现代集成电路分析方法

Project 7 Shooting method

专业: 电子科学与技术

学号:22112020002

姓名: 蔡志杰

目录

1 题目

采用 shooting method 求解如下线性系统的周期稳态解:

$$C\dot{X} + GX = BU \tag{1}$$

$$Y = L^T X \tag{2}$$

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) 输入 u(t) = cos $(200\pi t)$, 也可以采用其他周期输入。

1.2 输出结果

(1) 通过 shooting method 求解得到的初值。

(2) 在求解的初值条件下, 通过后向欧拉方法得到的周期稳态解

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。

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。对于周期性的输入激励信号系统一般会得到一个周期稳态解, 使用前面基于有限差分的方法 (前向、后向欧拉和梯形法) 可以得到系统的稳定解, 但是需要很大的预算量 (从 0 时刻开始), 因此可以采用 Shooting 法来求解周期稳态解的初始解, 再基于这个初始解求解周期稳态解, 或是使用谱方法将周期稳态解用一组基函数表示, 再减小基函数线性组合后于原函数的残差来获得周期稳态解。

本项目的整体思路是首先根据 shooting method 将周期性稳态信号转换成一个状态函数, 并将周期信号在一个周期中收尾相等的约束转换成对一个函数零点的求解问题。之后使用牛顿法来迭代前面构造的函数零点, 进而求解周期信号的初值。最后根据求解得到的初值再使用后向欧拉来求解在周期激励信号的作用下系统的周期稳态解。接下来将分别介绍整个框架中涉及到的算法, 包括 Shooting method, 牛顿迭代法, 后向欧拉法和广义共轭残差法。

2.1 Shooting Method

Shooting method 以瞬态分析为基础直接在时域中求解电路的周期稳态响应, 求解的方法则是令一个周期起始与终止点中电路的状态一致, 即 $x(0) = x(T)$, 可以通过牛顿迭代法来不断减小误差, 让其小于一定的误差容限 $|x(T) - x(0)| < \xi$ 进而得到一个初始解。假设状态变量为 $x(t)$, 电路方程为 $C\frac{dx(t)}{dt} + Gx(t) = BU(t)$。假设电路激励信号的周期为 T, 当电路达到稳定状态时应满足 $x(0) = x(T)$。为了能够使用牛顿迭代法进行求解, 构造状态转换函数 $\phi(y, t_0, t_1) = x(t_1)$。其含义是, 对于 $x(t_0) = y$ 的情况,t_1 时刻的状态值 $x(t_1)$ 等于 $\phi(y, t_0, t_1)$。进而初值求解问题可以转换成求解下面函数零点的问题:

$$H(x(0)) = \phi(x(0), 0, T) - x(0) \tag{3}$$

使用牛顿迭代法对该函数零点进行求解可以得到迭代公式 $H\left(x^k\right) + J_H\left(x^k\right)\left(x^{k+1} - x^k\right) = 0, J_H\left(x^k\right)\left(x^{k+1} - x^k\right) = -H\left(x^k\right)$。其中 J_H 为 $H(x)$ 的 Jacobi 矩阵:

$$J_H(x) = \frac{\partial H(x)}{\partial x} = \frac{\partial(\phi(x, 0, T)) - x}{\partial x} = \frac{\partial(\phi(x, 0, T))}{\partial x} - I \tag{4}$$

为了求解牛顿迭代法的 Jacobi 矩阵, 采用后向欧拉法中的思路, 首先将整个周期 T 等分为 L 个区间,$\Delta t = \frac{T}{L}$, 此时由后向欧拉可得:

$$C\frac{x_{l+1}-x_l}{\Delta t}+Gx_{l+1}=BU(t_{l+1}) \tag{5}$$

等式两边对 x_l 求导可得:

$$C\frac{\partial x_{l+1}}{\partial x_l}-C\frac{\partial x_l}{\partial x_l}+\Delta tG\frac{\partial x_{l+1}}{\partial x_l}=\Delta t\frac{\partial BU(t_{l+1})}{\partial x_l}=0 \tag{6}$$

整理可得下式, 对任意 $l\in[0,L-1]$ 成立:

$$\frac{\partial x_{l+1}}{\partial x_l}=C(C+\Delta tG)^{-1} \tag{7}$$

因此可得:

$$\frac{\partial(\phi(x,0,T))}{\partial x}=\prod_{l=0}^{L-1}\frac{\partial x_{l+1}}{\partial x_l}=\prod_{l=1}^{L}C\left(C+G_\Delta t\right)^{-1}=\left(C\left(C+G_\Delta t\right)^{-1}\right)^L \tag{8}$$

进而得到 Jacobi 矩阵:

$$J_H(x)=\frac{\partial(\phi(x,0,T))}{\partial x}-I=\left(C\left(C+G_\Delta t\right)^{-1}\right)^L-I \tag{9}$$

因此牛顿迭代公式为:

$$\left(\left(C\left(C+G_\Delta t\right)^{-1}\right)^L-I\right)\left(x^{k+1}-x^k\right)=-H\left(x^k\right)=x^k-\phi(x^k,0,T) \tag{10}$$

迭代求解时先设定初值, 然后根据迭代公式进行迭代, 直到两轮迭代结果小于误差容限且函数值的绝对值也小于误差容限时停止迭代。

得到初始解后可以通过后向欧拉方法根据电路初值求解电路响应，后向欧拉的求解方法将在 2.3节进行介绍。

2.2 牛顿迭代法

牛顿迭代法是求解非线性方程的一种方法, 其最大优点是在方程的单根附近具有平方收敛, 而且该法还可以用来求方程的重根、复根, 此时线性收敛, 但是可通过一些方法变成超线性收敛。其主要思路是通过将非线性方程 $f(x)=0$ 用线性近似 (泰勒展开), 进而得以进行求

解。忽略二次及以上的高阶项后泰勒展开的结果如下:

$$0 = f\left(x^*\right) \cong f\left(x^k\right) + \frac{df}{dx}\left(x^k - x^*\right) \tag{11}$$

其中 x^* 为准确解,x^k 为第 k 次迭代结果, 上式整理后可得如下的迭代公式:

$$x^{k+1} = x^k - \left[\frac{df}{dx}\left(x^k\right)\right]^{-1} f\left(x^k\right) \tag{12}$$

收敛条件为 $\left|\left[\frac{df}{dx}(x)\right]^{-1}\frac{d^2f}{d^2x}(x)\right| \leq L$ 其中 L 为有界的常数, 即 $\frac{df}{dx}(x)$ 不收敛于 $0, \frac{d^2f}{d^2x}(x)$ 不趋于无穷。此外为了保证较好的收敛性还需要一个较好的初值同时需要两个收敛条件来防止错误的收敛:

$$\left\|x^{k+1} - x^k\right\| < \varepsilon_{x_\alpha}, \qquad \left\|f\left(x^{k+1}\right)\right\| < \varepsilon_{f_\alpha} \tag{13}$$

2.3 后向欧拉法 Backward Euler

后向欧拉法为有限差分法的一种, 通过对时间的离散化以及使用差分来逼近微分的方法求解一阶偏微分方程。对于如下的偏微分方程, 使用时间上更早的数据点做差分来逼近梯度, 具体公式如下。

$$\frac{dx(t)}{dt} = Ax(t) \tag{14}$$

$$x(t_0) = x_0 \tag{15}$$

$$\frac{d}{dt}x\left(t_{l+1}\right) = Ax\left(t_{l+1}\right) \cong \frac{x\left(t_{l+1}\right) - x\left(t_l\right)}{\Delta t} \tag{16}$$

$$x\left(t_{l+1}\right) \cong x\left(t_l\right) + \Delta t Ax\left(t_{l+1}\right) \tag{17}$$

公式 1中后向欧拉的迭代公式为:

$$(C + \Delta tG)X(t_{l+1}) = CX(t_l) + \Delta tBU(t_{l+1}) \tag{18}$$

2.4 广义共轭残差法 GCR

广义共轭残差法是求一种解线性方程组的方法, 通过残差来决定每次的探索方向, 每次探索方向之间彼此正交来保证每次搜索时目标函数的最优性。对于线性方程组 $Mx = b$, 广义共轭残差的求解原理如下。　　第 k 轮残差的定义为,$r^k \equiv b - Mx^k$, 第 k 轮迭代的解 x_k 定义为,$x^{k+1} = \sum_{i=0}^{k} \alpha_i w_i$, 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,

$$\min_x \|r^{k+1}\|_2^2 = \|b - \sum_{i=0}^{k} \alpha_i M w_i\| \tag{19}$$

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)

$$f(x) = \frac{1}{2} x^T M x - x^T b \tag{20}$$

$$\nabla_x f(x) = Mx - b \tag{21}$$

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:

$$\alpha_k = \frac{\left(r^k\right)^T (Mp_k)}{(Mp_k)^T (Mp_k)} \tag{22}$$

$$x^{k+1} = x^k + \alpha_k p_k \tag{23}$$

$$r^{k+1} = b - Mx^{k+1} = b - Mx^k - \alpha_k M p_k = r^k - \alpha_k M p_k \tag{24}$$

$$p_{k+1} = r^{k+1} - \sum_{j=0}^{k} \frac{\left(Mr^{k+1}\right)^T (Mp_j)}{(Mp_j)^T (Mp_j)} p_j \tag{25}$$

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。

$$p_{k+1} = r^{k+1} - \sum_{j=0}^{k} \frac{\left(Mr^{k+1}\right)^T (Mp_j)}{(Mp_j)^T (Mp_j)} p_j \Rightarrow p_{k+1} = r^{k+1} - \frac{\left(Mr^{k+1}\right)^T (Mp_k)}{(Mp_k)^T (Mp_k)} p_k \tag{26}$$

3 算法实现

3.1 算法流程

算法流程

- 使用提供的 stamp 动态库对 hsipce 网表进行解析得到相应的矩阵数据, 使用 hspice 进行仿真, 得到相应的波形 lis 文件, 调用 read_data 函数来得到相应 lis 文件中 hspice 仿真的波形结果。(main.m)
- 根据 stamp 解析到的输入信号信息生成相应的激励信号数和相应的信号周期。(GenerateSrc.m)
- 根据函数的输入参数决定 shooting method 的分段数 L 和牛顿迭代法的最大轮数和误差, 并进行初值的求解。(main.m)
- 使用后向欧拉根据初值求解微分方程。(BackwardEuler.m)
- 上面求解过程中需要对线性方程组求解的地方提供了两种求解的方法:matlab 自带的“\”和自己实现的 GCR(Golden.m,GCR.m)
- 将得到的仿真结果通过线性插值得到与 hspice 相同采样时间的数据点, 并计算相应的误差 (最大绝对误差和均方绝对误差)。(main.m)
- 生成结果对比图。(report.m)

3.2 输入激励信号

脉冲源

语法:

Vxxx n+ n- PU<LSE> <(>v1 v2 <td <tr <tf <pw <per> > > > > <)>

Ixxx n+ n- PU<LSE> <(>v1 v2 <td <tr <tf <pw <per> > > > > <)>

其中 v1 为脉冲源的低压,v2 为高压,td 为延迟,tr 为上升时间,tf 为下降时间,pw 为脉冲宽度,per 为信号周期。下图展示了一个脉冲信号源的例子:

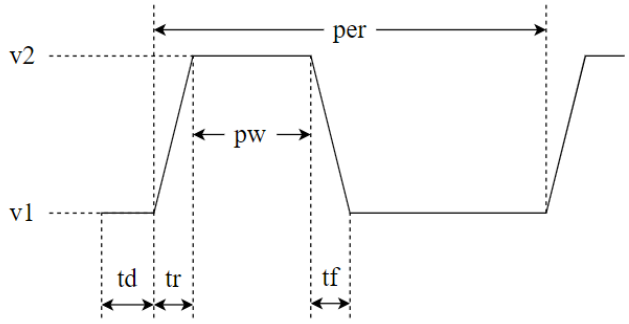

图 1: 脉冲信号源示例

正弦源

语法:

Vxxx n+ n- SIN <(> vo va <freq <td <θ <ϕ»» <)>

Ixxx n+ n- SIN <(> vo va <freq <td <θ <ϕ»» <)>

其中 vo 为直流幅度,va 为交流幅度,freq 为频率,td 为延时,θ 为衰减因子,ϕ 为相位延时。信号的公式可以如下:

$$y(t) = e^{-\theta} \cdot va \cdot sin(2\pi freq \cdot (t - td) - \phi) + vo \tag{27}$$

3.3 输出

为了将仿真结果与 hspice 的结果进行对比, 首先需要通过插值法得到自己仿真结果在 hspice 采样时间上的值, 然后求解最大绝对误差和均方绝对误差。同时为了方便访问结果, 使用 save 命令将仿真的数据存储到.mat 文件中（文件在 Benchmark 目录下，文件命名与图中对应）。对于有多个输出节点的电路, 为了方便观察选取其中一个的波形进行绘制。

3.4 编译与运行

程序基于 Matlab 2015b(32 位) 平台, 将整个解析网表文件, shooting method 过程和结果输出的过程都打包在 main 函数中, 由于所有 benchmark 都在 Benchmark 目录中, 所以需要在 matlab 中将当前目录切换到 Benchmark 下运行。main 函数的接口为 $main(caseName, L, newtonMaxIter, newtonErrorThres, errorThres)$. 其中 caseName(RCL_s3(正弦信号作为激励),RCL_p3(方波信号作为激励),bus8bit8seg(正弦信号作为激励),bus8bit8seg_p(方波信号作为激励),bus32seg16(正弦信号作为激励),bus32seg16_p(方波信号作为激励))。L 为

Shooting 的分段数,newtonMaxIter, newtonErrorThres 为牛顿迭代法中的最大迭代轮数和两个误差容限 (两个误差容限放在一个向量变量中),errorThres 为 GCR 的误差阈值。下面给出一个运行示例:

$$main('RLC_s3', 50, 50, [1e-9, 1e-9], 1e-9) \tag{28}$$

4　结果与分析

4.1　实验结果

4.1.1　RLC 电路

4.1.1.1　正弦激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50, 输入的正弦信号为 $sin(200\pi t)$, 仿真的时间为 0-20ms, 采样时间间隔为 0.004ms。shooting 得到的初值为:x_0=[-0.0000,-0.1699,-0.0693,-0.0733,-0.1267,-0.1300,-0.1675,0.0198,0.0198,0.0153,0.0107]。下图为仿真结果:

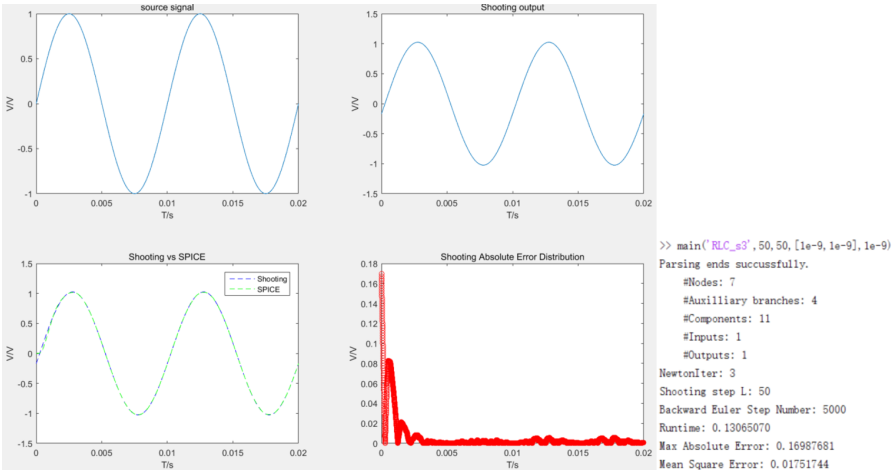

图 2: RLC_s3_golden 仿真结果

从结果中可以看出由于直接使用了 shooting 得到的初值, 在刚开始的一段时间内与 hspice 得到的结果误差较大, 这是由于 hspice 的结果考虑了电路稳态的建立过程, 而基于 shooting 得到的结果则直接得到周期稳态解, 随着时间的推进初始解分量不断衰落, 可以看

出仿真结果与 hspice 仿真结果的误差变小。下图绘制了电路稳定后的一段时间内 (10ms-20ms) 的仿真结果, 也可以印证前面的结论。

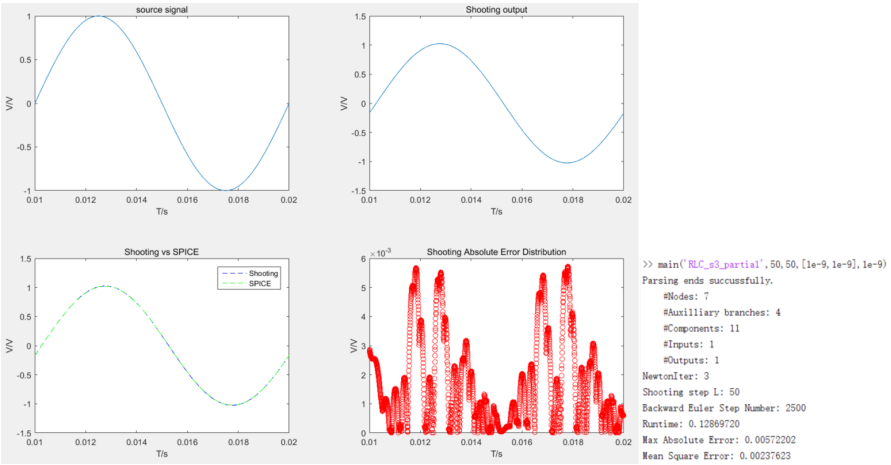

图 3: RLC_s3_partial 仿真结果

4.1.1.2　方波信号激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50。输入的方波信号为周期为 8ms, 高低电平分别为 3V 和 0V, 上升下降时间为 0.5ms, 高电平时间为 3ms, 信号延迟为 1ms。仿真的时间为 0-20ms, 采样时间间隔为 0.004ms。shooting 得到的初值为:x_0=1.0e-03 * [0,0.3100,0.1340,0.1340,0.2373,0.2373,0.3100,-0.0383,-0.0383,-0.0295,-0.0208]。下图为仿真结果, 绝对误差同样也体现了初始解分量的衰减。

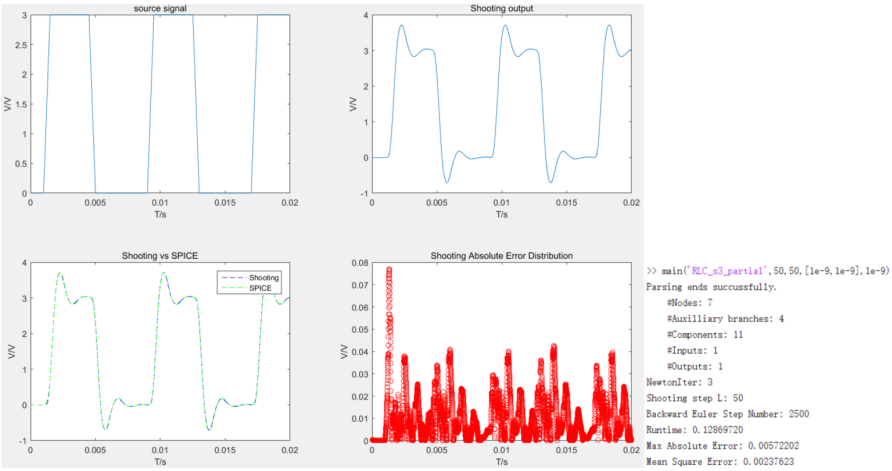

图 4: RLC_p3_golden 实验结果

4.1.2 bus8bit8seg 电路

4.1.2.1 正弦激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50。输入的正弦信号为 $sin(200\pi t)$。仿真的时间为 0-3e-8s, 采样时间间隔为 1e-11s。由于解的维度过大 (有 250 个节点) 只展示前十个初始解 $x_0(1:10)$=[0.0733,-0.0159,-0.0057,-0.0052,-0.0051,-0.0050,-0.0050,-0.0050,-0.0017,-0.0010]。下图为仿真结果:

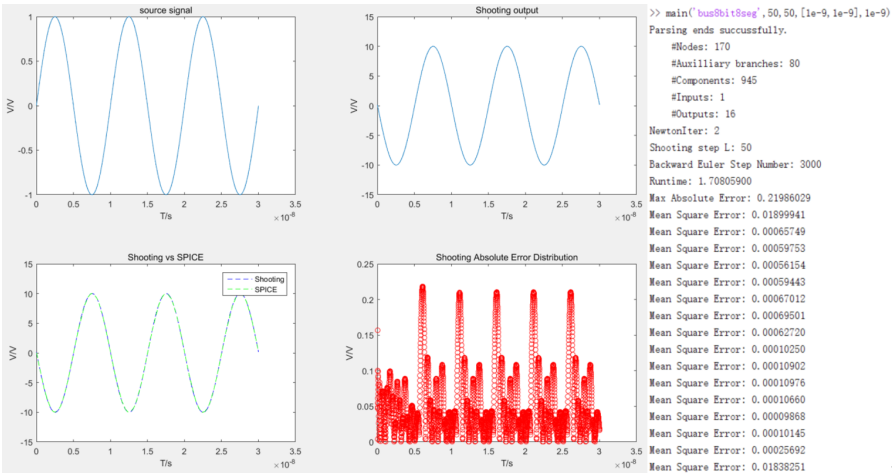

图 5: bus8bit8seg_golden 实验结果

采用自己实现的 GCR 算法进行线性方程组的求解 (误差容限 1e-9) 得到如下的结果 (除了线性方程组的求解方法变化外, 其余部分与上面设置相同)。从结果可以看出使用 GCR 的运行速度比使用 matlab 自带的求解器慢了很多, 整个 shooting 过程的运行时间增加了两个数量级 (1.7s->279.5s), 但是计算得到的误差是相同的, 这说明了实现的 GCR 算法功能是正确的, 但是在计算效率上和 matlab 自带的还有很大的差距。

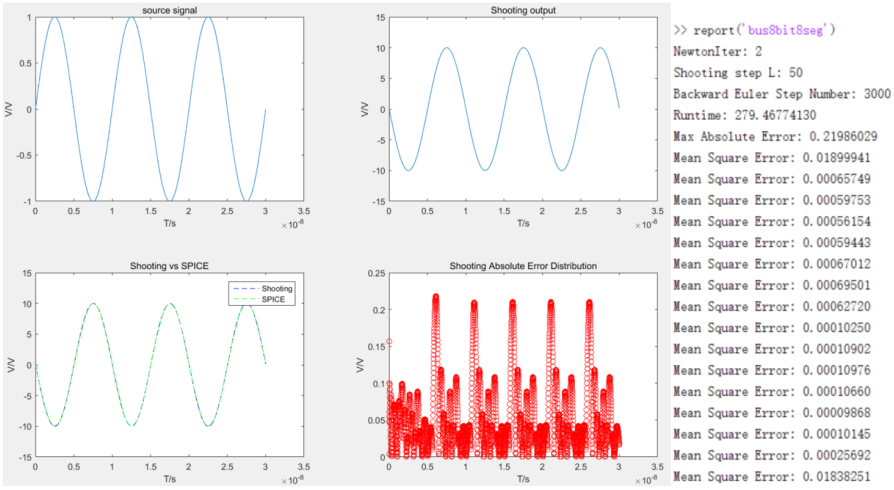

图 6: bus8bit8seg_gcr 实验结果

4.1.2.2　方波信号激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50。输入的方波信号为周期为 1e-8s, 高低电平分别为 3V 和 0V, 上升下降时间为 0.5e-9s, 高电平时间为 4e-9s, 信号延迟为 1e-9s。仿真的时间为 0-3e-8s, 采样时间间隔为 1e-11s。由于解的维度过大 (有 250 个节点) 只展示前十个初始解 $x_0(1:10)$=[0,0,0,0,0,0,0,0,0,0]。下图为仿真结果:

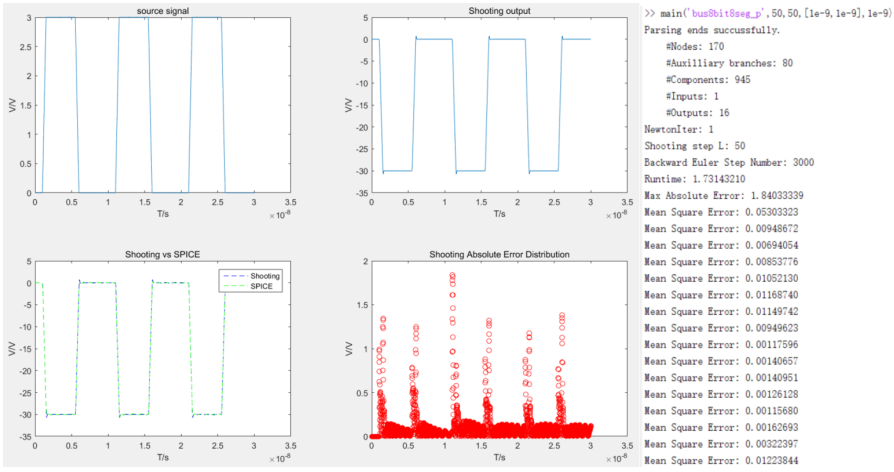

图 7: bus8bit8seg_p_golden 实验结果

采用自己实现的 GCR 算法进行线性方程组的求解 (误差容限为 1e-9) 得到如下的结果 (除了线性方程组的求解方法变化外, 其余部分与上面设置相同)。从结果同样可以看出使用 GCR 的运行速度比使用 matlab 自带的求解器慢了很多, 整个 shooting ing 过程的运行时间

增加了两个数量级 (1.7s->280.1s), 但是计算得到的误差是相同的, 同样说明了实现的 GCR 算法功能是正确的, 但是在计算效率上和 matlab 自带的还有很大的差距。

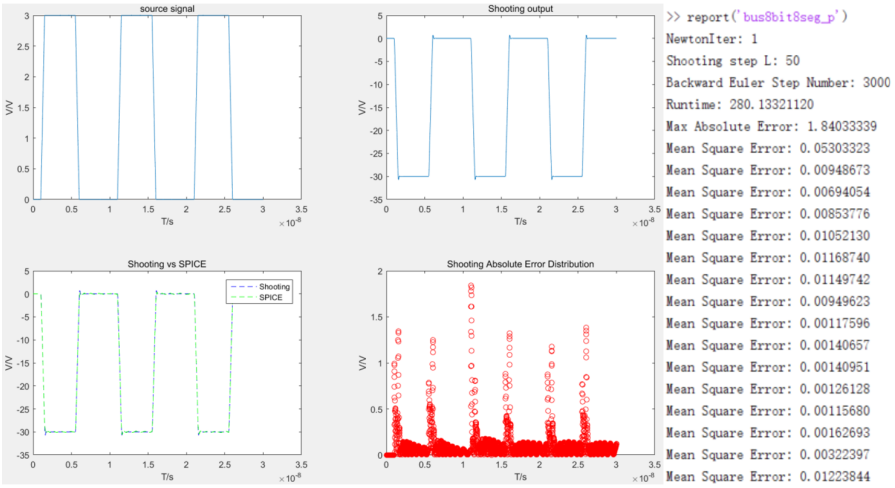

图 8: bus8bit8seg_p_gcr 实验结果

4.1.3 bus32seg16 电路

4.1.3.1 正弦激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50。8 个电流源都输入 $sin(200\pi t)$ 的正弦信号。仿真的时间为 0-3e-8s, 采样时间间隔为 1e-11s。由于解的维度过大 (有 1666 个节点) 只展示前十个初始解 $x_0(1:10)$=[0.3963,0.3777,0.3771,0.3770,0.3768,0.3766,0.3767,0.3948,-0.1089,-0.1092]。下图为仿真结果:

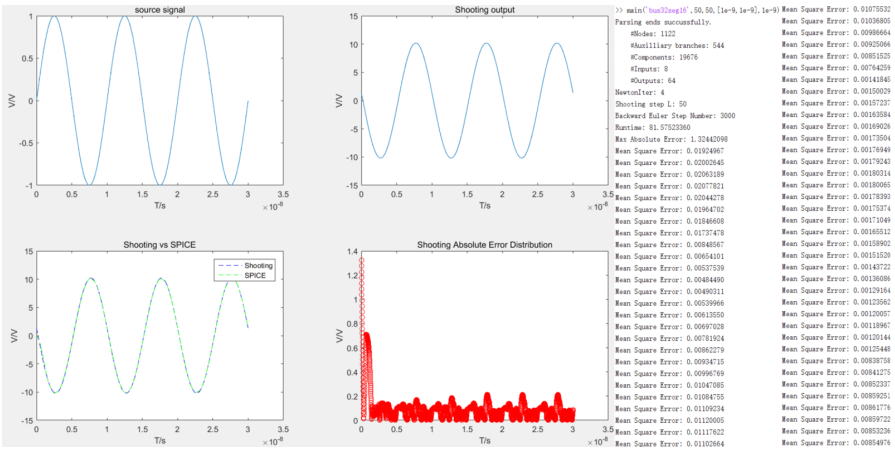

图 9: bus32seg16_golden 实验结果

4.1.3.2　方波信号激励

仿真设置: 牛顿迭代法的最大迭代轮数为 50 轮, 同时两个误差容限都为 $1e^{-9}$, 采用 matlab 自带线性方程组求解方法,shooting 的分段数量为 50。8 个电流源都输入方波信号, 详细设置: 周期为 1e-8s, 高低电平分别为 3V 和 0V, 上升下降时间为 0.5e-9s, 高电平时间为 4e-9s, 信号延迟为 1e-9s。仿真的时间为 0-3e-8s, 采样时间间隔为 1e-11s。由于解的维度过大 (有 1666 个节点) 只展示前十个初始解 $x_0(1:10)$=[0,0,0,0,0,0,0,0,0,0]。下图为仿真结果:

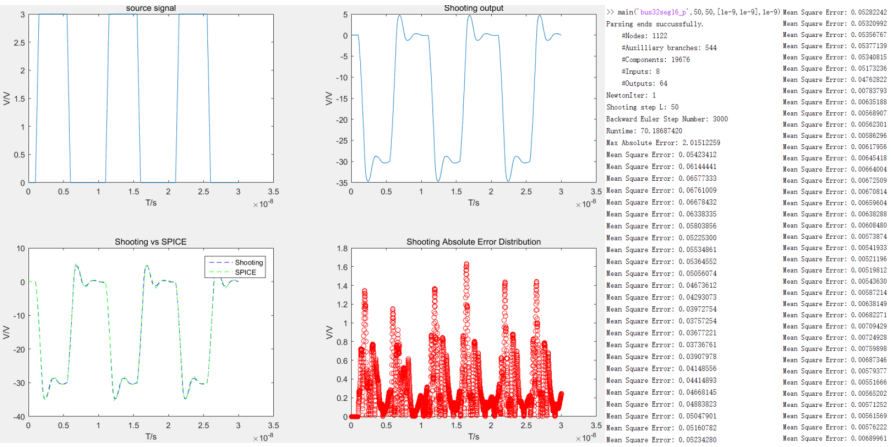

图 10: bus32seg16_p_golden 实验结果

4.2　结果分析

从结果中可以看出对于输入正弦激励信号的情况, 由于直接使用了 shooting 得到的初值, 在刚开始的一段时间内与 hspice 得到的结果误差较大, 这是由于 hspice 的结果考虑了电路稳态的建立过程, 而基于 shooting 得到的结果则直接得到周期稳态解, 随着时间的推进初始解分量不断衰落, 可以看出仿真结果与 hspice 仿真结果的误差变小。但是对于输入方波的情况刚输入的误差并不大, 分析原因为方波信号有一定的延时, 故起始的输出也都为 0, 和 hspice 接近。

此外对比了使用自己实现的 GCR 算法 matlab 自带“反除操作 \”的进行线性方程组的求解。从结果可以看出使用 GCR 的运行速度比使用 matlab 自带的求解器慢了很多, 整个 shooting 过程的运行时间增加了两个数量级, 但是计算得到的误差是相同的。这说明了实现的 GCR 算法功能是正确的, 但是在计算效率上和 matlab 自带的还有很大的差距。

5 文件结构

- main.m: 包含整个 shooting method 求解初值和基于初值求解电路方程响应的框架和流程。
- BackwardEuler.m: 实现后向欧拉求解法。
- GCR.m: 实现 GCR 算法。
- Golden.m: 调用 matlab 自带的反除操作 (“\”用于线性方程)。
- GenerateSrc.m: 生成激励信号和并返回信号周期。
- report.m: 分析数据并绘制对比图。