolveEquationDll.dll、SolveEquationDll.lib、SolveEquationDllD.lib、SolveEquationDllT.lib。后面的模块或程序会用到它们。

2.2　dllATL

vc\dllATL 存放着 dllATL 项目。它使用 dllSDK 模块，将核心算法封装为 COM 组件。

2.2.1　目录结构

目录结构请见下表：

文件夹	说　　明
bin	存放编译结果文件（SolveEquationCom.dll）
make	VC 项目文件
src	存放源代码

2.2.2　编译

1、设置 SolveEquationCom.idl，如下图所示：

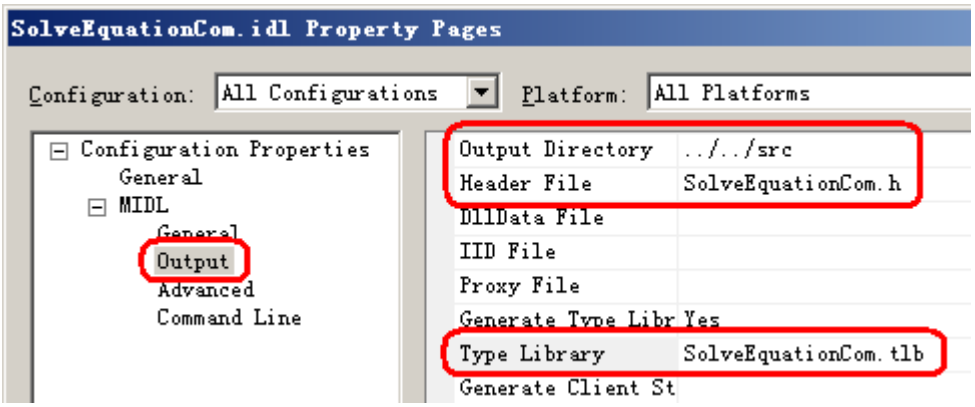

图 2.1

编译时，MIDL.exe 将根据 SolveEquationCom.idl 生成 SolveEquationCom.h（C/C++访问 COM 组件的头文件）和 SolveEquationCom.tlb（COM 组件的类型库文件）。生成的文件将在目录../../src 内。

2、设置 SolveEquationCom 的 C 运行时库

RA、RU 请设置为“多线程”；DA、DU 请设置为“多线程 Debug”。如下图所示：

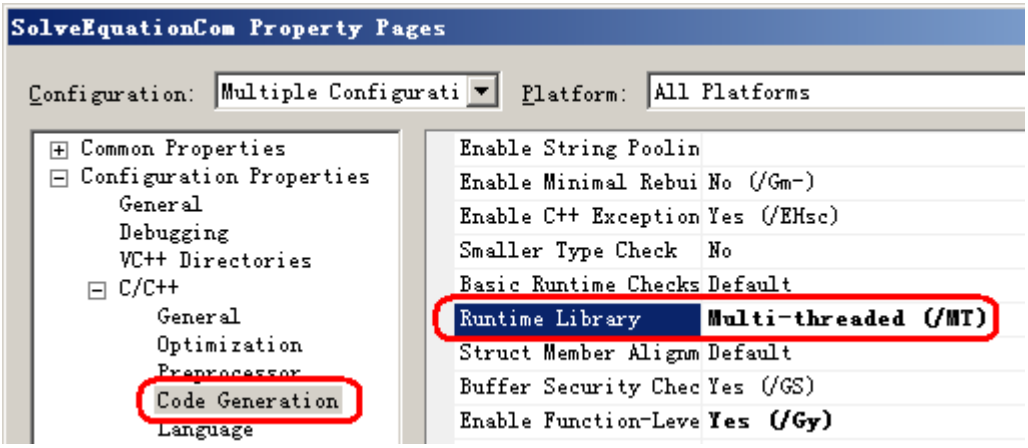

图 2.2

2.2.3　注册 COM 组件

注册 32 位 COM 组件，可进入 vc\dllATL\bin\vc6-win32-RA 目录，运行 reg.bat 即可。

注册 64 位 COM 组件，可进入 vc\dllATL\bin\vc2010-x64-RU 目录，运行 reg.bat 即可。

reg.bat 的内容如下图所示：

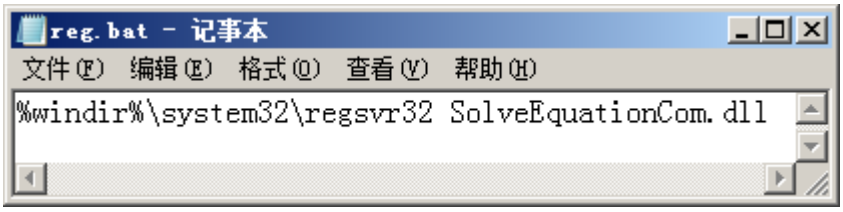

图 2.3

regsvr32.exe 的执行原理是：首先载入 COM 组件（SolveEquationCom.dll），然后运行 COM 组件的导出函数 DllRegisterServer。

在 64 位操作系统上%windir%\system32\regsvr32.exe 是 64 位的，它是无法载入 32 位 COM 组件的。那为什么上图所示的命令仍能正常执行呢？原因在于：64 位的 regsvr32.exe 会判断 COM 组件是 32 位的还是 64 位的。64 位的就载入 COM 组件、调用 DllRegisterServer 函数；32 位的就启动 32 位的 regsvr32.exe 程序（在%windir%\SysWOW64 目录内），由它负责载入 32 位的 COM 组件、调用 DllRegisterServer 函数。下图显示 32 位的 SolveEquationCom.dll 注册成功，任务管理器里可以看到有两个 regsvr32.exe 进程，一个是 32 位的一个是 64 位的。

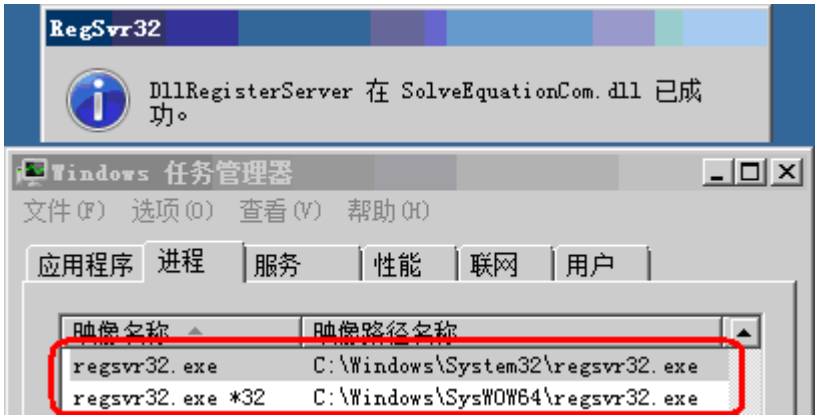

图 2.4

2.2.4　代码说明

CEquation::Solve 是本模块的核心函数，它调用了 dllSDK 模块的接口函数

SolveEquation，完成核心计算。

CEquation::Solve 大量的工作是参数转换：它调用 CEquation::GetZ(VARIANT *z,double dz[10])将客户程序传递过来的参数 VARIANT*z 转换为 SolveEquation(const double z[],double x[]);所需的第一个参数。分别考虑了 double 数组（VB6/VBA 传过来的）、VARIANT 数组（VBS 传来的）、数组 COM 对象（JS 传来的）、字符串这几种情况。具体请参考代码。

2.3　dllNET

本项目用来将 dllSDK 静态库进一步封装为.NET 组件。

编译器请选择 vc2010~2015。如：使用 vc2010 打开 vc\dllNET\make\vc2010\SolveEquationNET.sln。然后打开/clr 编译开关，如下图所示：

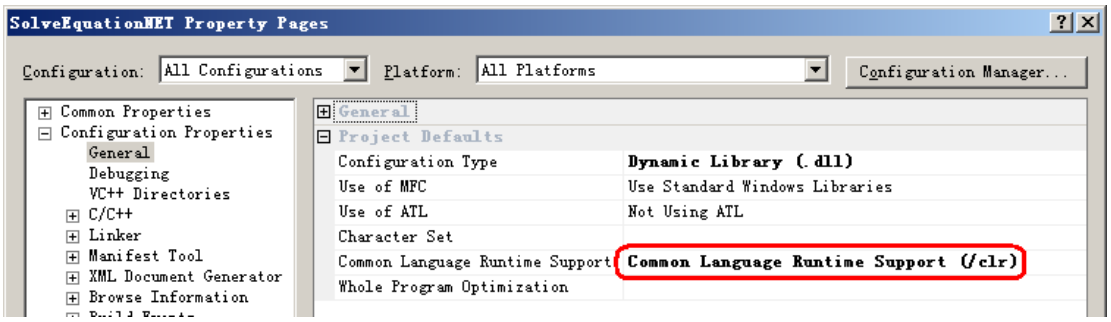

图 2.5

vc 编译生成的.NET 组件，必须动态连接 C 函数库。这就意味着发布程序时必须带上 MSCR100.dll、MSVCP100.dll 这些 C 函数库。另外，更致命的缺陷是：它分为 32 位、64 位版本。c#程序引用这样的.NET 组件，就要求平台必须为 32 位或 64 位的，失去了自适应能力（即在 32 位操作系统上以 32 位运行，在 64 位操作系统上以 64 位运行）。

解决上述缺陷的办法就是使用 c#来生成 dllNET。

2.4　exeMFC

exeMFC 是使用 MFC 编写的一个执行程序，它调用 dllSDK 模块，完成一元四次方程的计算。如下图所示：

图 2.6

2.5　exeSDK

exeSDK 是使用 Win32 SDK 编写的一个执行程序，它调用 dllSDK 模块，完成一元四次方程的计算。如上图所示。

2.6　exeATL

exeATL是使用ATL编写的一个执行程序，它调用dllSDK模块，完成一元四次方程的计算。如图 2.6所示。

2.7　exeNET

exeNET是一个使用.NET托管代码的VC++程序。它调用dllSDK模块，完成一元四次方程的计算。如图 2.6所示。

编译器请选择 vc2008~2015。vc2012~vc2015 需要进行如下配置：

1、打开/clr 编译开关，如下图所示：

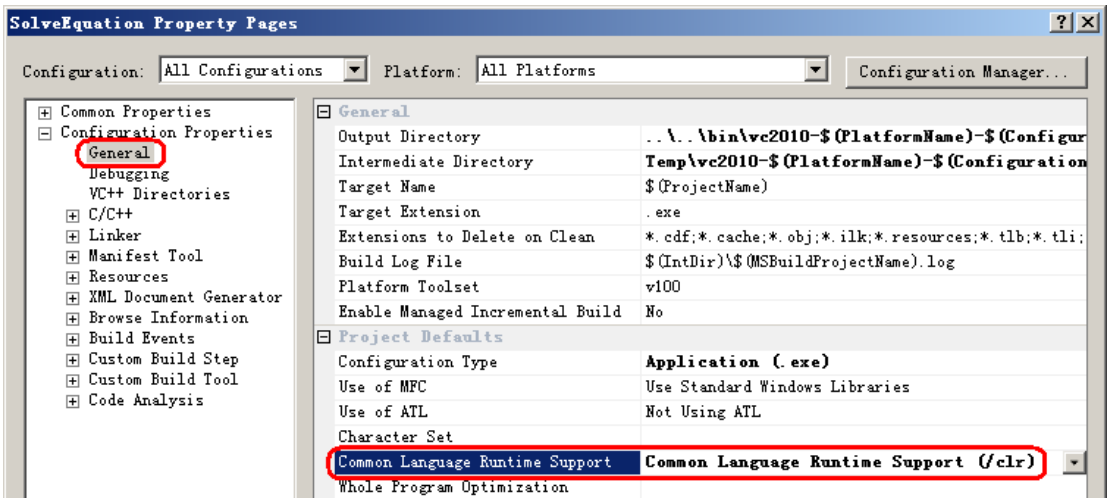

图 2.7

2、选择 C 运行时库

RA、RU 请设置为“多线程 DLL”；DA、DU 请设置为“多线程 Debug DLL”。如下图所示：

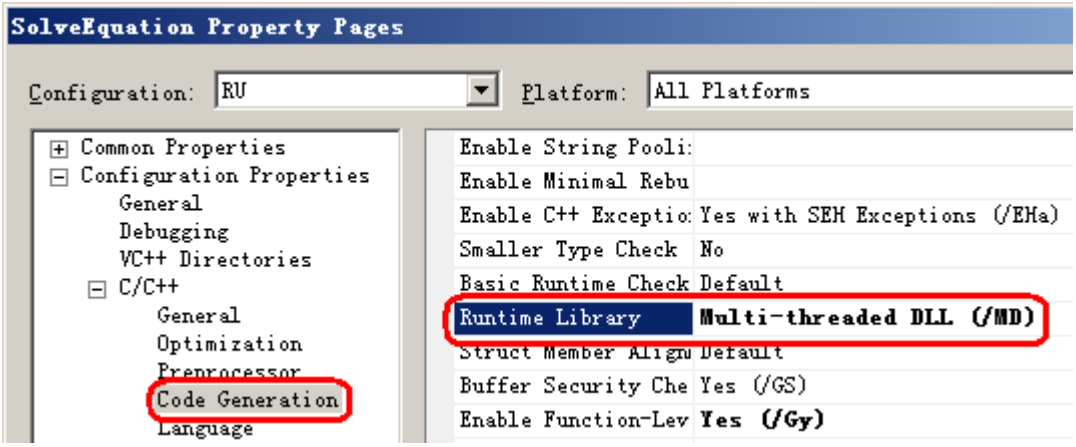

图 2.8

3、移除*.resx 文件，如下图所示：

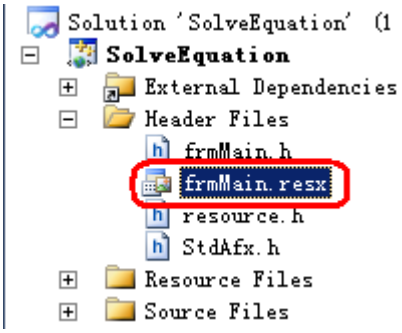

图 2.9

4、修改 frmMain.h 的类型为 C++ Form，如下图所示：

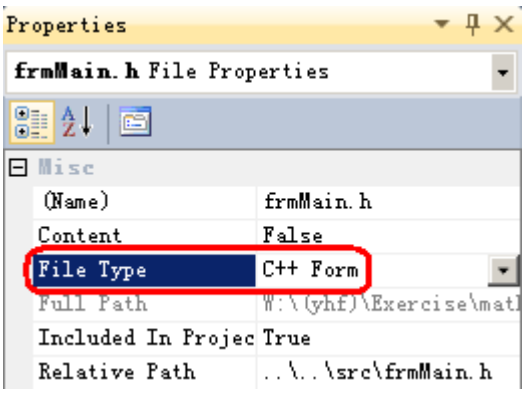

图 2.10

5、Visual Studio 里单击【File】【Close Solution】菜单项，然后使用记事本修改文件 SolveEquation.vcxproj。具体修改如下：

1）指定.NET Framework 版本

增加下面的蓝色部分，指定.NET Framework 版本为 4.0

```
<PropertyGroup Label="Globals">
  ...  ...  ...
  <TargetFrameworkVersion>v4.0</TargetFrameworkVersion>
  <Keyword>Win32Proj</Keyword>
  ...  ...  ...
</PropertyGroup>
```

2）增加.NET 引用

增加下面的内容

```
<ItemGroup>
  <Reference Include="System" />
  <Reference Include="System.Data" />
```

```
    <Reference  Include="System.Drawing"  />
    <Reference  Include="System.Windows.Forms"  />
    <Reference  Include="System.Xml"  />
</ItemGroup>
```

3）修改.resx 文件的依赖项

下面的蓝色部分，修改前是 frmMain.h。事实上..\..\src\frmMain.resx 依赖于..\..\src\frmMain.h，所以应该改过来。这个应该是 Visual Studio 的一个 BUG：frmMain.h、frmMain.resx 必须与 vcxproj 在同一文件夹内，否则就会出错。

```
<ItemGroup>
    <EmbeddedResource  Include="..\..\src\frmMain.resx">
        <DependentUpon>..\..\src\frmMain.h</DependentUpon>
    </EmbeddedResource>
</ItemGroup>
```

2.8　exeUseCom

exeNET通过调用COM组件（SolveEquationCom.dll），完成一元四次方程的计算。如图 2.6所示。

2.8.1　编译

设置*.c 文件（4.c、5.c、comMIDL.c）不使用预编译头文件，如下图所示

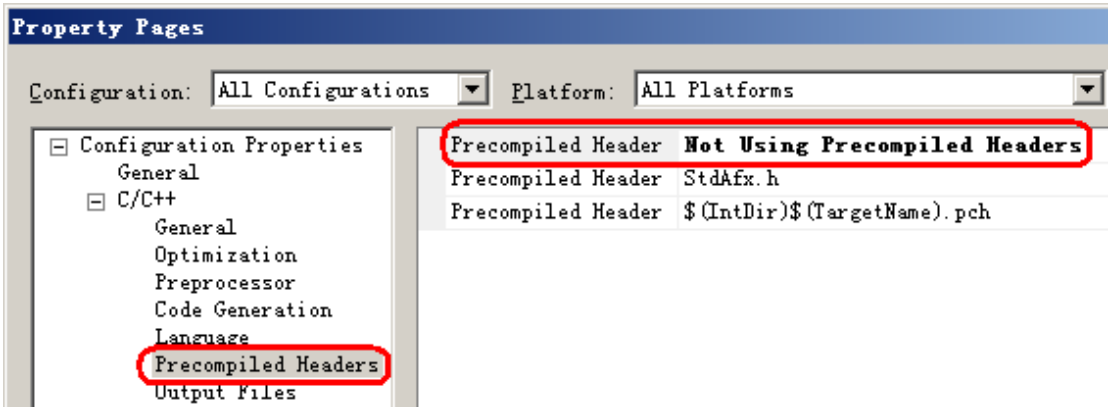

图 2.11

2.8.2　函数说明

详见下表

函数名	说　　明
Solve1MFC	使用 MFC 包装类访问 COM 组件 注意：C 语言无法使用 MFC 包装类
Solve2import	使用#import 语句访问 COM 组件 注意：C 语言无法使用#import
Solve3MIDL_cpp	编译 COM 组件时会生成 C/C++头文件 此函数使用 C++语言访问 COM 组件
Solve4MIDL_c	编译 COM 组件时会生成 C/C++头文件 此函数使用 C 语言访问 COM 组件
Solve5IDispatch	使用 C 语言，通过 IDispatch 访问 COM 组件
SolveEquation	求解一元四次方程时调用此函数，它随机的调用上述几个函数

第 3 章　Excel

3.1　执行

使用 Excel 打开 Excel\SolveEquation.xls，显示如下。输入方程系数，即可求出方程的根：

	A	B	C	D
1	求解方程：$ax^4+bx^3+cx^2+dx+e=0$			
2	系数	实部	虚部	
3	a	1		
4	b	-4		
5	c	6		
6	d	-4		
7	e	1		
8	根	实部	虚部	误差
9	x1	1.00000000000000	0.00000000000000	0.00E+00
10	x2	1.00000000000000	0.00000000000000	0.00E+00
11	x3	1.00000000000000	0.00000000000000	0.00E+00
12	x4	1.00000000000000	0.00000000000000	0.00E+00

图 3.1

3.2　宏的安全性

上一节的功能要正常运行，需要对 Excel 宏的安全性进行设置：

3.2.1　Excel 2003

单【工具】【宏】【安全性】菜单项，然后设置宏的安全性，如下图所示：

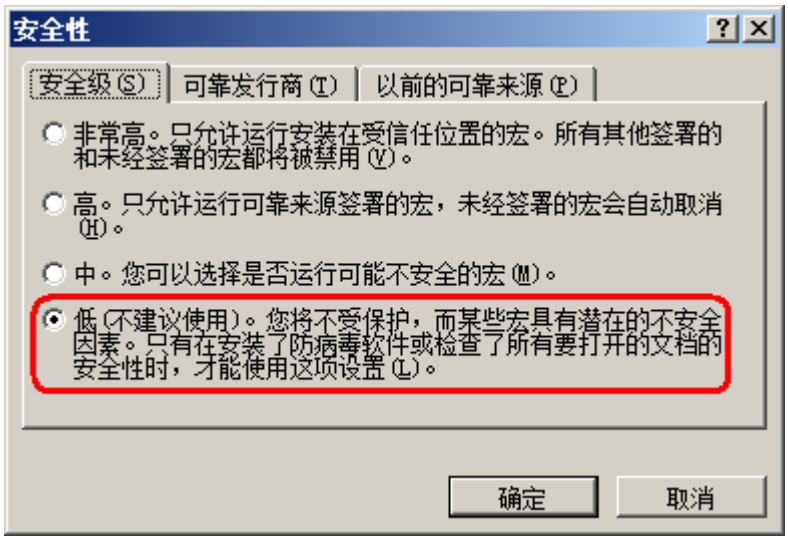

图 3.2

3.2.2　Excel 2007

单击左上角的按钮，然后单击下图的“Excel 选项”按钮

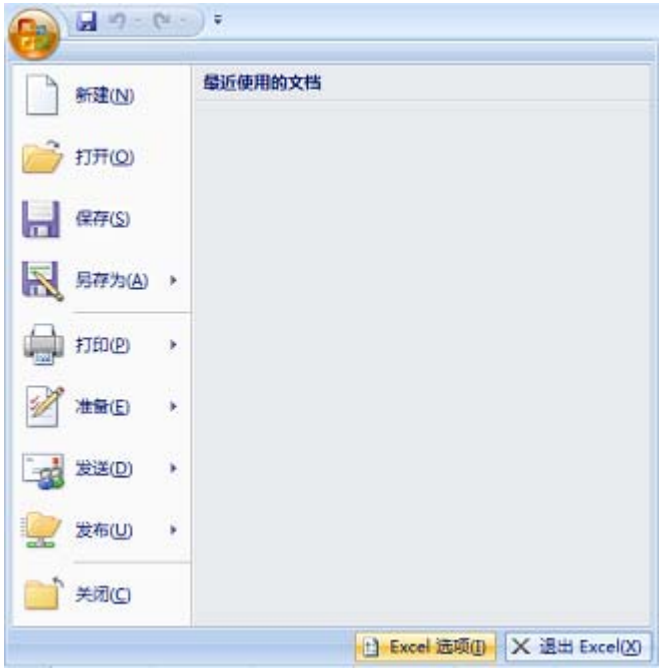

图 3.3

下图所示界面里，请勾中“在功能区显示"开发工具"选项卡”复选框。

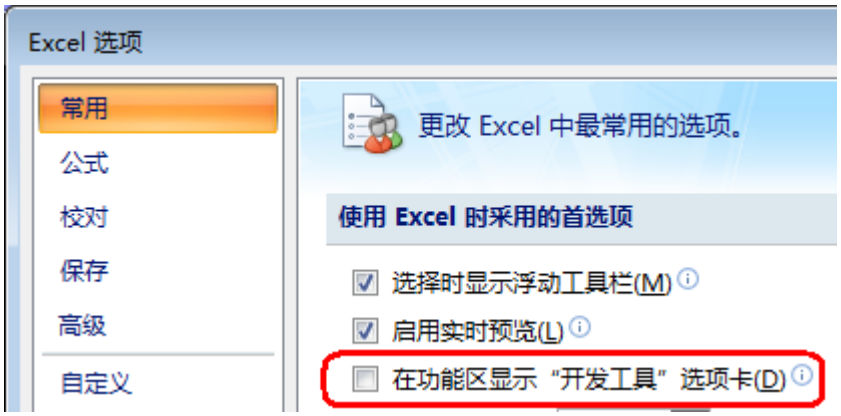

图 3.4

进入“开发工具”选项卡，然后单击“宏安全性”

图 3.5

显示如下界面。请选择“启用所有宏”。

图 3.6

3.2.3　Excel 2013

单击左上角的“文件”，然后单击左边的“选项”，显示下图所示的对话框：

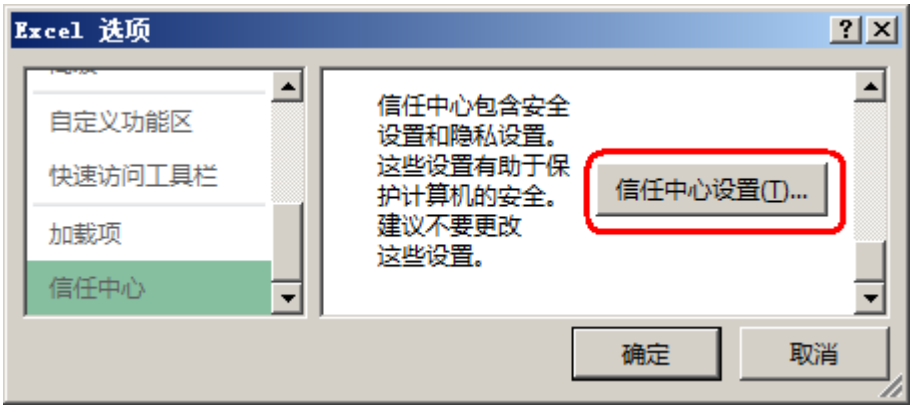

图 3.7

单击上图的“信任中心设置”按钮，显示如下界面

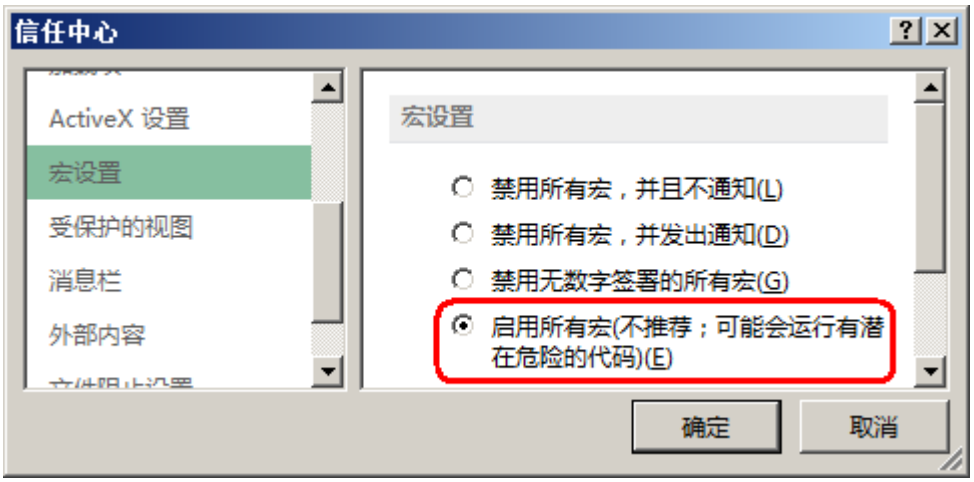

图 3.8

3.3　代码解析

Excel 打开 Excel\SolveEquation.xls 后，按下 Alt+F11 即可进入 VBA 代码编辑窗口。

单元格里输入一元四次方程系数后，将触发 Worksheet 的 Change 事件。函数 Worksheet_Change 将被调用。这个函数读取一元四次方程的系数后，调用函数 SolveEquationDll.dll 的导出函数 SolveEquation 进行方程求解，最后把根填入单元格内。

函数 SolveEquation 的声明有两个，如下所示：

```
#If Win64 Then
  '64 位 Office 下声明 DLL 里的函数
  Private Declare PtrSafe Function SolveEquation Lib "SolveEquationDll.dll" (ByRef z As Double, ByRef x As Double) As Long
#Else
  '32 位 Office 下声明 DLL 里的函数
  Private Declare Function SolveEquation Lib "SolveEquationDll.dll" (ByRef z As Double, ByRef x As Double) As Long
#End If
```

32 位的 SolveEquationDll.dll 在 xls 文件所在目录的 vc6-win32-RA 子目录内；64 位的 SolveEquationDll.dll 在 xls 文件所在目录的 vc2010-x64-RU 子目录内。调用函数 SolveEquation 之前，会调用 ChDrive ThisWorkbook.Path 和 ChDir ThisWorkbook.Path + "???"修改当前目录。这样，程序将根据 Excel 版本的不同，载入 32 位或 64 位的 SolveEquationDll.dll，然后再调用函数 SolveEquation。

第 4 章　html

4.1　注册 COM 组件

进入目录 html\vc6-win32-RA，运行 reg.bat 注册 32 位的 COM 组件 SolveEquationCom.dll；

64 位操作系统下，进入目录 html\vc2010-x64-RU，运行 reg.bat 注册 64 位的 COM 组件 SolveEquationCom.dll。

4.2　运行

运行 html\js.html（JavaScript 脚本调用 COM 组件 SolveEquationCom.dll）或 html\vbs.html（vbs 脚本调用 COM 组件 SolveEquationCom.dll），显示如下：

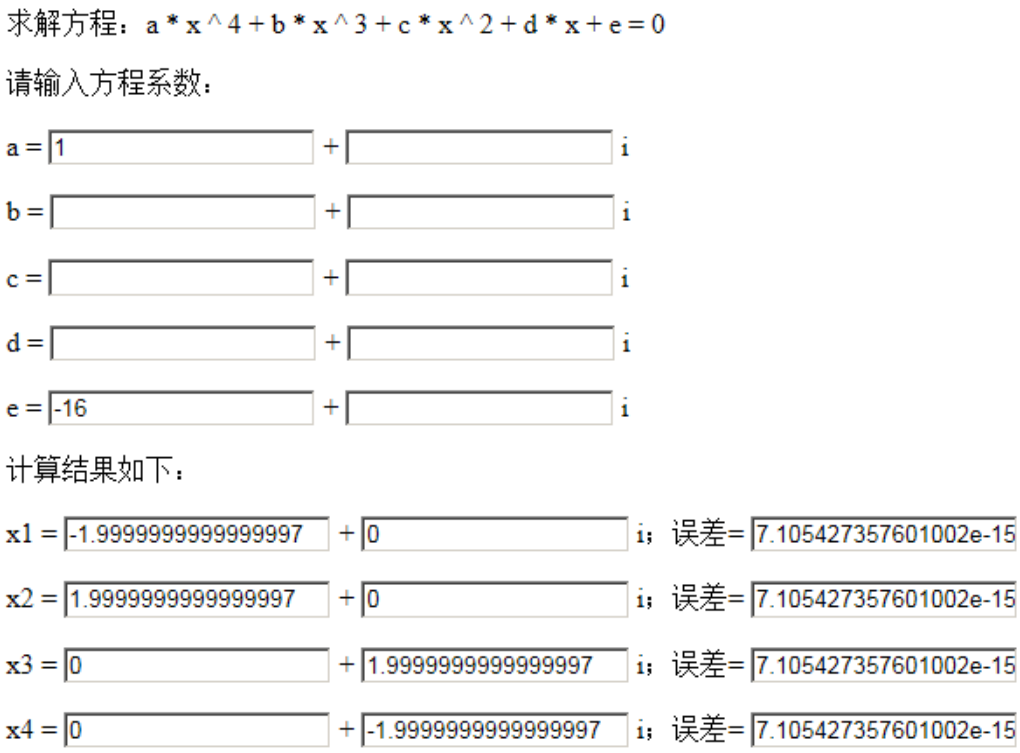

图 4.1

4.3　代码解析

以 JavaScript 脚本为例进行说明：

```
var obj = new ActiveXObject("SolveEquation"); //创建 COM 对象
var n = obj.Solve(z);                          //解方程，返回根的个数
obj.real(i);      //返回第 i 个根的实部，i 的范围 [0,n)
obj.imag(i);      //返回第 i 个根的虚部，i 的范围 [0,n)
obj.diff(i);      //返回第 i 个根的误差，i 的范围 [0,n)
```

第 5 章　c#

5.1　dllNET

dllNET 对 SolveEquationDll.dll 里的导出函数 SolveEquation 进行了封装，生成 SolveEquationNET.dll，便于.NET 语言调用。

SolveEquationDll.dll 分为 32 位、64 位，具体载入哪一个，代码里有两套思路。具体请参考代码。

5.2　exeWF

一个WinForm程序，完成一元四次方程的计算，如图 2.6所示。

求解一元四次方程时有两个选择：

```
#if  false
      double[]  x  =  SolveEquationNET.Equation.Solve(z);
#else
      double[]  x  =  Solve.SolveEquation(z);
#endif
```

SolveEquationNET.Equation.Solve 调用的是 SolveEquationNET.dll，后者又调用了 SolveEquationDll.dll。

Solve.SolveEquation 调用的是 c#代码，用到了 c#版的复数类 Complex。