高等学校电子与通信类专业“十三五”规划教材

时域有限差分法

(基于MATLAB)

姬金祖　马云鹏　张生俊
黄沛霖　刘战合　编著

西安电子科技大学出版社
http://www.xduph.com

高等学校电子与通信类专业“十三五”规划教材

时域有限差分法(基于 MATLAB)

姬金祖　马云鹏　张生俊　黄沛霖　刘战合　编著

西安电子科技大学出版社

内 容 简 介

本书主要介绍时域有限差分法的 MATLAB 实现方法，书中充分利用编程技巧，用紧凑的代码来实现算法。本书共 10 章，主要内容包括时域有限差分法的差分格式、吸收边界条件、完全匹配层边界条件、连接边界条件、远场外推、色散介质模拟、周期边界条件等，并通过典型几何体的电磁散射、界面的反射系数、一维光子晶体、二维光子晶体等算例进行验证。本书提供了部分 MATLAB 源代码，可供相关领域学者学习和参考。

本书可作为高等学校电磁学专业本科生、研究生的教学用书，亦可供其他有关专业的师生及科研人员参考。

图书在版编目(CIP)数据

时域有限差分法(基于 MATLAB)/姬金祖，等编著. —西安：西安电子科技大学出版社，2018.5
ISBN 978-7-5606-4913-9

Ⅰ. ① 时… Ⅱ. ① 姬… Ⅲ. ① 电磁波—时域分析—有限差分法—Matlab 软件
Ⅳ. ① O441.4-39

中国版本图书馆 CIP 数据核字(2018)第 093980 号

策划编辑 刘小莉
责任编辑 雷鸿俊
出版发行 西安电子科技大学出版社(西安市太白南路 2 号)
电 话 (029)88242885 88201467 邮 编 710071
网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com
经 销 新华书店
印刷单位 陕西天意印务有限责任公司
版 次 2018 年 5 月第 1 版 2018 年 5 月第 1 次印刷
开 本 787 毫米×1092 毫米 1/16 印张 16
字 数 377 千字
印 数 1～2000 册
定 价 37.00 元
ISBN 978-7-5606-4913-9/O
XDUP 5215001-1

前 言

时域有限差分法(FDTD)是一种时域电磁算法，参数设置灵活，对复杂介质的模拟具有先天优势。该算法自1966年由Yee提出以来发展迅速，获得了广泛应用。FDTD方法将电场和磁场分别在空间和时间上交错采样，将麦克斯韦方程组转化为差分方程，表述十分简单，容易理解。但是在具体编程实现时，涉及多个维度、多种场量，处理起来非常繁琐。本书采用MATLAB语言编程实现FDTD，充分利用MATLAB向量化编程的特点，将复杂的运算在尽量短的代码内完成，大大简化了编程。对于初学者，这是一本很好的入门教材；对于已经具有一定基础的学者，本书也能够给予一定的参考。

本书共10章。第1章主要介绍MATLAB的一些编程技巧。市面上已经有大量关于MATLAB的教材，本书不再详述，而是只挑选一些与本书的代码密切相关的内容进行讲解，如向量化运算、维度拓展等。第2章介绍电磁波基础理论。该章中的一些内容可作为理论基础，应用到后续章节的算法中；另一些内容求出了典型问题的解析解，其结果可以作为验证算法的依据。第3章介绍了FDTD的网格划分方法以及时间推进方法，讨论了空间和时间步长对仿真的影响。第4章和第5章分别介绍了Mur吸收边界条件和完全匹配层(PML)吸收边界条件。研究开域问题时，由于计算机内存有限，只能计算有限区域的场，因此必须在截断边界处加以处理，吸收外向行波，以模拟无限大空间。第6章介绍各种激励源的特点。第7章介绍连接边界条件使用，通过连接边界入射波引入到总场区。第8章介绍远场外推方法。在很多问题中，人们更关心的是远场情况，如计算RCS。FDTD的优势之一就是通过脉冲响应的傅立叶变换得到整个频域上的解，因此这一章以瞬态场的外推为主。第9章介绍了色散介质的处理方法，包括递归卷积法、Z变换方法以及辅助微分方程法。第10章介绍了周期边界的处理方法，以垂直入射情形为主，通过光子晶体、频率选择表面等算例对算法进行了验证。

北京航空航天大学的姬金祖主笔完成了本书的大部分内容，北京航空航天大学的马云鹏参与了第1章MATLAB编程技巧方面的撰写，北京航天长征飞

行器研究所的张生俊参与了第 10 章色散光子晶体和频率选择表面部分内容的撰写，北京航空航天大学的黄沛霖参与了第 8 章远场外推计算 RCS 部分内容的撰写，郑州航空工业管理学院的刘战合参与了第 2 章电场基础理论部分内容的撰写。

MATLAB 软件可以将计算结果表示为曲线图，但绘图质量并不高，字体、字号设置比较困难。本书的策略是尽量将计算结果都保存成文本形式的数据，然后用 LaTeX 的 pgfplots 宏包绘制。

本书所有的代码都在 Linux Ubuntu 17.10 版本上安装的 MATLAB 2017a 中完成。此外，本书的代码保存了两个版本，分别用于在 Linux 和 Windows 上运行。两个版本只有换行符有区别，其他完全相同。本书完整源代码及运行结果可登录出版社官网下载。

由于作者水平有限，书中不足之处在所难免，若读者有修改的意见和建议，请及时给作者反馈，以期再版时提高教材质量。作者电子邮箱：jijinzu@buaa.edu.cn。

作　者

2018 年 1 月

目 录

第 1 章　MATLAB 编程技巧

具有 Fortran 或 C 语言编程经验的读者可能有这样的体会，当涉及矩阵运算时，编程会很麻烦。例如，想要求解一个线性方程组，需要首先写一个主函数，然后编写一个子程序去读入各个矩阵的元素，之后再编写一个子程序，求解相应的方程，最后输出计算结果。这样一个简单的问题往往要编写很多代码，仅键入和调试就很繁琐。

1980 年前后，MATLAB 的首创者 Cleve Moler 博士在 New Mexico 大学讲授线性代数课程时，看到了用高级语言编程解决工程问题的诸多不便，因而构思开发了 MATLAB 软件。该软件利用了 Moler 博士在此前开发的 LINPACK(线性代数软件包)和 EISPACK(基于特征值计算的软件包)中可靠的子程序，用 Fortran 语言编写而成，集命令翻译、工程计算功能于一身。20 世纪 80 年代初期，Cleve Moler 和 John Little 采用 C 语言改写了 MATLAB 的内核。不久，他们成立了 Mathworks 软件开发公司，并将 MATLAB 正式推向市场。

现在 MATLAB 功能早已不只停留在工程计算的功能上了。它由主包、Simulink 以及功能各异的辅助工具箱组成。它以矩阵运算为基础，将计算、可视化、程序设计融合到了一个简单易用的交互式工作环境中。

经过不断的发展，MATLAB 已经成为国际公认的优秀数学应用软件之一。在美国等发达国家的大学里，MATLAB 是一种必须掌握的基本工具。而在国外的研究设计单位和工业部门，MATLAB 更是研究和解决工程计算问题的一种标准软件。在国内也有越来越多的科学技术工作者参加到学习和倡导这门语言的行列中来。

关于 MATLAB 教程已经有很多出版物，该软件自带的帮助文档也非常完备，基本能够满足学习的需求。因此，本书不对 MATLAB 的使用做深入详细的说明，而只介绍 MATLAB 最基本的使用方法以及 FDTD 的编程技巧，同时，对代码的维护、编写风格也提出一些建议。

1.1　MATLAB 基本操作

启动 MATLAB 后，将打开一个欢迎界面，随后打开桌面系统。桌面系统由桌面平台以及组件组成，其组件主要包括命令窗口、历史命令窗口、路径浏览器、工作空间浏览器、编辑器等。

MATLAB 是一种解释语言，代码不需要经过编译即可运行。运行 MATLAB 代码有两种方式：一种是交互方式，即在命令窗口中输入代码，然后按回车键，代码随即开始运行，计算完成后，结果显示在命令窗口中；另一种是脚本方式，将 MATLAB 代码保存在后缀为“.m”的脚本文件中，然后通过按快捷键 F5 或点击运行按钮运行代码。

在实验一些不熟悉的命令或者执行一些简单的操作时，一般采用交互方式运行命令，

及时获得结果。在连续运行大量的代码时，逐句输入指令效率很低，也很不方便修改，此时一般将代码保存在脚本文件中。用交互方式运行代码，方便代码的调试和修改。

1.2 向量化编程方法

MATLAB 语言对循环语句的处理效率较低，但其内置了向量化编程的方法，可以极大提高计算效率。下面用一些简单代码进行实验，以对比向量化程序语句和循环语句的效率。对两个有 10^8 个元素的一维数组进行相加，代码如下所示。

代码 1.1 一维数组相加运算(1/m1.m)

```
n=10^8;
a=1:n;
b=a;
tic;
c=zeros(1,n);
for  ii=1:n
     c(ii)=a(ii)+b(ii);
end
t1=toc;
tic;
c=a+b;
t2=toc;
data=[t1 t2 t1/t2];
save([mfilename,'.dat'],'data','-ascii');
```

用作者的计算机进行仿真，计算结果表明，采用循环语句和采用向量化编程方式的运行时间分别是 1.0393 s 和 0.0892 s，两者之比为 11.651。

下面的程序对两个 $10^4 \times 10^4$ 的二维数组对应元素进行相加，实验计算所需要的时间。

代码 1.2 二维数组相加运算(1/m2.m)

```
n=10^4;
a=reshape(1:n^2,n,n);
b=a;
tic;
c=zeros(n,n);
for ii=1:n
    for jj=1:n
        c(ii,jj)=a(ii,jj)+b(ii,jj);
    end
end
t1=toc;
tic;
c=a+b;
t2=toc;
```

```
15 data=[t1 t2 t1/t2];
16 save([mfilename,'.dat'],'data','-ascii');
```

结果表明，采用循环语句和采用向量化编程方式的运行时间分别是 2.028 s 和 0.0902 s，两者之比为 22.483。

下面的程序对两个 500×500×500 的三维数组对应元素进行相加，实验计算所需要的时间。

代码 1.3　三维数组相加运算(1/m3.m)

```
1 n=500;
2 a=reshape(1:n^3,n,n,n);
3 b=a;
4 tic;
5 c=zeros(n,n,n);
6 for ii=1:n
7     for jj=1:n
8         for kk=1:n
9             c(ii,jj,kk)=a(ii,jj,kk)+b(ii,jj,kk);
10        end
11    end
12 end
13 t1=toc;
14 tic;
15 c=a+b;
16 t2=toc;
17 data=[t1 t2 t1/t2]
18 save([mfilename,'.dat'],'data','-ascii');
```

结果表明，采用循环语句和采用向量化编程方式的运行时间分别是 10.6818 s 和 0.1132 s，两者之比为 94.3622。可见，采用向量化编程的方式能够极大地提高计算速度，充分发挥 MATLAB 的优势。本书所编写的代码，都尽量采用向量化的编程风格，减少循环语句的使用。只有算法在无法采用向量化语句实现的时候，才采用循环语句。

在一些问题中，往往能遇到一个二元函数表示成级数的形式，而级数的每一项都可以分离变量。设函数 f 是 x 和 y 的函数，可以表示为

$$f(x, y) = \sum_{k=1}^{\infty} g(x, k)h(k, y) \tag{1.2.1}$$

在数值计算中，x、y 和 k 都只能取有限项。k 取 K 项，则上式变成有限项的和式：

$$f(x, y) \approx \sum_{k=1}^{K} g(x, k)h(k, y) \tag{1.2.2}$$

再设 x 的离散点为 x_1，x_2，…，x_M，y 的离散点为 y_1，y_2，…，y_N，则上式变为

$$f(x_m, y_n) \approx \sum_{k=1}^{K} g(x_m, k)h(k, y_n) \tag{1.2.3}$$

上面的求和式可以表示为两个矩阵的乘积，即

$$F_{mn}=\sum_{k=1}^{K}G_{mk}H_{kn} \tag{1.2.4}$$

其中

$$F_{mn}=f(x_m,\ y_n) \tag{1.2.5a}$$

$$G_{mk}=g(x_m,\ k) \tag{1.2.5b}$$

$$H_{kn}=h(k,\ y_n) \tag{1.2.5c}$$

可以利用矩阵乘法的性质简化代码，提高效率。用矩阵乘积的方式可以写成

$$\boldsymbol{F}=\boldsymbol{GH} \tag{1.2.6}$$

其中 $\boldsymbol{F}$、$\boldsymbol{G}$、$\boldsymbol{H}$ 分别是 $M\times N$、$M\times K$、$K\times N$ 的矩阵。

本书中大量使用将求和化为矩阵乘积的方法。如在求解导体圆柱或导体球的散射问题中，RCS 随频率和双站角的变化就表示成这种级数形式。在将时域计算结果过渡到频域结果时，也采用这种方式处理。

1.3 数组的自动扩展

在数组运算中，经常会遇到对一个数组的某个维度所有数据进行同样运算的操作，如对一个二维数组的每一行所有元素加上一个数，不同的行加的数不同。如一个 3×4 的数组，第一行加上 1，第二行加上 2，第三行加上 3。第一种方法是采用循环语句，如下所示。

代码 1.4 循环语句处理不同维度数组的运算(1/m4.m)

```
a=reshape(1:12,3,4);
b=[1;2;3];
c=zeros(size(a));
for  ii=1:3;
     c(ii,:)=a(ii,:)+b(ii);
end
```

经前面的分析和算例验证可知，这种采用循环语句的方法效率较低。

另一种方法是将数组进行扩展，使两个数组的维度相同，如下所示。

代码 1.5 数组扩展方式处理不同维度数组的运算(1/m5.m)

```
a=reshape(1:12,3,4);
b=[1;2;3];
c=a+repmat(b,1,4);
```

这种方法需要手动将数组进行扩展。

第三种方法是用 bsxfun 函数。该函数将不同尺寸的矩阵自动进行扩展，使扩展后的维度尺寸相同，进而可以进行运算，如下所示。

代码 1.6 bsxfun 方式扩展数组(1/m6.m)

```
a=reshape(1:12,3,4);
b=[1;2;3];
c=bsxfun(@plus,a,b);
```

上面的代码中，bsxfun 函数有 3 个参数，第一个参数表示函数句柄，第二个和第三个

参数是两个数组。并非所有的二元函数都能够作为 bsxfun 的参数，必须要满足一定的条件。MATLAB 的一些内置二元函数可以作为 bsxfun 的参数，这些函数包括 plus、minus、times、rdivide、ldivide、power、eq、ne、gt、ge、lt、le、and、or、xor、max、min、mod、rem、atan2、atan2d、hypot。自己定义的函数满足一定的条件后，也可以使用 bsxfun 函数进行维度扩展。

参与运算的两个数组 A 和 B 的维度要满足相容性条件，即两个数组对应维度的长度要么相等，要么其中之一是 1。如果两个数组的维数不同，那么维数小的数组自动将维数扩大到与维数大的数组相等，方法是将高维的维数自动设为 1。如上面的例子，一个数组是二维数组，尺寸是 3×4，另一个是 3 行 1 列的一维数组。首先将后一个数组扩展为二维数组，尺寸为 3×1。然后观察两个数组对应的维度，第一个维数都是 3，第二个维数分别是 4 和 1，满足相容性条件，于是可以通过 bsxfun 函数进行计算。

自 MATLAB 的 R2016a 版本开始，上述这些内置函数在运算时会将维度尺寸相容的数组自动扩展成尺寸完全相同，因此可以省去 bsxfun 函数，直接进行处理。如上面的例子可以用下面的语句完成。

代码 1.7 数组自动扩展方式处理不同维度数组的运算(1/m7.m)

```
1 a=reshape(1:12,3,4);
2 b=[1;2;3];
3 c=a+b;
```

因此，代码又可以极大简化。在 Python 语言中，这种不同尺寸的数组进行运算时自动扩展的机制称为“广播(broadcast)”。

1.4 计算结果可视化

MATLAB 自带了数据可视化的函数，可以将计算结果显示为曲线、曲面、散点图、极坐标图等。本书用得最多的是 plot 和 imagesc 函数。

plot 函数可将数据显示为曲线形式，如下面的例子。

代码 1.8 MATLAB 生成的曲线图(1/m8.m)

```
1 a=1:360;
2 b=sind(a);
3 plot(a,b);
4 saveas(gcf,[mfilename,'.png']);
```

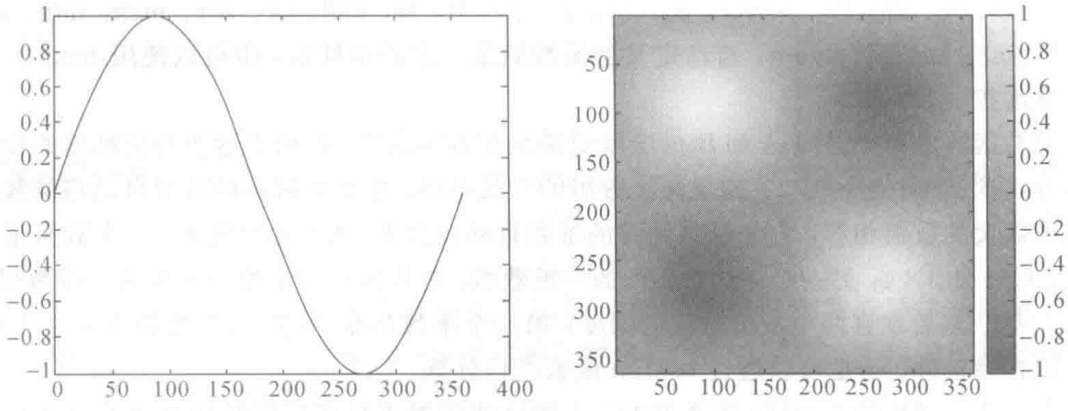
imagesc 函数用来将二维数组可视化，如下面的例子。

代码 1.9 MATLAB 生成的灰度图(1/m9.m)

```
1 a=-180:180;
2 b=a';
3 c=sind(b)*sind(a);
4 imagesc(c);
5 axis equal tight;
6 colorbar;
```

```
7 saveas(gcf,[mfilename,'.png']);
```

上面两个例子的显示结果如图 1.1 所示。



(a) plot函数绘制的曲线图(1/m8.m)　　(b) imagesc函数绘制的灰度图(1/m9.m)

图 1.1　MATLAB 数据可视化结果

以上两个函数还可以加更多的参数来对可视化结果进行精细设置，如设置坐标轴、标题等。但本书代码所用的方法大都是将计算结果保存在文本文件里，再用 LaTeX 的 pgfplots宏包进行绘图，可以得到高质量的插图。同时，计算结果也用 MATLAB 语句保存成 png 图片。MATLAB 保存的图片只显示必要的信息，更多的目的是便于直接查看结果，所以并不对图片做过多的修饰。为简洁起见，MATLAB 作出的曲线图和灰度图中坐标轴、图例、图题等都没有给出来，需要结合代码来判断曲线的含义。

1.5　MATLAB 编程原则

根据多年来的编程经验，作者总结了几条采用 MATLAB 进行程序开发的原则，部分原则受 UNIX 编程哲学的影响。好的编程习惯便于代码的维护、阅读和修改，不好的编程习惯写出来的代码则繁冗拖沓，难以阅读，而且不好修改。如果只是研究某种算法，而不是开发大型程序，则更应该尽量遵循这些原则。

(1) 简洁原则：设计要简洁，复杂度能低则低，尽量用更短的代码实现算法。越短的代码，越不容易出错，出错了也越容易修改。如果一个变量只在一个地方出现，则不用设置此变量，在出现的地方用数字代替。

(2) 拿来原则：尽量使用 MATLAB 自带的函数，而不要花费时间重新去写算法。

(3) 独立原则：脚本文件能够独立运行，而不需要调用自己编写的其他脚本。采用这个原则可以减少代码的维护成本，避免在运行过程中到处找调用其他脚本。尽量在一个脚本文件中将前处理、迭代求解、后处理同时实现。

(4) 最小原则：在验证某个算法时，脚本文件只实现所要验证的算法，不加入其他功能。如验证差分格式的代码里，就不加入吸收边界条件。

(5) 自动原则：MATLAB 能够自动处理的变量就让 MATLAB 去处理，而不用手动干涉。手动处理得越多，出错的概率越大。如果一个常数在多个地方出现，则将其设置为一个变量。

(6) 吝啬原则：除非确无他法，不要编写庞大的程序。宁可写很多个小程序，也不要写一个大程序。大程序维护起来非常繁琐，容易出错，调试困难。

(7) 优化原则：雕琢前先要有原型，跑之前先学会走。《计算机程序设计艺术》和 TeX 的作者 Donald Ervin Knuth(高纳德) 曾经在论文中说过，过早的优化是万恶之源。本书中的代码相当精炼，不过它们也是作者从一些繁琐的代码优化而来的。读者在实践的时候，可以尝试用自己风格的代码实现算法，再逐渐优化。

针对 FDTD 的仿真，采用 MATLAB 编程时，最好再遵循以下原则：

(1) 程序中尽量使用无量纲量。使用无量纲量使得程序中的变量大小适中，避免产生很大或很小的数，以提高精度。而且，采用无量纲量的计算结果更能体现电尺寸的特点。

(2) 尽量完善理论分析和公式推导，程序只做重复的工作。这个原则可以进一步简化代码，便于查错。如果程序中有很多常数之间运算的语句，则检查这些语句需要花费额外的时间，而且出错了也不容易发现。

(3) 变量尽量选用较短的变量名，减少代码行的长度。

(4) 用统一的命名规则对变量名进行命名。本书最后附录里列出了书中代码的变量名命名规范。

复习思考题

1. 通过编程对比研究循环语句和向量化语句的运行效率差异。
2. 对比几种不同尺寸的数组运算方法，比较优劣。

第 2 章　电磁学基础

本章介绍电磁学的基本原理，包括场的概念、麦克斯韦方程组的各种形式、平面波的解、典型散射问题的解析解等。通过本章的学习，能够对电磁学有基本的了解，为后面章节麦克斯韦方程组的离散、差分、边界条件处理以及远场外推奠定基础。

2.1　场　　论

2.1.1　场的基本概念

“场”是物理学中经常遇到的概念，如电场、磁场、温度场、引力场、重力场等。从数学角度来看，场其实是“空间和时间的函数”。从数学角度理解“场”的概念会更加容易，避免被复杂的物理意义干扰。场体现出某种物理量在空间中“分布”的概念。以温度为例，表述某个区域的温度大小，可以只给出一个数，表示这个区域温度的代表值。尽管某个时刻一个城市不同区域的温度有很大差异，但城市温度的实时播报一般也只用一个数字表示温度大小，反映这个城市温度的平均水平，不能反映温度在城市里不同区域的差异。但“温度场”以函数的形式给出了给定区域任意位置的温度大小，函数的定义域是给定区域，自变量是给定区域的位置，因变量是该位置的温度。用数学语言来表示，在三维空间中，温度场是实数三元函数，每个自变量表示一个维度。当需要表示温度随时间的变化时，就要加入第四个变元，即时间。温度场表示为三维空间和一维时间的函数时，就能够完整表示空间中任意一点在任意时刻的温度情况。连续介质力学中的速度场、压力场、应力场、应变场等概念也与此类似。

按照场随空间变化的维度，场可以分为三维场、二维场和一维场。三维场随三维空间变化，空间坐标必须用三个实数来表示。当三维场在某一个或两个维度上不发生变化时，维度就可以降低一维或二维，按照二维场或一维场来研究，使问题加以简化。一般情况下，低维的问题比高维的问题更简单，在研究过程中，要尽量将高维问题简化为低维问题。这里的维度不一定是直角坐标的维度，也可以是曲线坐标的维度。如果某球内的温度只与该点到球心的距离有关，则温度可以写成到球心的距离 r 的函数，此时温度场可以用一维函数来描述，但中间过程可能需要进行坐标变换。

按照随时间的变化情况，场可以分为时变场和稳恒场。时变场是随时间变化的场，稳恒场是不随时间变化的场。显然，稳恒场不随时间变化，变元更少，一般情况下更容易处理。一种场是稳恒场还是时变场，与所研究的场的性质有很大关系。如果研究的问题时间很短，场在所关心的时间范围内基本上不发生变化，则可以近似看做稳恒场。

时谐场是一种特殊的时变场，也是重要的时变场，场随时间呈正弦变化。时谐场的研究可以通过引入复数的形式，转化为对等效稳恒场的研究。