1. 实验内容

结点电压法是电路的系统分析方法之一，其结点电压方程变量数少，适合任意结构的电路。本实验将编写一个可视化电路分析软件，其能够利用结点电压法，求解给定的包含电阻、（受控）电压/电流源电路的所有结点电压，并在此基础上验证叠加定理和求解戴维宁等效电路。

2. 程序设计

2.1 开发环境

本程序采用 python 编写，使用 sympy 库进行代数运算及方程求解，使用 pygame 库进行 UI 界面的开发。其中，python 版本为 3.7.2，sympy 版本为 1.6.2，pygame 版本为 2.0.0。

2.2 项目结构

```
.
├── app.py
├── core
│   ├── __init__.py
│   ├── analyzer.py
│   ├── common.py
│   └── element.py
└── ui
    ├── __init__.py
    ├── cmd.py
    ├── common.py
    ├── component.py
    ├── logger.py
    ├── run.py
    ├── sidebar.py
    ├── widget.py
    └── assets/
```

程序共分为三大部分：核心逻辑模块(core)，用户界面模块(ui)，启动器(app.py)

对于核心逻辑模块，common.py 为公用代码，内含一些常用工具类及函数；element.py 为电路元件类的实现，包括电阻(Resistance)、独立电流源(CurrentSource)、独立电压源(VoltageSource)、CCCS、VCCS、CCVS 及 VCVS；analyzer.py 为电路分析逻辑，即对于传入的结构化电路数据进行结点电压的求解。

对于用户界面模块，assets 文件夹内包含该模块所需的所有静态资源，如字体文件、图标等；common.py 为公用代码；component.py 中包含所有电路元件控件类的实现，其可以在 UI 中直接渲染，注意与 core/element.py 中元件类的区分；widget.py 为一些窗体控件基类；cmd.py, logger.py, sidebar.py 为窗体控件的再封装，可直接在 UI 中创建使用，其中 cmd.py 还拥有处理用户命令的功能，掌握着程序的大部分运行逻辑；run.py 为界面运行主循环，起到 UI 启动器的作用。

启动器负责两个模块的调度，以及模块间的数据传输和数据结构化整理。

2.3 设计分析

2.3.1 核心逻辑

（1）数据输入

给定电路为结构化数据。具体地，输入包含结点总数 `node_count` 和一个元件列表 `elements`。对于元件列表中的每个元件，包含公共属性：元件类型 `name`，元件 ID `eid`，正极结点编号 `pos_id`，负极结点编号 `neg_id`；以及不同元件子类的参数，如电阻阻值 `R`、电压源的电压 `U`、受控源的控制系数 `A` 等。需要说明的是，所有元件 ID `eid` 应为和元件列表长度相同的一个排列（从 0 开始），并且所有结点编号应在 0 到 `node_count - 1` 闭区间之内，其中 0 号结点作为接地结点，其结点电压为 0V。

（2）结点电压求解

在用结点电压法求解电路时，首先应为每个非接地结点列写方程，共 `node_count - 1` 个：

$$\sum_{j=1}^{\mathrm{n}-1} G_{ij}\, u_j = i_{Si}$$

其中 $i = 1 \ldots n-1$。G_{ii} 表示结点 i 的自导，其值等于该结点连于相邻各结点支路电导之和；G_{ij} 表示结点 i 到结点 j 的互导 $(i \neq j)$，其值等于连接两结点支路电导的负值。i_{Si} 表示结点 i 的注入电流之和。

在处理独立电压源、CCVS 和 VCVS 时，设其流过的电流为 i，并附加方程：

$$u_{\mathrm{pos\_id}} - u_{\mathrm{neg\_id}} = U_s$$

其中，对于独立电流源来说，U_s 为给定电压值常数；对于 CCVS，有 $U_s = A \cdot i_{\text{cid}}$；对于 VCVS，有 $U_s = A \cdot (u_{\text{ct\_pos\_id}} - u_{\text{ct\_neg\_id}})$。

在处理 CCCS 和 VCCS 时，设其流过的电流为 i，并附加方程：

$$i = I_s$$

其中，对于 CCCS，有 $I_s = A \cdot i_{\text{cid}}$；对于 VCCS，有 $I_s = A \cdot (u_{\text{ct\_pos\_id}} - u_{\text{ct\_neg\_id}})$。

为减少编程中对于特殊细节的判断及处理受控源存在的情况，在处理电阻时，设其流过的电流为 i，并由 VCR 附加方程：

$$iR = u_{\text{pos\_id}} - u_{\text{neg\_id}}$$

全部方程列写完成之后，若电路连接正确，则显然方程数和变量数相等，此时直接使用 sympy 库解方程组即可。

需要注意的是，电阻支路上的电流不应在列写结点电压方程时，被计入结点的注入电流之中。另外，为方便起见，所有元件的电流和电压均为关联参考方向，在输入电路时应特别注意。

（3）叠加定理验证

首先对于原电路求解结点电压，记录总响应结果。然后将电路中所有独立源置零，然后依次单独恢复某一独立源的参数，求解此时的结点电压并记录该独立响应的结果。最后将所有独立响应叠加，检查是否与总响应相同。

（4）戴维宁等效电路求解

采用分别外加两个不同电流源的方法求解。设两次外加的电流源电流值为 I_1 和 I_2，求解结点电压得到的端口电压分别为 U_1 和 U_2。由戴维宁定理及 KVL 可得以下方程：

$$\begin{cases} U_1 = I_1 R_{eq} + U_{oc} \\ U_2 = I_2 R_{eq} + U_{oc} \end{cases}$$

联立解得开路电压和等效电阻：

$$\begin{cases} U_{oc} = \dfrac{I_1 U_2 - I_2 U_1}{I_1 - I_2} \\ R_{eq} = \dfrac{U_1 - U_2}{I_1 - I_2} \end{cases}$$

即为所求戴维宁等效电路。

2.3.2 用户界面

用户界面使用 pygame 库编写，其优势在于绘图较为方便和灵活。

界面共划分为四个区域：占绝大部分面积的电路绘制板，下方第一行的命令输入框，下方第二行的信息提示框，以及右侧的结果显示框。用户需要在绘图区内完成电路的绘制，同时在命令输入框中输入指令予以辅助；当电路绘制完成后，输入计算指令即可在右侧显示框中得到结果。

3. 使用方法

3.1 操作说明

3.1.1 命令行

操作键盘即可在命令行中输入命令。按上下方向键 UP DOWN 可以快速浏览并输入历史命令；按 DELETE 键可清空当前已输入命令；按 ENTER 键以发送指令。

3.1.2 电路绘制

运行命令 `select|se <r|cs|vs|cccs|vccs|ccvs|vcvs|w>` 以选择当前元件类型。其中 `se` 为命令 `select` 的缩写，两者使用时等价。
`r|cs|vs|cccs|vccs|ccvs|vcvs|w` 为元件类型，分别表示：电阻、独立电流源、独立电压源、电流控制电流源、电压控制电流源、电流控制电压源、电压控制电压源、导线。

运行命令 `direction|dc <u|d|l|r>` 以选择元件摆放方向。`u|d|l|r` 分别表示上下左右。

鼠标左键单击绘制区的灰色格点，将当前已选元件按照当前设置的方向摆放。注意，鼠标点击的格点处将为元件的正极。当摆放导线时，操作略有不同：用户需要先点击一个非导线元件的某一端点以开始导线的铺设，之后在绘制区内依次点击

若干个未占用格点以摆放导线，最终再点击一个非导线元件的端点以结束导线铺设。注意，在导线铺设过程中，你无法进行任何命令行操作。

运行命令 number|num [<element_id> <pos|neg>] 以给所有结点编号。若添加可选参数，则表示将编号为 <element_id> 的元件的正极（输入 pos 时）或负极（输入 neg 时）置为 0 号结点。注意，当你运行完此命令后，如若再进行放置元件的操作，之前的编号将会失效并不再显示。

运行命令 set <element_id> <arg>... 以设置元件参数。其中 <element_id> 为元件编号，<arg>... 为该元件对应的参数。如元件类型为电阻/独立电流源/独立电压源时，应输入 set <element_id> <arg>，分别表示元件编号、元件参数（电阻/电流/电压值）；元件类型为 CCCS/CCVS 时，应输入 set <element_id> <A> <cid>，分别表示元件编号、控制系数、控制电流所在元件编号；元件类型为 VCCS/VCVS 时，应输入 set <element_id> <A> <ct_pos_id> <ct_neg_id>，分别表示元件编号、控制系数、控制电压所在元件的正极/负极结点编号。

运行命令 clear 以清空绘制区。

3.1.3 电路分析

（1）结点电压求解

运行命令 run 以求解所有结点电压，其结果将会显示在右边的文本框中，并存储在内存中。注意，该命令只能在结点已编号且有效的情况下使用。

（2）叠加定理验证

首先绘制原电路，运行命令 run 并记录初始总结果的编号（例如结果框标题为“Result #3:”，则该结果编号为 3）

然后将所有独立源用 set 命令置零，然后依次对于每个独立源，进行以下操作：用 set 命令恢复当前独立源的参数，运行命令 run，并记下当前独立结果的编号。

运行命令 superposition|sp <result_id>... 以计算若干个结果的叠加，叠加结果将显示在右边文本框中。其中 <result_id>... 为之前记录的所有独立结果编号，用空格分隔。

运行命令 print|pr <result_id>，可查看编号 <result_id> 对应的结果。此时应当令 <result_id> 为初始总结果的编号，以验证叠加响应是否和原响应相同。

（3）戴维宁等效电路求解

运行命令 `thevenin|th <pos_id> <neg_id> [<i_1> <i_2>]`，以求解戴维宁等效电路，其开路电压 U_{oc} 及等效电阻 R_{eq} 将会显示在右边文本框中。其中，`<pos_id>` 和 `<neg_id>` 为正/负极端子的结点编号。有可选项 `<i_1>` 和 `<i_2>`，代表求解时所用的外加电流源的电流值，默认为 1A 和 2A。注意，该命令只能在结点已编号且有效的情况下使用。

3.1.4 其他命令

运行命令 `exit|quit` 以退出程序。

运行命令 `about` 以查看关于信息。

运行命令 `debug|dbg <args>`，将打印出 `eval(<args>)` 的结果，用于开发调试。

3.2 操作示例

3.2.1 结点电压求解

首先搭建电路大致结构，执行以下命令与操作：

注：`(cmd)` 字样为命令提示符，不在命令行中输入；不带提示符的文字部分为鼠标操作，同样不在命令行中输入。此事项之后不再进行说明。

```
(cmd) se r
放置横向的两个电阻
(cmd) dc d
放置纵向电阻
(cmd) se vs
放置独立电压源
(cmd) se vccs
放置 VCCS
(cmd) se cs
(cmd) dc r
放置独立电流源
(cmd) se w
铺设导线
```

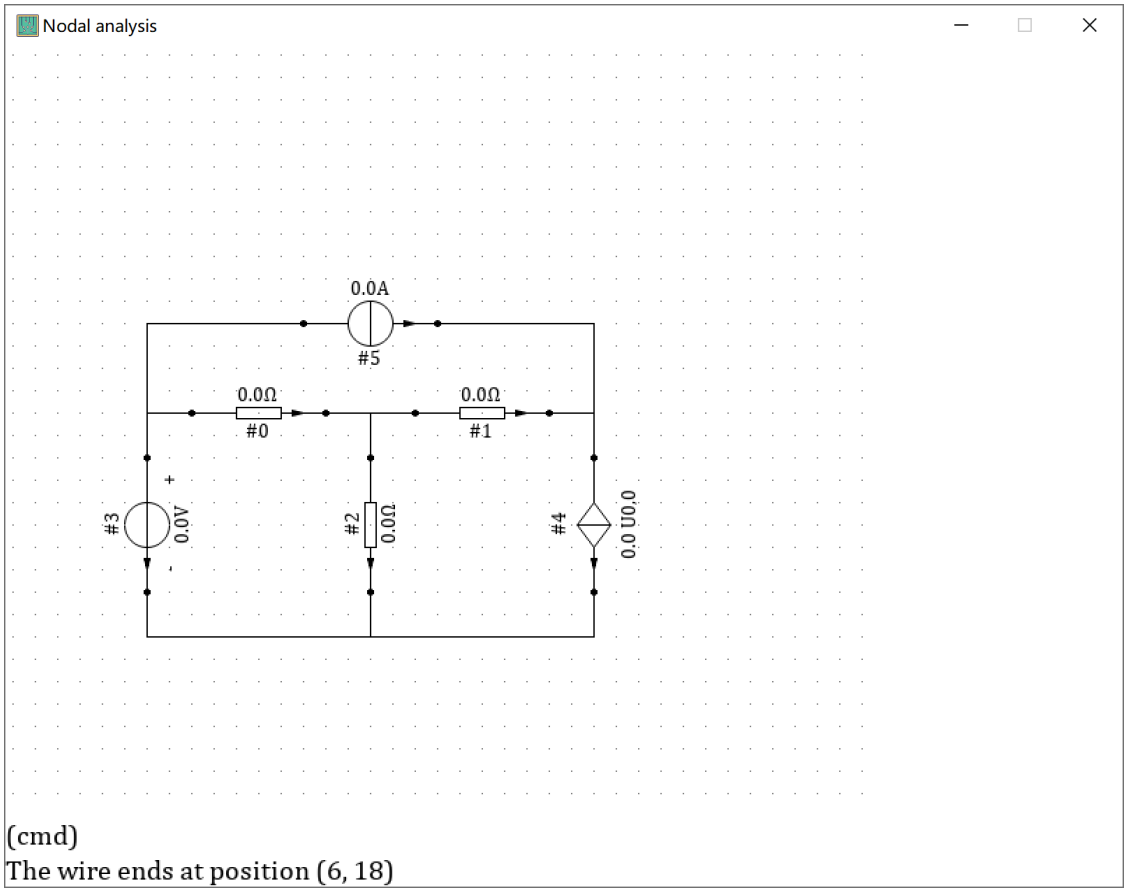

图 1-1 求解结点电压：电路结构搭建完成

然后编号并设置元件参数：

```
(cmd) num 2 neg
(cmd) set 0 24
(cmd) set 1 8
(cmd) set 2 32
(cmd) set 3 20
(cmd) set 4 0.05 1 0
(cmd) set 5 0.15
```

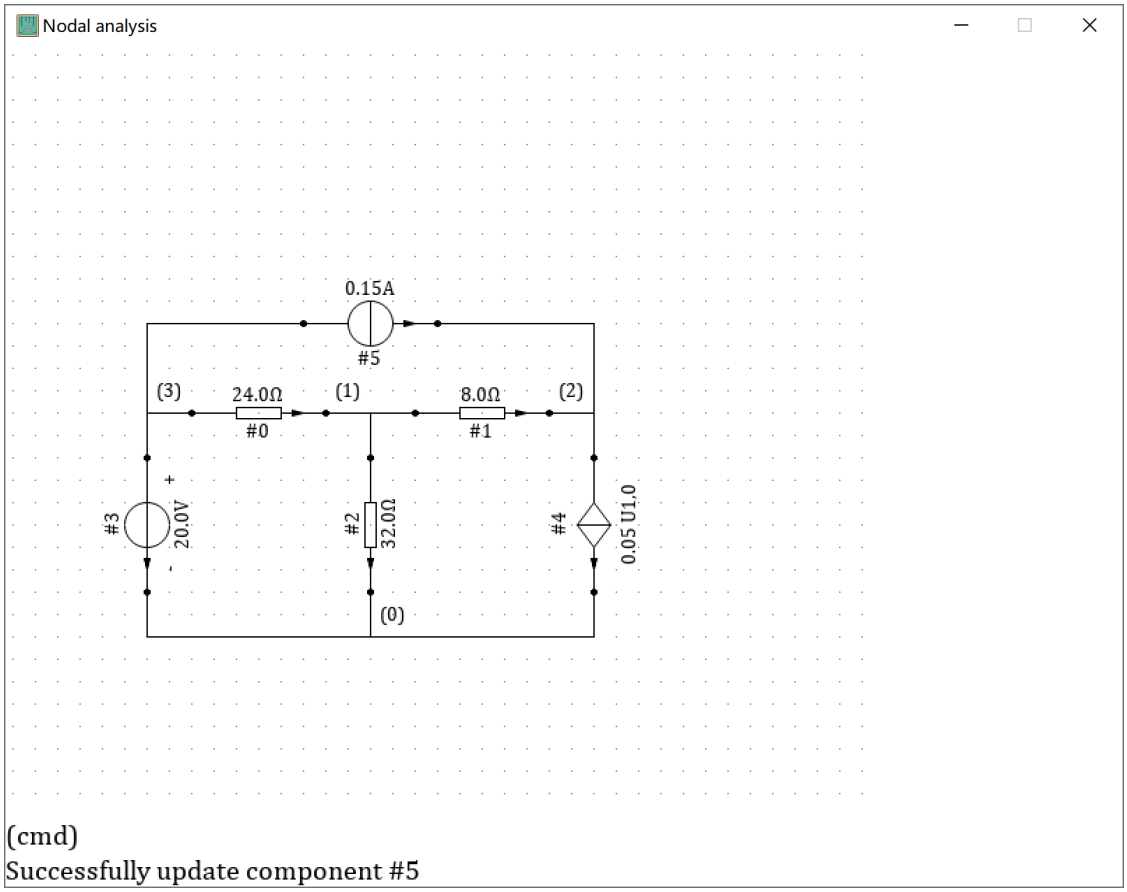

图 1-2 求解结点电压：元件参数设置完成

最后运行计算结点电压命令：

```
(cmd) run
```

显示计算结果如图：

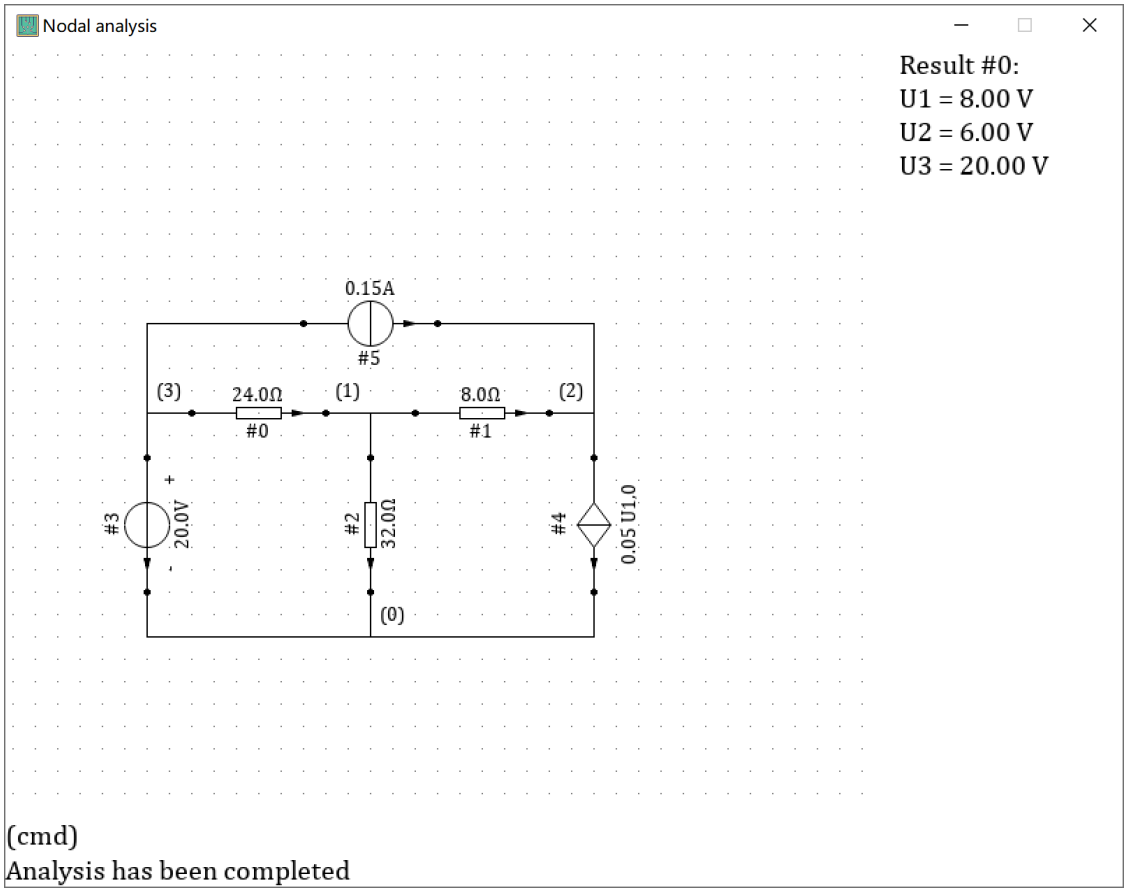

图 1-3 求解结点电压：结点电压计算完成

3.2.2 叠加定理验证

首先搭建电路结构：

```
(cmd) se vs
(cmd) dc d
放置独立电流源
(cmd) se r
放置纵向电阻
(cmd) dc r
放置横向电阻
(cmd) se cccs
(cmd) dc l
放置 CCCS
(cmd) se cs
(cmd) dc u
放置独立电流源
(cmd) se w
铺设导线
```

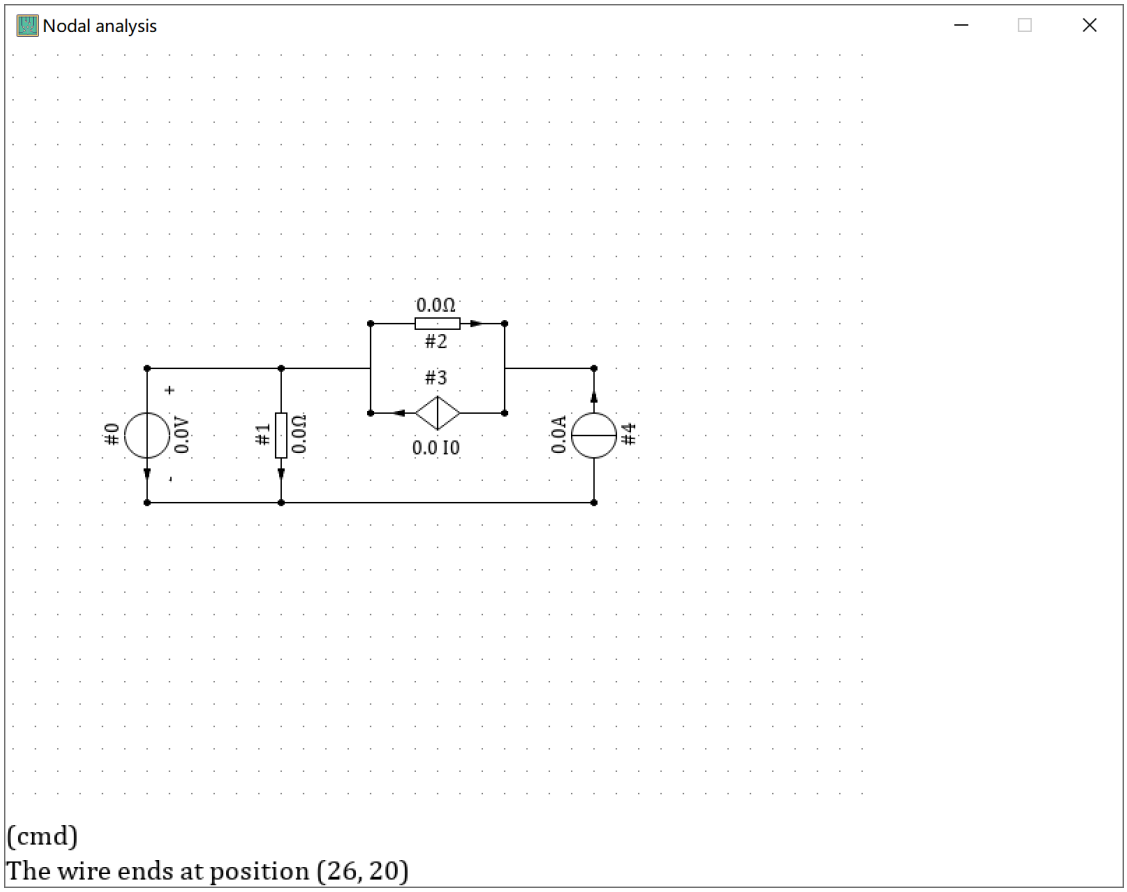

图 2-1 验证叠加定理：电路结构搭建完成

然后编号并设置元件参数：

```
(cmd) num
(cmd) set 0 2
(cmd) set 1 4
(cmd) set 2 3
(cmd) set 3 2 1
(cmd) set 4 3
```

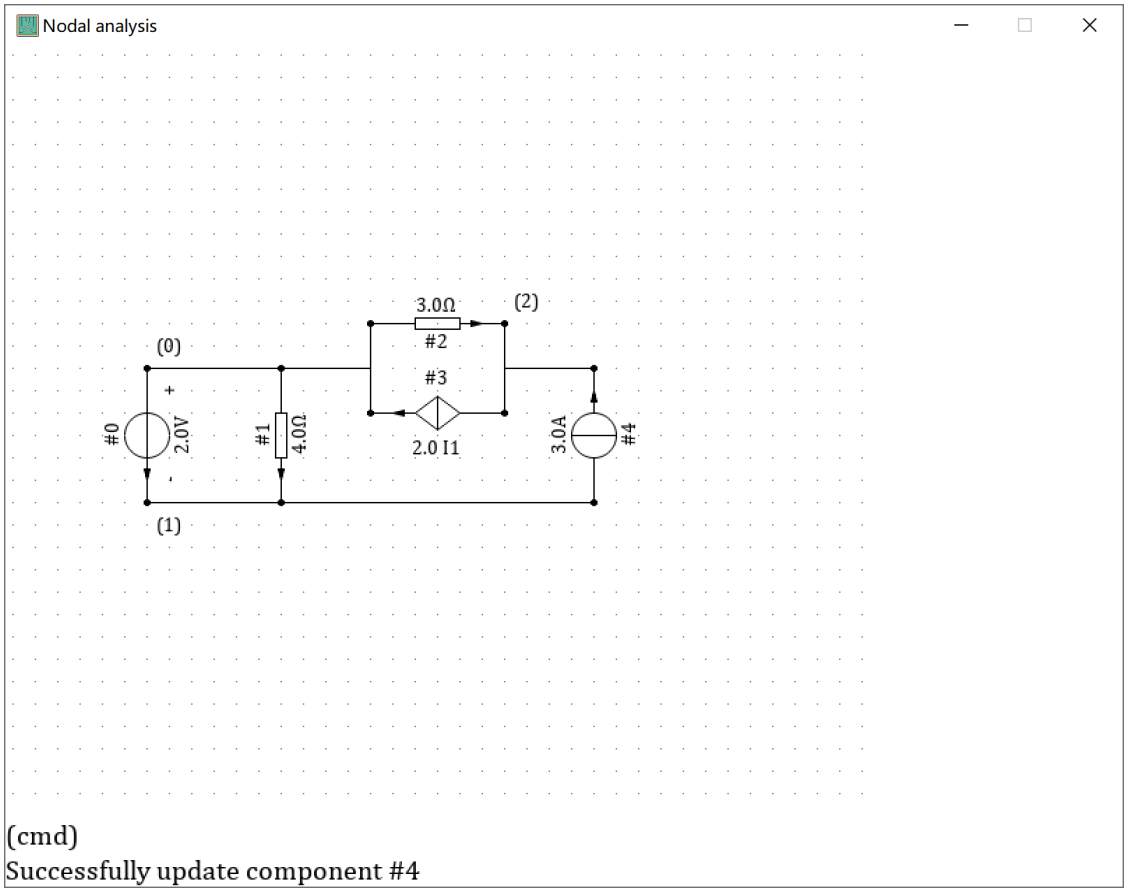

图 2-2 验证叠加定理：元件参数设置完成

先计算共同作用时的结点电压：

```
(cmd) run
```

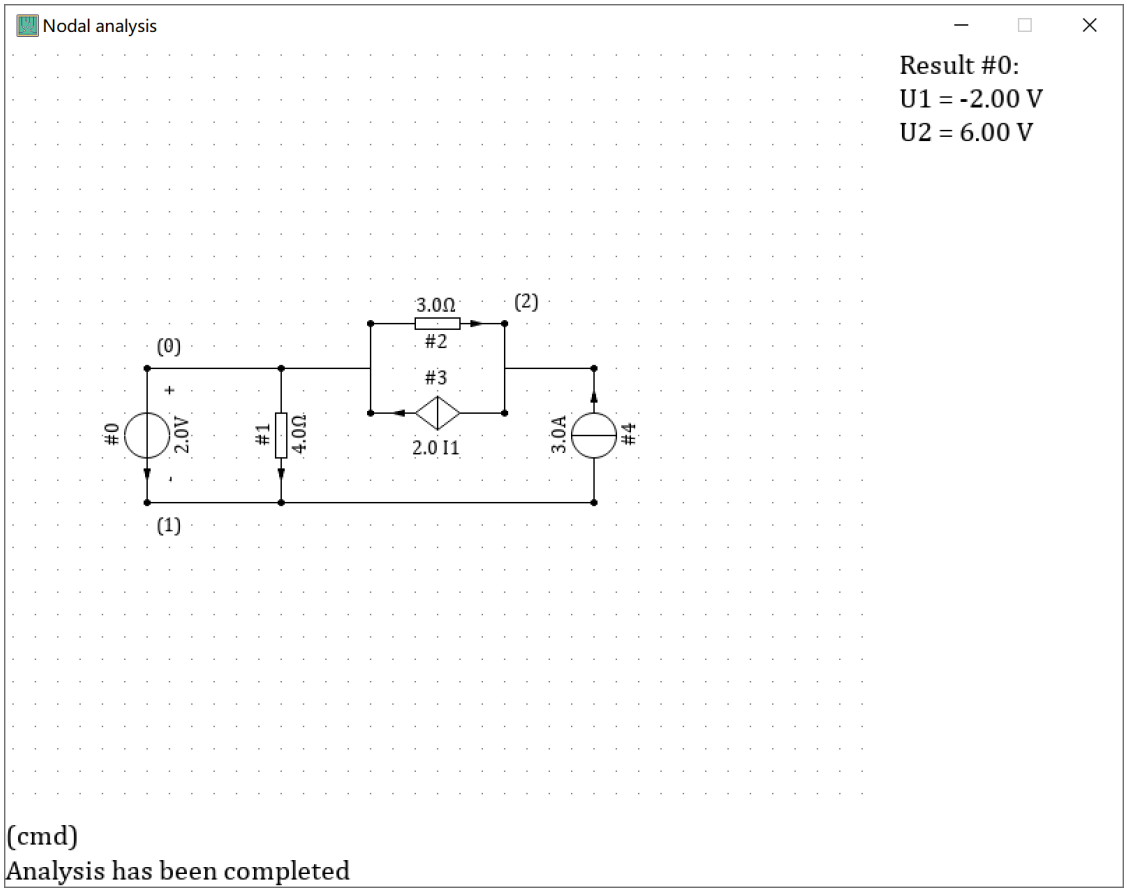

图 2-3 验证叠加定理：共同作用时的计算结果

然后先让独立电压源单独作用，即将独立电流源置零，并计算结点电压：

```
(cmd) set 4 0
(cmd) run
```

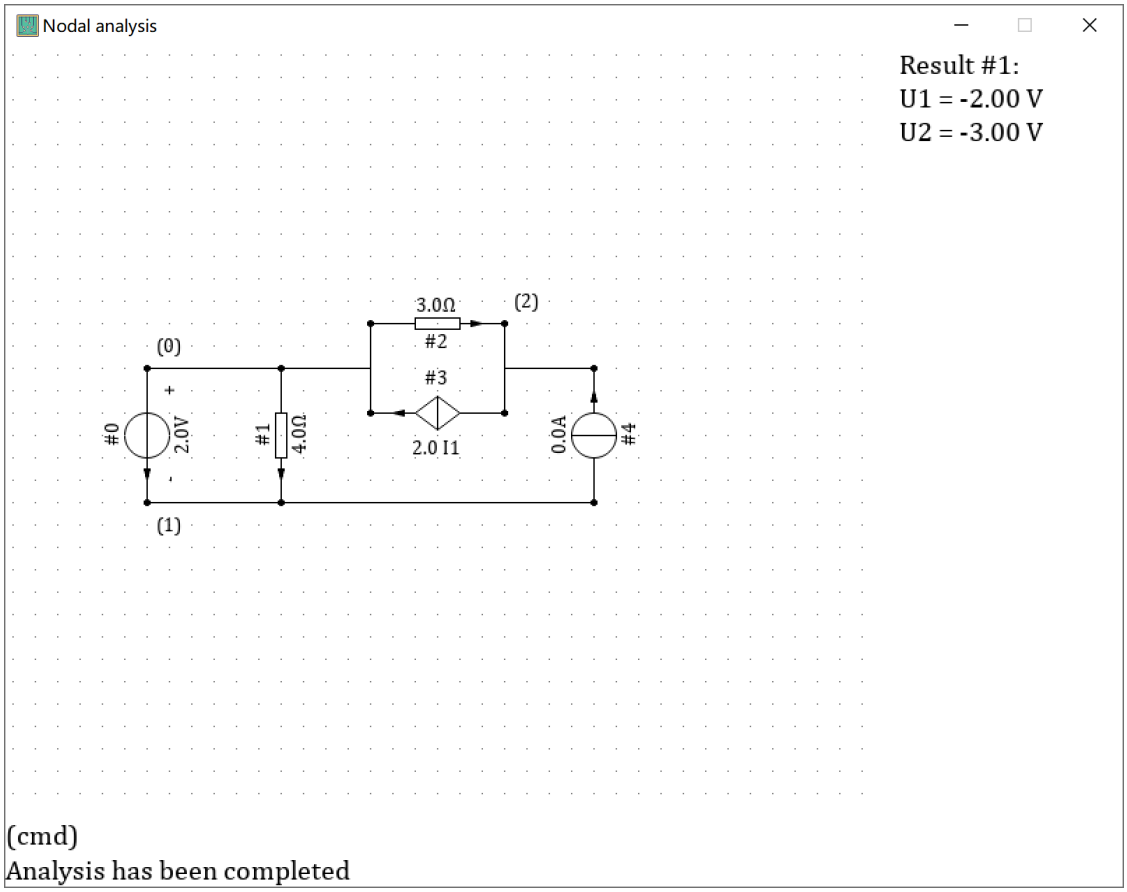

图 2-4 验证叠加定理：电压源单独作用时的计算结果

恢复独立电流源的电流值，并将独立电压源置零，计算独立电流源单独作用时的结点电压：

```
(cmd) set 4 3
(cmd) set 0 0
(cmd) run
```

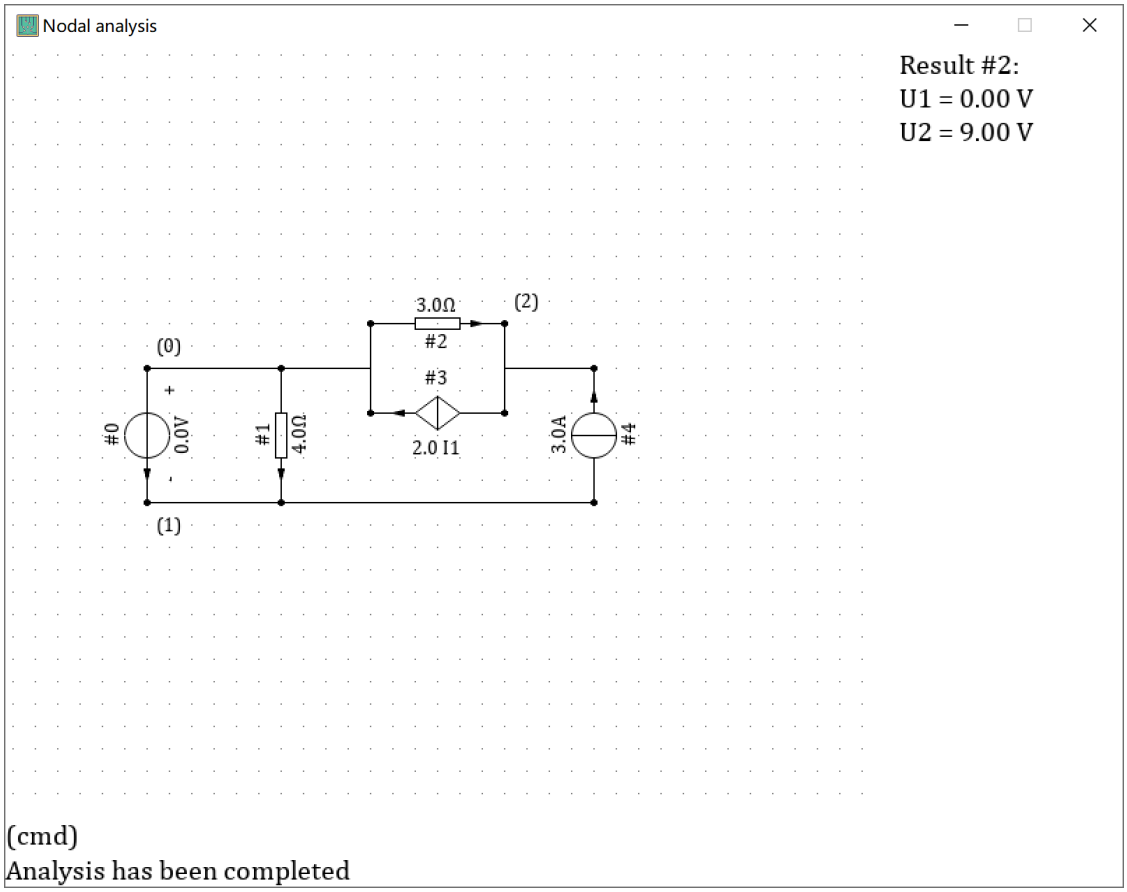

图 2-5 验证叠加定理：电流源单独作用时的计算结果

将两次单独作用的结果叠加：

```
(cmd) sp 1 2
```

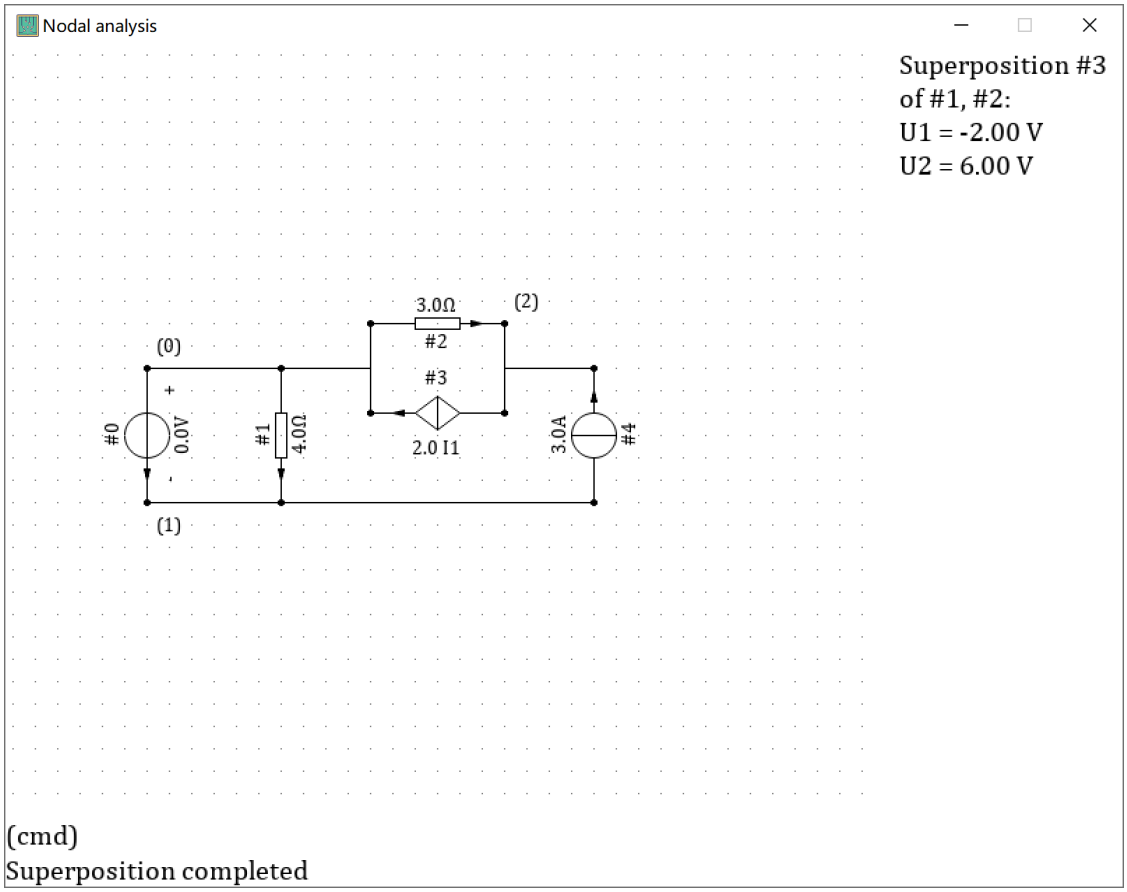

图 2-6 验证叠加定理：叠加后的结果

查看刚开始计算出的两个独立源共同作用时的结果，与当前结果比较：

```
(cmd) pr 0
```

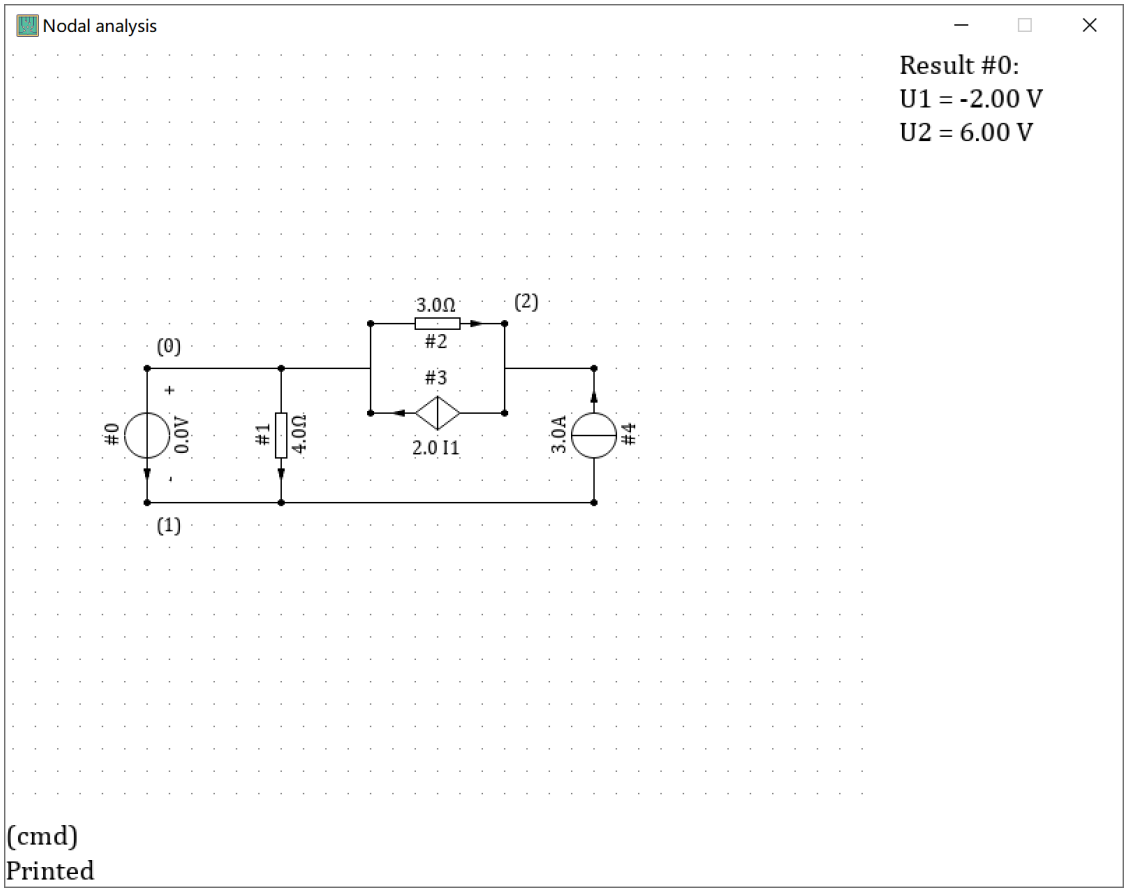

图 2-7 验证叠加定理：查看原始结果

发现两次的结点电压结果相同，即在一定程度上成功验证了叠加定理。

3.2.3 戴维宁等效电路求解

首先搭建电路结构：

```
(cmd) se cs
(cmd) dc u
放置独立电流源
(cmd) se r
(cmd) dc d
放置两个纵向电阻
(cmd) dc r
放置横向电阻
(cmd) se vccs
(cmd) dc l
放置 VCCS
```

```
(cmd) se w
铺设导线
```

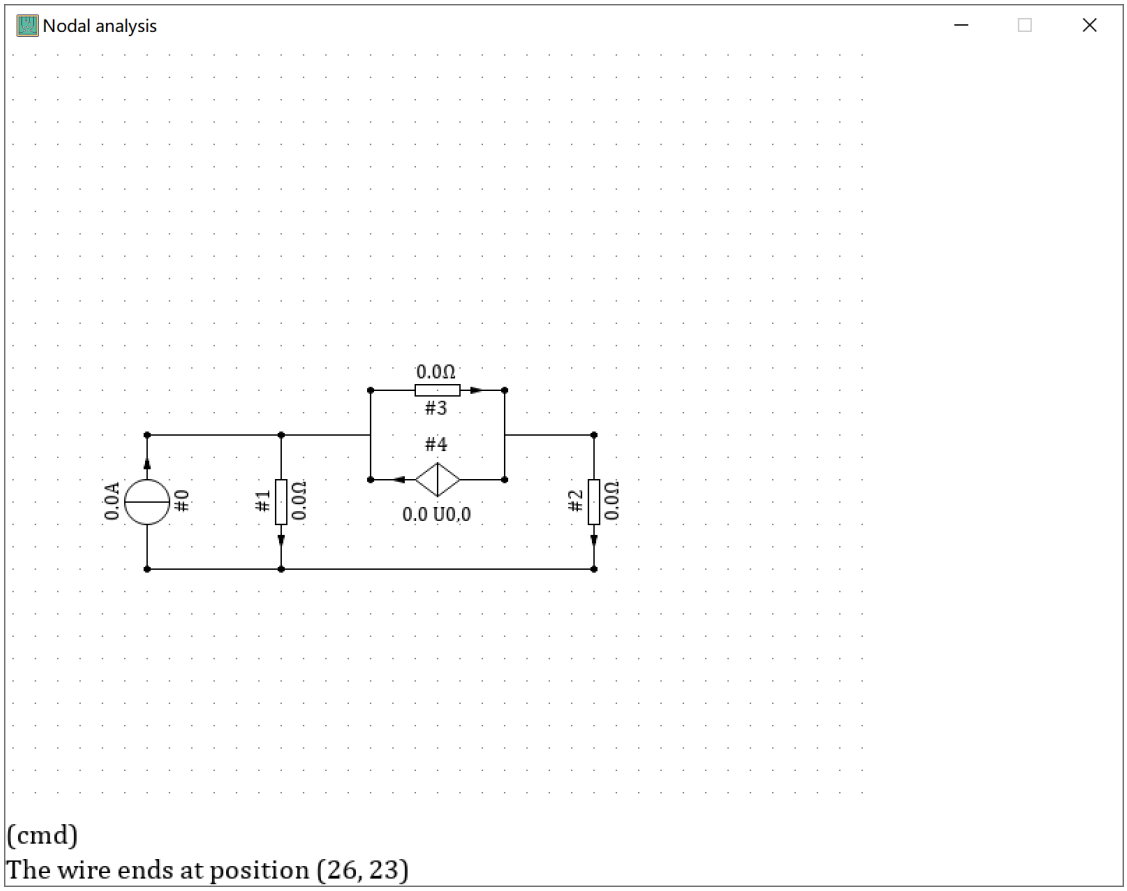

图 3-1 戴维宁等效电路：电路结构搭建完成

然后编号并设置元件参数：

```
(cmd) num
(cmd) set 0 4
(cmd) set 1 8
(cmd) set 2 5
(cmd) set 3 2
(cmd) set 4 2 1 0
```

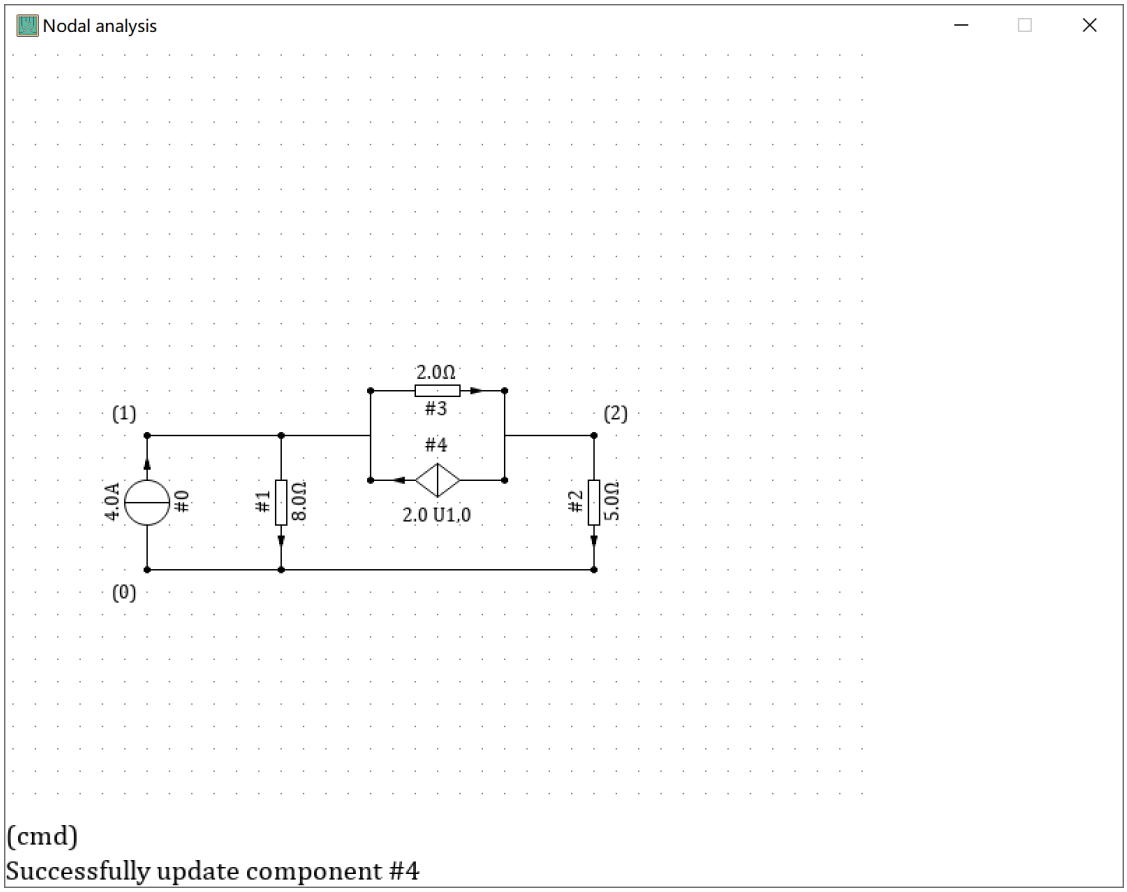

图 3-2 戴维宁等效电路：元件参数设置完成

最后计算 2-0 端口的戴维宁等效电路：

```
(cmd) th 2 0
```

显示计算结果如下：

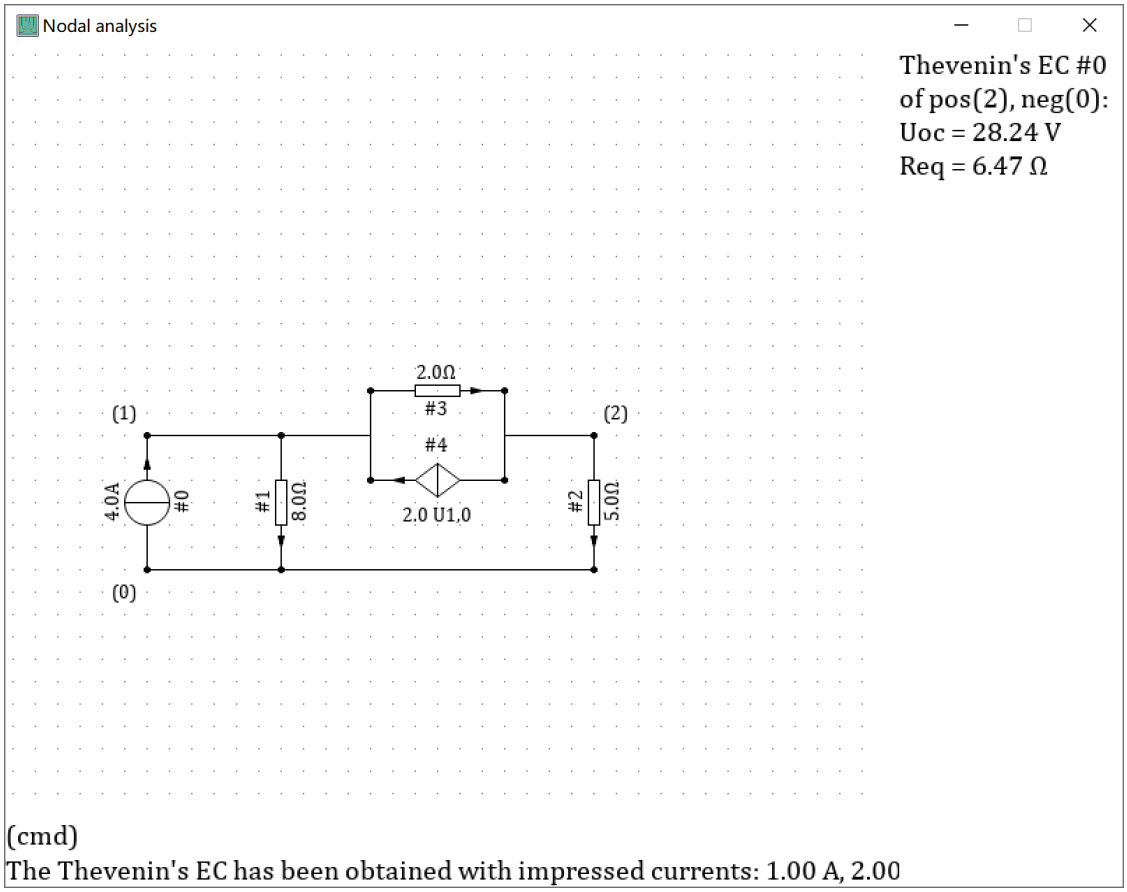

图 3-3 戴维宁等效电路：戴维宁等效电路计算完成

4. 总结

在本次实验中，我们选择 python 作为编程语言，利用 sympy 和 pygame 库，编写了一个可视化电路分析软件，其能够利用结点电压法，求解给定的包含电阻、（受控）电压/电流源电路的所有结点电压，并在此基础上验证叠加定理和求解戴维宁等效电路。

通过本次实验，我对于结点电压法这一电路系统分析方法有了更深刻的学习和认识，彻底理解了结点电压法的基本原理，明白了有独立电压源及受控源存在时如何通过新增未知量并附加方程的方法求解结点电压，更加透彻地理解了叠加定理和戴维宁定理，并清楚了如何通过两次外加电流法求解给定电路的戴维宁等效电路。另一方面，我对于使用 python 编写可视化程序以及 sympy 和 pygame 库使用方法

更加熟悉并能够灵活掌握运用，这对于我今后在计算机专业上的学习研究将有着很大的帮助。

本次实验中编写的程序还有着诸多不足之处。如在用户界面中大部分操作通过用户输入命令行来实现，虽然这种方案具有较强的灵活性和可扩展性且代码易于实现，但用户友好程度较低；又如当用户使用搭建错误的电路进行计算命令时，程序不能给出电路出错的具体原因。这些问题是我们今后将要改进的方向，并预计在下一次的电路编程实验中使之尽可能地得到解决。

5. 参考文献

[1] 应柏青,陈欣琰,王仲奕,王曙鸿,罗先觉. 电路开放实验的实践——电阻网络有限可测的故障诊断[J]. 高校实验室工作研究,2013,115(1),41-43.

[2] 邱关源,罗先觉. 电路[M]. 第 5 版. 北京:高等教育出版社,2006.

[3] 刘崇新,罗先觉. 电路（第 5 版）学习指导与习题分析[M]. 北京:高等教育出版社,2006.

[3] Wikipedia. Nodal analysis - Wikipedia[EB/OL]. (2020 - 03 - 26) [2020 - 11 - 15]. https://en.wikipedia.org/wiki/Nodal_analysis

[4] Khan Academy. Node voltage method (article) | Khan Academy[EB/OL]. [2020 - 11 - 15]. https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-dc-circuit-analysis/a/ee-node-voltage-method

[5] SymPy. Welcome to SymPy’s documentation! — SymPy 1.6.2 documentation[EB/OL]. (2020 - 08 - 09) [2020 - 11 - 15]. https://www.pygame.org/docs/

[6] Pygame. Pygame Front Page — pygame v2.0.1.dev1 documentation[EB/OL]. (2020 - 11 - 12) [2020 - 11 - 15]. https://docs.sympy.org/latest/index.html