

# 復旦大學



## 现代集成电路分析方法

### Project 5 Solve MNA

专业：电子科学与技术

学号：22112020002

姓名：蔡志杰

# 1 题目

请设计一个程序，用后向 Euler 法/梯形法求解电路系统的 MNA 方程（该方程由 stamp 程序得到），其中线性方程的求解可以选择如下几种方法：直接法 (Project III)，稳态迭代法 (Jacobi, G-S, SOR, Project IV)，GCR 迭代法 (Project V)。

$$C\dot{X} + GX = BU \quad (1)$$

$$Y = L^T X \quad (2)$$

系统的初值  $X_0$  通过求解  $GX_0 = BU(0)$ ，也就是 DC 分析得到。

## 1.1 输入

输入的线性电路为 SPICE 格式的网表文件

(1) 电路方程：以提供的 stamp 程序的输出作为本程序的输入。

(2) 部分参数：模拟时间长度  $T$ ，误差容限  $E_{pi}$ 。

(3) 电路激励：可以将源写成独立的函数，对不同的问题相应手工修改该函数，应至少支持 sin 以及 pulse 输入 (关于这两种输入的具体参数请参考 HSPICE 手册)。

(4) 标准结果：以 SPICE 的模拟结果作为标准结果，用于和程序模拟结果比较。为方便比较结果，可以将程序的时间步长设为与 SPICE 一样，在时间不长不一致的情况下，则可以通过插值的方法得到在相同的时间步长下的结果。

## 1.2 输出结果

输出为四张图示和部分数值结果。

图 1 为输入波形。图 2 为模拟得到的输出波形。图 3 为模拟得到的输出波形和标准结果 (SPICE 结果) 的比较。图 4 为整个模拟区间上的绝对误差分布。同时程序应该输出：Euler 方法的计算点数，总模拟时间，整个模拟区间上的最大绝对误差 (绝对值) 和均方绝对误差。

\* 计算误差的采样点即为标准结果中给出的时间点。如该点不是 Euler 方法的计算点，则模拟结果通过邻近两个计算点模拟结果的线性插值得到。

\*\* 均方误差的定义： $MSE = \sqrt{\frac{1}{n} \sum_{i=0}^n E_i^2}$ ，其中  $E_i$  为每个采样点上的误差， $n$  为采样点数。

### 1.3 测试用例

Benchmark 目录下提供三个测试用例 RLC\_s3.sp, bus32bit8seg.sp 以及 bus8bit8seg.sp, 并提供 Matlab 下的 stamp 程序供构造电路矩阵。stamp 用法请参考 Benchmark 目录下的 stamp\_man 文件。同时提供 read\_data 程序来读取 hspice 输出 \*.lis 文件中的波形数据。read\_data 程序的用法请参考 benchmark 目录下的 read\_data\_man 文件。Stamp 程序支持 sin 以及 pulse 的输入, 读取的数据在 SRC 变量中。

### 1.4 提交结果

程序建议采用 MATLAB 完成, 需提交以下内容:

- (1) 源程序, 应有必要的注释。
- (2) 使用除 MATLAB 外其他语言的, 需要提交最终编译的可执行代码。
- (3) 一份完整的说明, 主要内容包括: 主要设计思想, 程序结构, 编译的环境和方法, 运行的环境和方法, 输入的格式或方法, 以及其他需要特别说明的地方。
- (4) 对测试用例的测试结果和分析。

## 2 算法原理

首先通过 stamp 程序获得 spice 网表对应的电路 MNA 方程。对于得到电路方程, 可以采用特征值分解的方法进行求解(耗时)或采用有限差分方法对其进行求解, 其中有限差分法, 包含前向欧拉(Forward Euler)、后向欧拉(Backward Euler)和梯形法(Trapezoidal Rule)。在有限差分法中又涉及到线性方程组的求解, 求解线性方程组的方法有直接法(高斯消元法, LU 分解, QR 分解等)、迭代法(Jacobi, Gauss-Seidel, SOR 等)和广义共轭残差法 GCR。

### 2.1 偏微分方程求解

对于如下的偏微分方程, 我们介绍前向欧拉、后向欧拉和梯形法。

$$\frac{dx(t)}{dt} = Ax(t) \quad (3)$$

$$x(t_0) = x_0 \quad (4)$$

### 2.1.1 前向欧拉法 Forward Euler

前向欧拉法的公式如下, 使用时间上更迟的数据点做差分来逼近梯度, 只需要矩阵乘法, 而不需要求解线性方程, 速度更快, 但是精度欠佳。

$$\frac{d}{dt}x(t_l) = Ax(t_l) \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t} \quad (5)$$

$$x(t_{l+1}) \cong x(t_l) + \Delta t Ax(t_l) \quad (6)$$

公式 1 中前向欧拉的迭代公式为:

$$CX(t_{l+1}) = (C - \Delta t G)X(t_l) + \Delta t BU(t_l) \quad (7)$$

### 2.1.2 后向欧拉法 Backward Euler

后向欧拉法的公式如下, 使用时间上更早的数据点做差分来逼近梯度, 需要求解线性方程, 速度慢于前向欧拉, 但是精度更好。在等步长  $\Delta t$  的情况下, 可省去重复的计算步骤来实现加速。

$$\frac{d}{dt}x(t_{l+1}) = Ax(t_{l+1}) \cong \frac{x(t_{l+1}) - x(t_l)}{\Delta t} \quad (8)$$

$$x(t_{l+1}) \cong x(t_l) + \Delta t Ax(t_{l+1}) \quad (9)$$

公式 1 中后向欧拉的迭代公式为:

$$(C + \Delta t G)X(t_{l+1}) = CX(t_l) + \Delta t BU(t_{l+1}) \quad (10)$$

### 2.1.3 梯形法 Trapezoidal Rule

梯形法的公式如下, 使差分来逼近相邻两点梯度的平均值, 计算比后向欧拉更加复杂, 但是精度更好。

$$\frac{1}{2} \left( \frac{d}{dt}x(t_{l+1}) + \frac{d}{dt}x(t_l) \right) = \frac{1}{2} (Ax(t_{l+1}) + Ax(t_l)) = \frac{x(t_{l+1}) - x(t_l)}{\Delta t} \quad (11)$$

$$x(t_{l+1}) = x(t_l) + \frac{1}{2} \Delta t A (x(t_{l+1}) + x(t_l)) \quad (12)$$

公式 1 中梯形法的迭代公式为:

$$(2C + \Delta t G)X(t_{l+1}) = (2C - \Delta t G)X(t_l) + \Delta t B(U(t_{l+1}) + U_l) \quad (13)$$

### 2.1.4 小结

以上三种方法都是 one-step 方法，只依赖于前一时间点的数据。前向欧拉是最简单的方法，后向欧拉更加复杂，而梯形法可能是最精确的。

从稳定性上看，前向欧拉的时间步长需要足够小来保证解的稳定性（极点不位于右半平面，解是等幅震荡或衰减的）。后向欧拉则能在原始解稳定的情况下保证解的稳定性，当原始解发散的时候只要时间步长足够大也能保证解的衰减性。梯形法同样在解稳定时能得到稳定的解，当解不稳定（幅度不断增大时），得到的解也不稳定。

## 2.2 线性方程求解

对于如下的线性方程组，为了求解  $x$  的值，通常有直接法（高斯消元法，LU 分解，QR 分解等）、迭代法（Jacobi, Gauss-Seidel, SOR 等）和广义共轭残差法 GCR。

$$Mx = b \quad (14)$$

接下来主要迭代法和广义共轭残差法 GCR。

### 2.2.1 Jacobi 迭代法

将矩阵拆分成上三角、对角线和下三角矩阵，通过移项得到迭代公式：

$$M = D - L - U \quad (15)$$

$$D - L - U x = b \quad (16)$$

$$Dx_k = (U + L)x_{k-1} + b \quad (17)$$

迭代速度较慢，没有合理利用前面迭代得到的信息。

### 2.2.2 Gauss-Seidel 迭代法

和 Jacobi 方法相比，利用了更多先前迭代中的结果，能获得更快的收敛速度：

$$D - L x_k = Ux_{k-1} + b \quad (18)$$

### 2.2.3 Successive Over Relaxation 逐次超松弛法

基于 Gauss-Seidel 迭代法，进行修改来提高迭代收敛速度，

$$x_k = w\overline{x}_k + (1+w)x_{k-1} \quad (19)$$

$$\overline{x}_k = D^{-1}Lx_k + D^{-1}Ux_{k-1} + D^{-1}b \quad (20)$$

$$x_k = (D - wL)^{-1}((1-w)D + wU)x_{k-1} + w(D - wL)^{-1}b \quad (21)$$

### 2.2.4 广义共轭残差法 GCR

第  $k$  轮残差的定义为，

$$r^k \equiv b - Mx^k \quad (22)$$

第  $k$  轮迭代的解  $x_k$  定义为，

$$x^{k+1} = \sum_{i=0}^k \alpha_i w_i \quad (23)$$

目标是最小化残差的 2 范数，当构成  $x$  的基向量关于  $M$  矩阵正交，即  $(Mw_i)^T(Mw_j) = 0(i \neq j)$  时残差的 2 范数很容易优化，

$$\min_x \|r^{k+1}\|_2^2 = \|b - \sum_{i=0}^k \alpha_i Mw_i\| \quad (24)$$

将其转换成一个二次型目标函数，取最优值的地方梯度为 0，梯度对应的就是残差，能够间接求解线性方程组。（只适用于矩阵  $M$  为对称正定阵）

$$f(x) = \frac{1}{2}x^T Mx - x^T b \quad (25)$$

$$\nabla_x f(x) = Mx - b \quad (26)$$

对目标函数  $f$  采用最速下降法，将梯度作为新的探索方向，经过分析可得如下的迭代公式：

$$\alpha_k = \frac{(r^k)^T (Mp_k)}{(Mp_k)^T (Mp_k)} \quad (27)$$

$$x^{k+1} = x^k + \alpha_k p_k \quad (28)$$

$$r^{k+1} = b - Mx^{k+1} = b - Mx^k - \alpha_k Mp_k = r^k - \alpha_k Mp_k \quad (29)$$

$$p_{k+1} = r^{k+1} - \sum_{j=0}^k \frac{(Mr^{k+1})^T (Mp_j)}{(Mp_j)^T (Mp_j)} p_j \quad (30)$$

如果矩阵  $M$  是稀疏的, 具有  $k$  个特征值, 你们只需要  $k$  轮迭代即可收敛, 收敛速度更快。当矩阵  $M$  为对称矩阵时,  $r^{k-1} \perp Mp_j, j < k, p_{k+1}$  的正交化只需要一步操作。

$$p_{k+1} = r^{k+1} - \sum_{j=0}^k \frac{(Mr^{k+1})^T (Mp_j)}{(Mp_j)^T (Mp_j)} p_j \Rightarrow p_{k+1} = r^{k+1} - \frac{(Mr^{k+1})^T (Mp_k)}{(Mp_k)^T (Mp_k)} p_k \quad (31)$$

## 2.3 算法实现

### 2.3.1 算法流程

### 2.3.2 输入激励信号

### 2.3.3 迭代终止条件

### 2.3.4 输出结果

## 2.4 编译与运行

程序基于 Matlab 2014a (32 位) 平台, 将整个解析网表文件、MNA 方程求解和结果输出的过程都打包在 main 函数中, main 函数的接口为  $main(caseName, methodName, algorithmName, startTime, caseName(RCL_s3, RCL_p3, bus8bit8seg, bus32seg16, methodName BE TR, algorithmName SOR, 1e-2, 1000, 1e-9))$  (32)

## 2.5 结果与分析

