传输过程数值模拟可视化编程开发

基于 HTML5 技术

王斌武　宋小鹏　吴国珊　著

北　京
冶 金 工 业 出 版 社
2018

前　言

由于一个项目的需要，笔者开始接触计算流体力学（CFD）模拟商业软件，经过一段时间的学习和使用，以为熟悉一个商业软件的操作也就理解计算流体力学。但实际上，仅仅会操作一款CFD软件，而不明白其所包含的原理与算法，离入门可能仍然有很远的距离。当笔者熟练操作商业软件，可对流体流动进行建模、设置、计算及分析时，也意识到庞大的商业软件程序包也可能对一个一维简单扩散传输方程束手无策，但一个数十行的C/C++代码却能完美解决问题。这样的差异源于对CFD理论知识的认知不足，笔者查阅了许多关于传输过程数值模拟程序开发计算的书籍，开始用C/C++编写一些简单的传输过程数值模拟程序。在辅导学生做关于数值模拟的毕业设计（论文）时，发现部分学生对编程存在着一种恐惧心理，他们反映程序的设计与编制工作很困难。而在笔者学习过程中，专门针对有限体积法的流动/传热计算程序设计的相关书籍少之又少，如果将最简单、最基础的流动/传热的程序呈现给读者，让读者在自行编写数值模拟程序时有所参考、有所比较，势必会对传输原理有更好的理解。

经过几年对传输过程数值模拟程序开发的浅显思考，笔者试图将传输过程数值模拟程序运行于浏览器端，使得执行简单数值模拟程序完全像打开一个网页一样简单，于是就有了本书的梗概。

传输过程包含热量、动量和质量传输三部分，其数值模拟涉及最多的是扩散方程和对流—扩散方程。本书主要介绍了传输过程仿真程序开发相关的入门知识，旨在编写完整的、便于阅读的

简单程序（包含前处理、计算和后处理）。其内容涵盖了 HTML5/JavaScript 程序开发、后处理之图形图像绘制、前处理之网格剖分、（非）线性方程组的求解、热传导过程温度场求解、相变过程传热计算、稳态不可压缩流体流动计算、三角单元有限元温度场计算理论等内容。

本书的章节是根据认知顺序以及难易程度由浅入深进行编排的，具体内容为：第 1 章概述传输过程数值模拟流程；第 2 章介绍本书所用的编程语言及后处理；第 3 章简要介绍网格剖分，网格是后续章节数值计算的基础；第 4、5 章分别介绍热量传输及简单动量传输过程程序开发；第 6 章介绍三角单元温度场有限元计算。本书由桂林航天工业学院王斌武、宋小鹏、吴国珊共同撰写，分工如下：王斌武撰写第 1~4 章；宋小鹏撰写第 5、6 章；其余部分由吴国珊撰写，全书由王斌武统稿。

在本书编写过程中，得到许多良师益友的帮助与支持。特别感谢我们的同事张桥艳在阅稿过程中给予的帮助。js 程序的调试是单调乏味的，一个 bug 的修复、一次绘图的改进可能要花费几天，甚至几周或一个月的时间。在此笔者要感谢家人、朋友及同事的关照和理解。

撰写本书的目的是希望能够起到抛砖引玉的作用，如果本书对读者能有一些启发那将是笔者莫大的欣慰。鉴于笔者水平所限，书中不妥之处，恳请读者批评指正。

著　者

2017 年 12 月

目　录

1 传输过程数值模拟程序开发综述

本章代码

本书介绍了传输过程数值模拟程序开发基础知识，主要参考文献［1］~［7］。本章通过一个高炉炉体传热实例[8]简要说明传输过程的数值计算的一般流程[9,10]。

柳宗元的《梓人传》记载了一位善于把握整体建筑架构的木匠，施工前先绘制好整体施工图，高屋建瓴，使役其他工匠，完成工程。程序开发同样如此，无论是动辄几百兆甚至数千兆的商业数值模拟软件，抑或一个几百行代码的小程序，在开发之前同样首先要理清程序架构（流程）。通常，传输过程数值模拟主要包含前处理、计算和后处理。炼铁高炉可视作高温反应容器，其侧壁通常为多种材料构成。图 1-1 为高炉墙体 3D 模型，包含炉皮、填充料、冷却壁体、镶砖、炉衬和渣皮，现探讨计算炉墙稳态温度场分布。

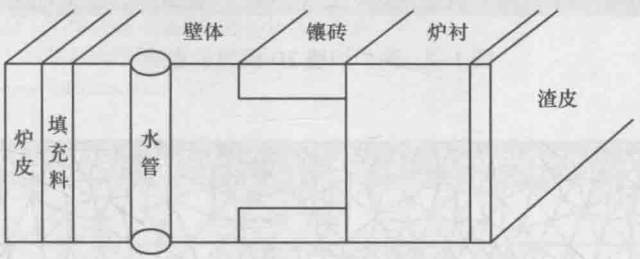

图 1-1　高炉炉体结构 3D 简化示意图

温度场求解计算步骤通常都包含了：

（1）建立物理几何模型：高炉炉腰炉腹部位的冷却壁，其传热方向主要为径向，故可简化为二维模型（图 1-2），同时也可以大大减少冷却壁温度场的计算量。

（2）确立数学模型（控制偏微分方程和定解条件）：该温度场的控制偏微分方程为二维稳态导热传输方程。定解条件为：炉皮与空气设为对流与辐射换热；渣皮或炉衬与高炉煤气设定为对流换热，或给定固定温度值；冷却水管与水为对流换热边界条件，给定水温与对流换热系数；其他为绝热边界条件。

（3）将计算区域进行网格剖分，如图 1-3 所示，并对各个单元格进行信息标记，即各单元格材料、边界条件等信息，以备后续计算使用。

（4）根据材料属性、边界条件、初始条件，确定各个节点温度场满足的方程组：

$$[K]\{T\} = \{P\} \tag{1-1}$$

式中，$[K]$ 为稳态温度场系数矩阵；$\{T\}$ 为温度矩阵；$\{P\}$ 为已知温度值。

（5）计算对方程组（1-1）进行计算，得到方程组的解，涉及到方程组求解相关知识。

（6）后处理：将结果以可视化方式呈现，如图 1-4 所示。

（7）分析：一般根据计算结果指导设计或者改进工艺等。

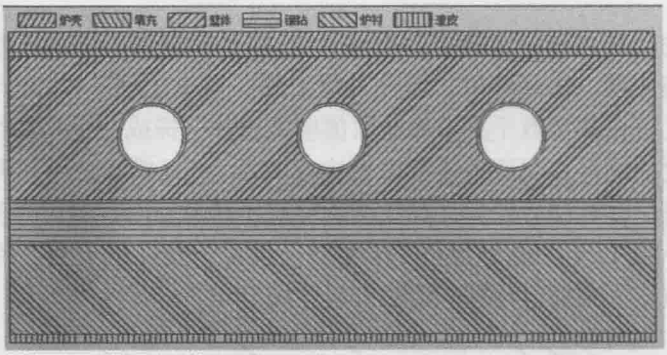

图 1-2　高炉炉墙 2D 模型示意图

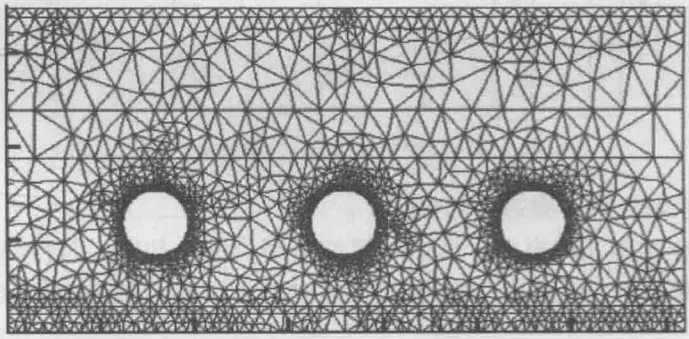

图 1-3　计算区域的网格剖分

本书按照上述步骤所涉及知识点对书中内容按照难易程度和认知顺序进行编排。第 2 章介绍了后处理——基于 HTML5 的数据可视化编程；第 3 章介绍了前处理——基于 Delaunay 算法的网格的剖分尝试；第 4 章以热传导为例介绍扩散方程；第 5 章对流—扩散方程的求解及介绍同位网格流场计算；第 6 章介绍使用非结构化网格基于有限元算法求解温度场。

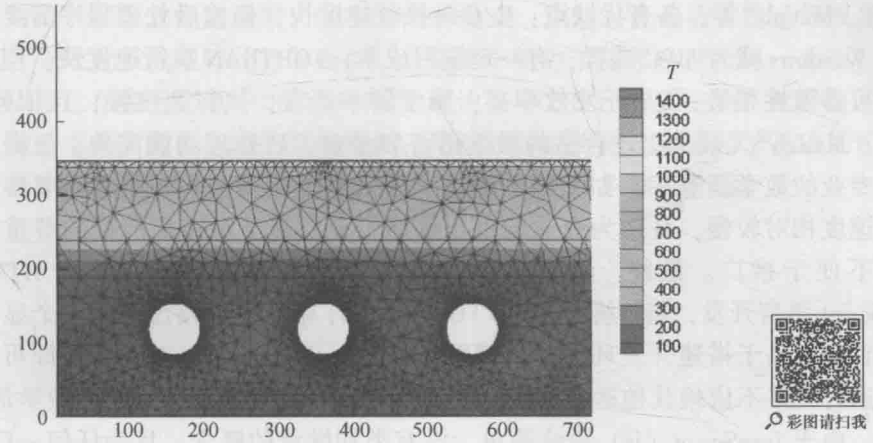

图 1-4 高炉炉墙温度场计算结果

根据上述数值模拟计算流程，本书开发的程序的架构流程示例如下（使用 JavaScript 编程语言）：

代码 1-1

```
function onSolve(){//本书所有模拟程序的计算流程图
    var nodes=[];//控制体序列
    var solution=new Solution(nodes);//

    var nx=50,dx=1;//↓,设置几何体并剖分网格,类似于商业软件的建模与网格剖分模块
    solution.SetUpGeometryAndMesh(nx,dx);//设置几何体并剖分网格,第三者详述网格剖分

    var lmd=1,Cp=1,rho=1;
    var steel=new SimpleMaterial(lmd,Cp,rho);
    solution.ApplyMaterial(steel);//设置材料

    var Tini=0,Tair=1;
    solution.Initialize(Tini,Tair);//初始化,给定初值

    solution.SetUpBoundaryCondition();//设置边界条件,分散与各章节,视算例而定

    var iterations=100,timeStep=0.1;/*时间步长 timeStep,迭代 Iterations 次*/

    solution.Solve(iterations,timeStep);//计算系数矩阵并求解方程组,详见第 4/5/6 章内容

    solution.ShowResults();/*显示计算结果,详见第 2 章后处理绘图*/
}
```

本书程序实现的流程也与常见商业计算软件的设置/计算流程相近，关于本书实现程序的特点的几点说明：

（1）笔者在学习传输过程数值模拟过程中，阅读了大量书籍和文献，程序

开发所使用计算机语言形形色色，主要有 C/C++、FORTRAN、Visual Basic（VB）、Matlab®等，各有优缺点：C/C++执行速度快，但前后处理程序需要额外学习 Windows 或者 MFC 编程，有一定学习成本；FORTRAN 执行速度快，但可读性和可移植性稍逊；VB 开发效率高，属于脚本语言，执行速度慢，且依赖 VB 运行；Matlab®无疑是比较合适的教学语言和平台，后处理功能完善，还提供了非常专业的数学运算库，如大型方程组求解，但同样作为脚本语言需要解释执行，速度相对较慢，且作为商业软件，价格不菲，基于对知识产权的尊重和保护，不便于推广。最终，经过作者慎重考虑，书中程序统一使用 HTML5/JavaScript 语言开发，与传统 C/C++/FORTRON 计算机语言相比具备几个显著优势：首先，易于搭建开发环境，不需要编译器，仅需一个文本编辑器即可；其次，运行几乎不依赖其他运行时（库），仅需一个浏览器，所以便于教学演示；再次，由于 JavaScript（js）语法简单，没有类和继承的概念，且会任何一门 C-Style 语言都会很快上手，学习成本低，入门快；再次，当前 js 可以高效绘图，便于对计算结果进行后处理操作；最后，跨平台可运行于几乎所有主流操作系统，也可运行于个人电脑、平板和手机等，只需要一个支持 HTML5 标准的浏览器。当然与传统 C/C++语言相比，HTML5/js 最大的不足是运行速度较慢。本书的目的在于介绍算法，给出一些易懂、便于扩展、简单的传输过程数值模拟程序，而不是进行大规模数值计算，所以使用了 HTML5 技术。

（2）笔者曾经尝试阅读优秀的开源计算流体力学程序包 OpenFOAM，由于代码量巨大，限于笔者理解能力而读后不得要领，没有坚持下来，所以本书中代码尽可能短，变量命名也尽量通俗易懂。本书给出源代码目的不是让读者复制和粘贴书中代码完成作业或项目甚至课题，而是希望能够激发读者编写出自己的更好的程序。我国明朝大思想家王阳明提出“知行合一”，强调理论与实践的结合，而传输过程的数值模拟程序开发也是一个理论与实践紧密结合的课题；只熟悉理论知识或只精通程序开发并不能完全做到“知行合一”。“基础在学，关键在做”，本书希望能够起到的作用是整理和回顾理论基础，把关键落实在程序编写上。另外，本书涉及很多开源程序库，请注意开源不等于完全免费，使用过程中若涉及到商业应用，请仔细阅读其授权说明，尊重知识产权。

（3）为了最大程度保证书中每个程序计算结果的合理性，笔者编排的例子大都是具有解析解的算例，这样可以验证数值模拟结果合理与否，如数学物理方法理论给出的二维泊松方程的解析解、纯物质凝固温度分布解析解及一维对流—扩散方程解析解。书中所有程序在作者笔记本电脑（操作环境为：Windows® 10Pro 32 - bit、Intel® Core Duo CPU T6600 2.20GHz、2GB RAM，浏览器为 Mozilla® FireFox 45.0.1）、NOKIA® Lumia 2520 平板电脑的 IE 浏览器、华为® Mate 7 手机及中兴® Blade A1 手机上正常运行。

限于篇幅及笔者水平，书中并没有推导诸如扩散方程显式迭代计算稳定性条件、对流—扩散方程的离散、QUICK 计算格式、涡量—流函数方法边界涡量计算公式，有限元系数矩阵计算等内容，而是集中精力于程序实现；另一方面，理论推导工作已在参考文献中给出，前辈的理论推导工作已近乎完美，无需赘述。

阅读本书所需要一定的基础知识：传输原理中传热及流体力学基础知识、数值计算中线性方程组求解等，同时需要一定的面向对象编程基础。

2 后处理之使用 HTML5/js 实现数据可视化的尝试

本章代码

本书将主要使用 JavaScript（js）语言实现书中算法，本章仅介绍最基础的 HTML5/js 内容，深入了解请参考官方文档[11]。如无程序编写基础，强烈建议先熟悉一门 C-Style 语言（如 C/C++），并了解一些面向对象编程的概念，学习 js 将会变得相对容易一些。除了类型声明外，js 的大部分基础语法和 C/C++并无二异。

2.1 开发平台搭建

本书程序在 Windows® 10 操作系统上进行开发，针对 HTML5（H5）应用的开发工具有文本编辑器和调试工具：（1）编辑器有 Notepad++、Sublime、记事本等，本书推荐使用 Sublime。（2）调试工具使用常见浏览器内置的调试器，如 Microsoft® IE/EDGE、Google® Chrome 及 Mozilla® FireFox，本书推荐使用 FireFox。（3）Adobe® Dreamweaver 及 JetBrains® WebStorm 等大型商业开发软件集成了编辑器和调试工具。

2.2 HTML5 基础入门

H5 包含了 js、HTML 及 CSS（Cascading Style Sheet）技术，本书对 CSS 技术不做详细介绍。

2.2.1 js 基础

js 语法与 C/C++/java 等 C-Style 类型语言语法相近，如基本数据类型：

代码 2-1

```
1.  var b=true;//定义布尔变量 b,并赋值为 true,js 注释与 C/C++完全相同,此处不详述
2.  var i=0;//定义型变量 i,并赋值为 0
3.  var j=-1000;//定义整型变量 j,并赋值为-1000
4.  var k=0.123;//定义浮点数 k,并赋值为 0.123
5.  var s="Hello World";//定义字符串变量 s,并幅值为"Hello World"
6.  var obj={Name:"Soong",Age:28};//定义对象 obj,设置其属性:Name 为"Soong",Age 为 28
```

js 不需要像 C 语言一样显式的指定具体类型，如 int、double 等，统一使用 var 关键字声明变量。四则运算与 C 语言相同，如：

代码 2-2

```
1.  i+=10;//等同于 i=i+10
2.  i++;//等同于 i=i+1
3.  j/=100;//等同于 j=j/100
```

条件转移与循环语句与 C 语言类似，如下：

代码 2-3

```
1.  var num=100;//定义整数
2.  if((num%2)==1){//求 num 的余数,"%"为求余数运算符,与 C 语言一样
3.    console.log("奇数");//console.log()为 FireFox 浏览器内置的输出函数,类似于 C 语言中 printf
4.  }else{
5.    console.log("偶数");
6.  }
```

for 循环求 0 到 9 之和：

代码 2-4

```
1.  var sum:uint=0;
2.  for(var i=0;i<=9;i++){
3.    sum+=i;
4.  }
```

do-while 循环求 0 到 9 之和：

代码 2-5

```
1.  var sum=0,num=1;
2.  do{
3.    sum+=num++;
4.  }while(num<10)
```

循环语句中 continue 和 break 语句的用法也与 C/C++完全相同。js 数组与 C++标准库（STL）中 vector 类类似，用法如下：

代码 2-6

```
1.  var v=newArray(5);//定义一个长度为 5 的数组
2.  v[0]=1;v[3]=4;
3.  var v2=[];//定义数组 v2 与 v2=new Array();作用完全相同
4.  console.log(v.length)//输出数组 v 的长度
5.  var arr=newArray("one","two","three",3,4,5);
6.  arr.push(1.2,2.3,3.4);//使用 push 函数从尾部添加数据
7.  arr.pop();//从尾部删除一个数据
```

js 中定义和使用求和函数：

代码 2-7

```
1.  function AddFun(a,b){
2.    var res;
3.    res=a+b;
4.    return res;
5.  }
6.
7.  var c=AddFun(1,2);
```

与 C 语言不同需要注意的是：部分浏览器不支持 js 函数使用默认参数，如 IE 和 EDGE。js 内置了一些数学函数，如指数运算、开方、三角函数等，类似于 C 语言中 math. h 中的数学函数，js 内置数学函数有：

代码 2-8

```
var rnd=Math. random( );//使用 random( )返回 0 到 1 之间的随机数
var max=Math. max(2,5,8,1);//使用 max( )返回两个给定的数中的最大值
var pi=Math. PI;//Math. PI 表示圆周率
var sqr=Math. sqrt(5);//求 5 的平方根
var i=Math. floor(3. 4);//求不超过 3. 4 的最大整数
var e=Math. exp(2);//指数函数
var s=Math. sin(Math. PI/6);//求 30 度角的正弦值
```

js 中没有类的概念，但可以实现类似于 C++中类的用法，便于模块化编程和代码维护，比如复数类的定义和使用，代码如下：

代码 2-9

```
//这是一个复数类,仅仅实现几个书中用到简单功能而已
var Complex=function(real,image){//function 关键字,一定情况下可以理解为类
  this. x=real;//成员变量:实部
  this. y=image;//成员变量虚部
    //如下定义成员函数
  this. lengthCPLX=lengthCPLX;
  this. getAngleCPLX=getAngleCPLX;
  this. addCPLX=addCPLX;
  this. subCPLX=subCPLX;
  this. pole2cplx=pole2cplx;
  this. rotateCPLX=rotateCPLX;
}
function lengthCPLX( ){//计算复数模
  returnMath. sqrt(this. x * this. x+this. y * this. y);
}
function getAngleCPLX( ){//计算复数角度
  return LineUtil. InclinationAngelOfPoint(this,true);
}
function addCPLX(cplx){//复数加法
  this. x+=cplx. x;this. y+=cplx. y;
  returnthis;
}
function subCPLX(cplx){//复数减法
  this. x-=cplx. x;this. y-=cplx. y;
  returnthis;
}
function pole2cplx(pLen,pAngle){//根据极坐标转换为笛卡尔坐标系
  returnnew Complex(pLen * Math. cos(pAngle),pLen * Math. sin(pAngle));
}
function rotateCPLX(theta,newLength){//将复数逆时针旋转 theat 弧度
  var newAngle=this. getAngleCPLX( )+theta;
```

```
32.     var cplx=this. pole2cplx( ( newLength)? newLength:this. lengthCPLX( ) ,newAngle) ;
33.     this. x=cplx. x;this. y=cplx. y;
34.     returnthis;
35.   }
36.   function test_Complex( ) {//测试使用刚刚定义的复数类
37.     var cplx=new Complex(1,1) ;//生成一个复数对象,设置其实部和虚部都为 1
38.     console. log( cplx. getAngleCPLX( ) ) ;//获取其角度
39.     cplx. rotateCPLX( Math. PI/4) ;//逆时针旋转 45 度
40.     console. log( cplx. x,cplx. y) ;//输出复数
41.   }
```

其中 js 中关键字 this 与 C++类构造函数内的 this 指针用法类似。本书大多程序段都用了类的概念，封装计算变量和计算函数，增强程序可读性。

2.2.2 HTML 基础

HTML（HyperText Mark-up Language）是由 HTML 标签嵌套和组合的描述性文本，HTML 标签可以描述文本（p，div 等）、表格（table）、图片（image）、音频（audio）、视频（video）、链接（a）等种类繁多内容。HTML 文件由头部（head）和主体（body）构成部分，头部用于制定标题及引用了那些 js/CSS 文件，主体用于描述具体呈现内容，如下例 CH2Contour. html：

代码 2-10

```
1.  <! doctype html>
2.  <htmllang=" en" >
3.    <head>
4.      <metacharset=" UTF-8" >
5.      <title>CH2:Contour Demo </title>
6.      <scriptsrc=" VisualizeLib. js" ></script>
7.      <scripttype=" text/javascript" src=" CH2Contour. js" ></script>
8.    </head>
9.    <body>
10.     <divstyle=" position:absolute;top:50px;left:50px;" >
11.       <canvasid=" canvasOne" width=" 600" height=" 400" >
12.       Your browser does not support HTML 5 Canvas.
13.       </canvas>
14.       </div>
15.   </body>
16. </html>
```

文件 CH2Contour. html 中声明了标题："CH2：Contour Demo"，指定引用 VisualizeLib. js 和 CH2Contour. js 两个 js 文档，类似于 C/C++包含头文件；主体里包含了标签 div，而 div 里面嵌套了一个 canvas 标签，而且给出该 canvas 的 id 为 canvasOne，宽度为 600，高度为 400。为便于理解，将上述 HTML 文档结构绘制

如图 2-1 所示。

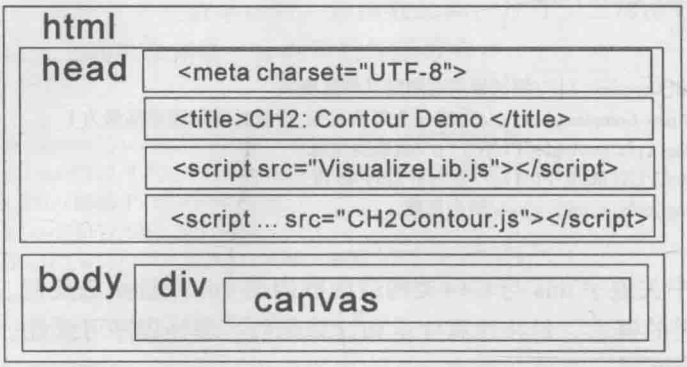

图 2-1　HTML 文档结构示意图

2.2.3　文档对象模型 DOM 及表单

文档对象模型（Document Object Model，简称 DOM），实现了通过 JavaScript 针对网页元素（标签）实现添加、删除、修改等操作，DOM 提供了大量函数来操作 HTML 文档，比如函数 getElementByID。标签的 id 是 HTML 元素的唯一标识符，js 中可以通过 document 的函数 getElementByID 来获取该元素，从而可以操作该标签元素。如例获取可用于绘图的 canvas 元素的绘图环境上下文，这个函数使用贯穿全书：

代码 2-11

```
1.  function GetCanvasContext(canvasID) {
2.    var theCanvas = document.getElementById(canvasID);//获取 id 为 canvasID 的标签元素
3.    return theCanvas.getContext("2d");//调用该元素函数,并返回调用结果
4.  }
```

再比如通过 js 修改网页标题：

代码 2-12

```
1.  document.title = "thisTitle";
```

HTML 标签中有一类特殊的标签：表单（form），用于显示控件，以使网页能够交互，如下代码定义了表单，内部包含了两个数字输入框和一个按钮：

代码 2-13

```
1.  <divstyle="width:500px;height:auto;float:left;display:inline">
2.    <formid="formA" width="500" height="400">
3.      设置计算参数:<br/>
4.      时间步长:<inputtype="number"id="timeStep"min="1"max="100"step=0.01value="1"/><br/>
5.      计算时间:<inputtype="number"id="during"min="1"max="1000"step=0.01value="120"/><br/>
6.      提交:<inputtype="button"value="求解"onclick="main()"/>
7.    </form>
8.  </div>
```

运行显示结果如图 2-2 所示。

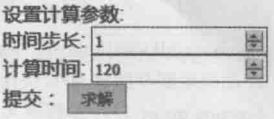

图 2-2 HTML form 使用的示例

如何在网页脚本中获取用户输入的参数呢？form 中的 button 定义了 onclick 属性，表明点击后会调用 main() 函数，main 函数获取用户输入，如下：

代码 2-14

```
1.  //根据 id 获取文本输入框的内容并转换为整形
2.  function GetInputNumber(id){
3.    var numberInput=document.getElementById(id);//获取控件
4.    var v=numberInput.value;//获取可见属性
5.    returnparseInt(v);//转换为整形
6.  }
7.  //程序入口
8.  function main(){
9.    var timeStep=GetInputNumber("timeStep")/1000;//获取时间步长
10.   var during=GetInputNumber("during");//获取求解时间
11.   console.log(timeStep,during);
12. }
```

2.2.4 HTML5 Canvas 绘图基础

Canvas 是 HTML 标准近年发展到 HTML5 时添加的新特性，用于在网页上高效绘制矢量图形，详细内容请参考文献[12]。早期版本的浏览器不支持 canvas，可以使用 modernizr.js 库（www.modernizr.com）封装的函数检测浏览器是否支持 HTML5 标准。H5canvas 绘图，与 MFC（Microsoft Foundation Classes）或 Visual Basic 等绘图步骤类似。下例给出 canvas 绘制直线路径、填充及输出文本示例，HTML 文本包含了一个 canvas 用于绘图：

代码 2-15

```
1.  <htmllang="en">
2.    <head>
3.      <metacharset="UTF-8">
4.      <title>CH2:Canvas Demo </title>
5.      <scripttype="text/javascript" src="CH2CanvasBasic.js"></script>
6.    </head>
7.    <body>
8.      <divstyle="position:absolute;top:50px;left:50px;">
9.        <canvasid="canvasOne" width="600" height="400">
10.       Your browser does not support HTML 5 Canvas.
11.     </canvas>
12.   </div>
13.  </body>
14. </html>
```

对应的 js 脚本文件 CH2CanvasBasic.js 如下：