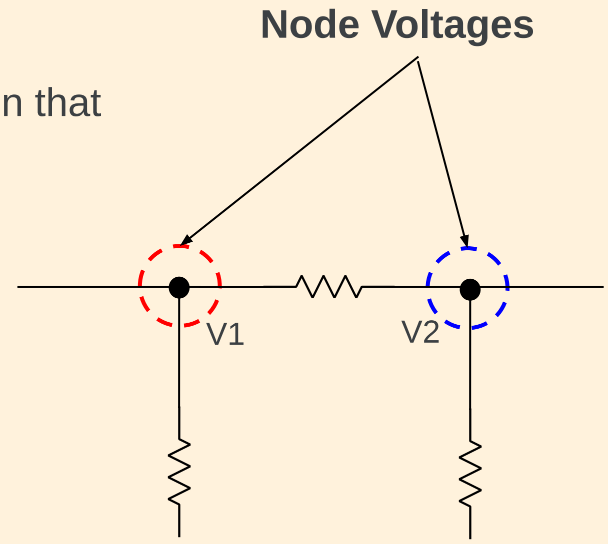
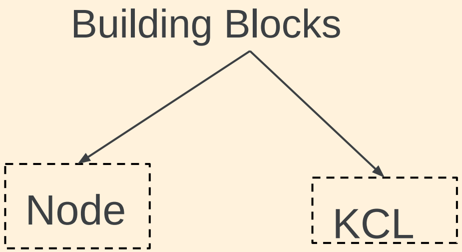
STANDARD NODAL ANALYSIS WITH PYTHON

What is nodal analysis?

Simply put, it's a systematic way to find the voltage at key points in a circuit.

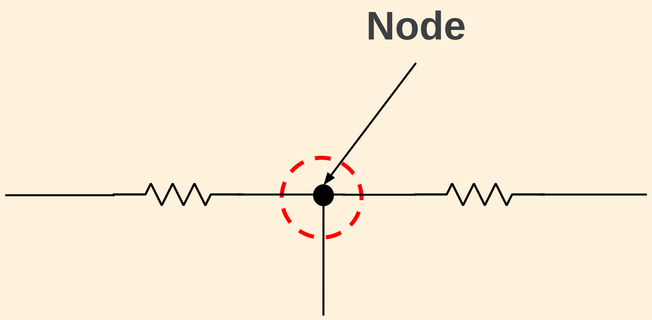
To really understand this, let's break down the key terms in that definition."





What is a node?

A node is simply a junction where two or more circuit elements—like resistors, sources, or capacitors—are connected.



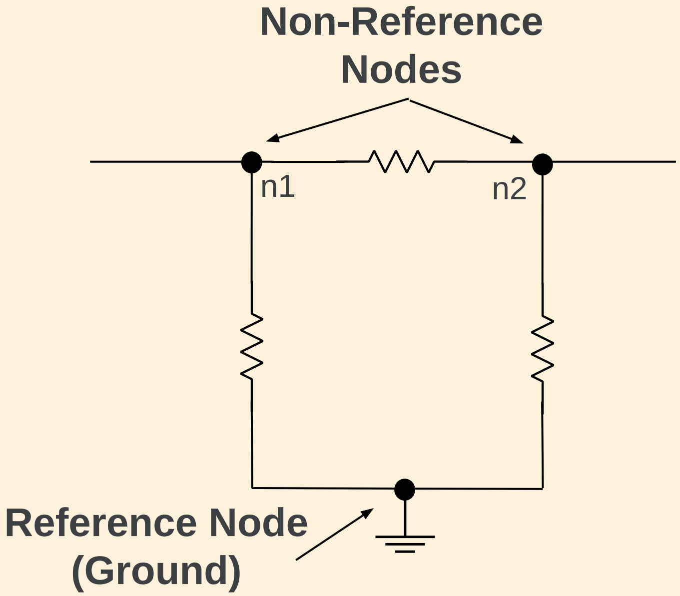
Reference vs Non-reference nodes

Reference Node

- In any circuit, we pick one node to be our **'ground' or zero-volt point**. This is the *reference node.*
- All other voltages in the circuit are measured relative to this point

Non-Reference Node

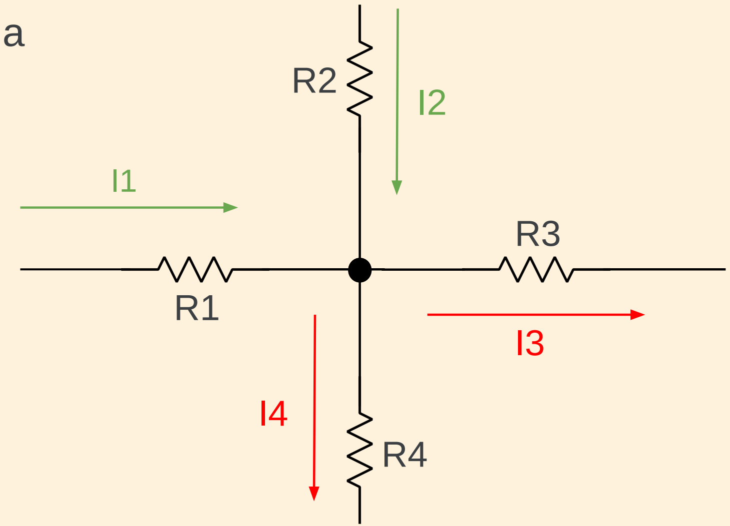
- The **other nodes,** like n1 and n2 here, are our *non-reference nodes.*
- These are the unknown voltages we want to solve for.



Kirchhoff’s Current Law (KCL)

Nodal analysis is built entirely on one fundamental principle: Kirchhoff’s Current Law. It states that the total current entering a node must equal the total current leaving it.

$$\sum i_{entering} = \sum i_{leaving}$$



$$i_1 + i_2 = i_3 + i_4$$

The Math: From Ohm's Law to Matrices

Now, how do we turn this into something we can solve? We start with Ohm's Law.

$$\Rightarrow v = i \times r$$
$$\Rightarrow i = \frac{v}{r}$$
$$\Rightarrow i = g \times v$$

Next we transform equation with matrices for a circuit with more than one node:

$$\Rightarrow I = G \cdot V$$

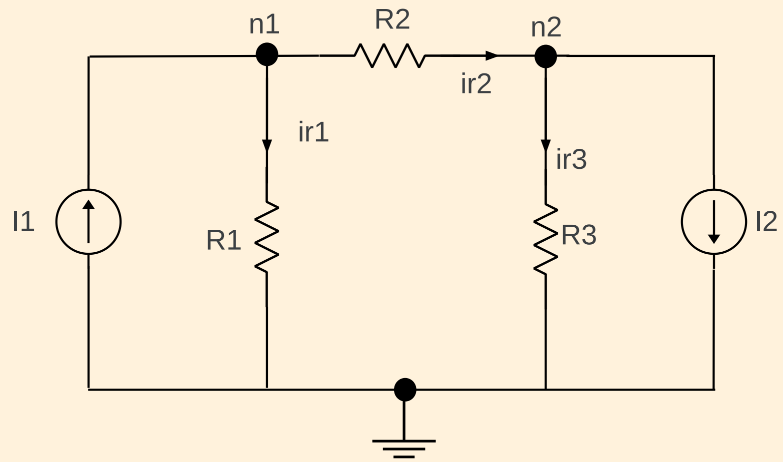
Our goal is to build the G and I matrices, and then solve for V.

$$\Rightarrow V = G^{-1} \cdot I$$

Building the Matrices: Derivation & Convention

For a two node circuit shown, we can write KCL equations as follows:

$$\text{Node1}: \; i_{r1} + i_{r2} - I_1 = 0$$
$$\text{Node2}: \; -i_{r2} + i_{r3} + I_2 = 0$$



Applying Ohm's Law:

$$\text{Node1}: \; \frac{V_1}{R_1} + \frac{(V_1 - V_2)}{R_2} = I_1$$

$$\text{Node2}: \; -\frac{(V_1 - V_2)}{R_2} + \frac{V_2}{R_3} = -I_2$$

Rearranging Terms:

$$\text{Node1}: \; \left(\frac{1}{R_1} + \frac{1}{R_2}\right)V_1 + \left(-\frac{1}{R_2}\right)V_2 = I_1$$

$$\text{Node2}: \; -\frac{1}{R_2}V_1 + \left(\frac{1}{R_2} + \frac{1}{R_3}\right)V_2 = -I_2$$

Expressing in matrix form:

$$\begin{bmatrix} (\frac{1}{R_1} + \frac{1}{R_2}) & -\frac{1}{R_2} \\ -\frac{1}{R_2} & (\frac{1}{R_2} + \frac{1}{R_3}) \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} I_1 \\ -I_2 \end{bmatrix}$$

Convention for forming Matrices

From this, we can see a clear pattern, which gives us our convention:

For the Conductance Matrix:

- The *diagonal elements* (top-left, bottom-right) are the **sum of all conductances** connected to that node.
- The *off-diagonal elements* are the **negative of sum of the conductances** connecting the two nodes.

For the Current Vector:

- If a current source **enters** a node, its value is **positive**.
- If a current source **leaves** a node, its value is **negative**.

$$\begin{bmatrix} (\frac{1}{R_1} + \frac{1}{R_2}) & -\frac{1}{R_2} \\ -\frac{1}{R_2} & (\frac{1}{R_2} + \frac{1}{R_3}) \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} I_1 \\ -I_2 \end{bmatrix}$$

N-Node Matrix Expression

We can extend the two node matrix to a general n-node matrix as follows:

$$\begin{bmatrix} (\frac{1}{r_{1a}} + \frac{1}{r_{1b}} + \dots) & -\frac{1}{r_{12}} & \cdots & \frac{1}{r_{1N}} \\ -\frac{1}{r_{21}} & (\frac{1}{r_{2a}} + \frac{1}{r_{2b}} + \dots) & \cdots & \frac{1}{r_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{r_{N1}} & \frac{1}{r_{N2}} & \vdots & (\frac{1}{r_{Na}} + \frac{1}{r_{Nb}} + \dots) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{bmatrix}$$

WARNING: The off diagonal resistance r21 is **total resistance between node 1 and 2**, not just a single resistance.

$$g_{21} = g_{21a} + g_{21b} + \dots$$

That is we can have multiple conductances/resistances in parallel between nodes, and we only use total conductance for off-diagonal elements.

Code Implementation

Alright, let's translate this into Python.

1. Loading Pre requisites:

```python
import numpy as np
from typing import List, Dict, Any
```

2. Creating a dummy **Component** variable for type hinting:

```python
Component = Dict[str, Any]
```

3. Calculating number of nodes and separating components

```python
def standard_nodal_analyis(components: List[Component])->Dict[str, Any]:
    # Get number of non-reference nodes and seperate components
    n_nodes = 0
    resistor_branches = []
    current_sources = []
    for component in components:
        n_nodes = max(n_nodes, component['n1'], component['n2'])
        if component['type'] == 'R':
            resistor_branches.append(component)
        elif component['type'] == 'I':
            current_sources.append(component)

```

Separate the components

4. Initialising Conductance matrix and current vector.

```python
    # Initialise matrices
    G = np.zeros((n_nodes, n_nodes))
    I = np.zeros((n_nodes, 1))
```

5. Calculating Conductance matrix

```python
for branch in resistor_branches:
    n1, n2 = branch['n1'], branch['n2']
    g = 1 / branch['val']

    # Index - 1 to account for python 0 indexing
    # Fill diagonal elements
    if n1 > 0:
        G[n1-1, n1-1] += g
    if n2 > 0:
        G[n2-1, n2-1] += g

    # Fill off diagonal elements
    if n1 > 0 and n2 > 0:
        G[n1-1, n2-1] -= g
        G[n2-1, n1-1] -= g
```

Add conductance for diagonal elements

Subtract conductance for off diagonal elements

6. Calculating Current vector

```python
for k, branch in enumerate(current_sources):
    n1, n2 = branch['n1'], branch['n2']
    current_val = branch['val']
    # Current is leaving n1
    if n1 > 0:
        I[n1-1] -= current_val
    # Current is entering n2
    if n2 > 0:
        I[n2 - 1] += current_val
```

Subtract current leaving nodes

Add current entering nodes

7.Solving Linear equation: GV = I

```python
V = np.linalg.solve(G, I)
```

8. Finding Branch Currents and return results in a dictionary.

```
results = {}
for i in range(n_nodes):
    results[f'V_{i+1}'] = V[i]

# Calculating branch currents
for branch in resistor_branches:
    n1, n2 = branch['n1'], branch['n2']
    g = 1 / branch['val']
    Vn1 = V[n1-1, 0] if n1 > 0 else 0
    Vn2 = V[n2-1, 0] if n2 > 0 else 0

    results[f"I_{branch['id']}"] = (Vn1 - Vn2) * g
return results
```

Calculating branch currents

EXAMPLE

Manual calculations:

$$\text{Node1}: \ -3 + i_1 + i_2 = 0$$
$$\text{Node2}: 12 - i_2 + i_3 = 0$$

Substituting node voltages:

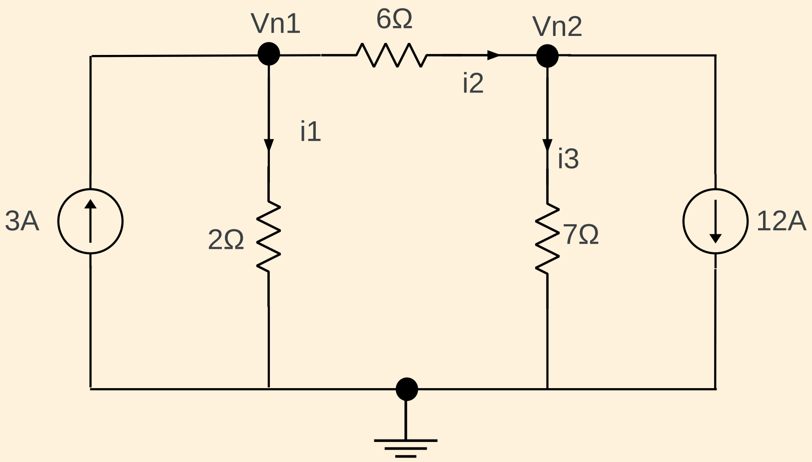
$$\text{Node1}: \ -3 + \frac{V_1}{2} + \frac{(V_1 - V_2)}{6} = 0$$
$$\text{Node2}: 12 - \frac{(V_1 - V_2)}{6} + \frac{V_2}{7} = 0$$

Simplifying equations:

$$\text{Node1}: \ -234 + 52V_1 - 13V_2 = 0$$
$$\text{Node2}: 504 - 7V_1 + 7V_2 + 6V_2 = 0$$

Solving two equations we get:

$$V_1 = -6V$$
$$V_2 = -42V$$



Solving for branch currents we get:

$$I_1 = \frac{-6}{2} = -3A \qquad I_2 = \frac{-6+42}{6} = 6A \qquad I_3 = \frac{-42}{7} = -6A$$

Solving circuit using solver

```python
components = [
    {'id': 'I1', 'type': 'I', 'n1': 0, 'n2': 1, 'val': 3},
    {'id': 'I2', 'type': 'I', 'n1': 2, 'n2': 0, 'val': 12},
    {'id': 'R1', 'type': 'R', 'n1': 0, 'n2': 1, 'val': 2},
    {'id': 'R2', 'type': 'R', 'n1': 1, 'n2': 2, 'val': 6},
    {'id': 'R3', 'type': 'R', 'n1': 2, 'n2': 0, 'val': 7},
]
```

```python
results = standard_nodal_analyis(components)

print(f"{'Variable':<30}Value")
for k, v in results.items():
    print(f'{k:<30} {np.round(v, 4)}')
```

```
Variable                      Value
V_1                            [-6.]
V_2                            [-42.]
I_R1                           3.0
I_R2                           6.0
I_R3                           -6.0
```

SUMMARY

$$\begin{bmatrix} (\frac{1}{r_{1a}} + \frac{1}{r_{1b}} + \dots) & -\frac{1}{r_{12}} & \cdots & \frac{1}{r_{1N}} \\ -\frac{1}{r_{21}} & (\frac{1}{r_{2a}} + \frac{1}{r_{2b}} + \dots) & \cdots & \frac{1}{r_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{r_{N1}} & \frac{1}{r_{N2}} & \vdots & (\frac{1}{r_{Na}} + \frac{1}{r_{Nb}} + \dots) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{bmatrix}$$

For the Conductance Matrix:

- The *diagonal elements* (top-left, bottom-right) are the **sum of all conductances** connected to that node.
- The *off-diagonal elements* are the **negative of sum of the conductances** connecting the two nodes.

For the Current Vector:

- If a current source **enters** a node, its value is **positive**.
- If a current source **leaves** a node, its value is **negative**.