Python – Testat 2 (FS2022)

1 Aufgabenstellung und Bewertungskriterien

☞ Implementieren Sie die Funktionen wie unten im Detail beschrieben und liefern Sie diese **als einzelne Datei mit dem Namen**[1] `testat2_<PID>.txt` der Dozierenden **bis spätestens am Dienstag, 31.5.2022 um 13:00 Uhr per Email** (an selina.malacarne@ost.ch) ab. Dieser Testataufgabe ist ein **gezippter Ordner beigefügt**, welcher alle **für die Aufgabenstellung relevanten Informationen** enthält. Darin finden Sie auch ein `testat2_<PID>.py`-File, welches Sie **für die Erarbeitung Ihrer Lösung verwenden sollen**.

Bewertungskriterien: Für jede korrekt implementierte Teilaufgabe erhalten Sie Punkte. Rechts neben jeder Detailbeschreibung, ist die Zahl der möglichen Punkte angegeben. Die Maximalpunktzahl dieses Testats beträgt **21 Punkte**. Direkt unterhalb der Zahl der möglichen Punkte finden Sie jeweils die Testnummern (bspw. T201, T202 etc.). Diese Nummern repräsentieren die automatischen Tests, welche mit der Prüferklasse geprüft werden. Sie erhalten nach Abgabe Ihrer Lösung die Auswertung in Form eines Textfiles mit einer Auflistung der Ergebnisse jedes einzelnen Tests. Beiliegend zu dieser Aufgabenstellung erhalten Sie ein Beispiel eines solchen Auswertungsfiles. Diesem können Sie weitere Details dazu entnehmen, was genau in den einzelnen Tests geprüft wird und welche Teilpunkte für jeden erfolgreichen Test vergeben werden.

Prüfungszulassung: Für die Zulassung zur Modulschlussprüfung müssen für jedes Testat **mindestens 40%** der erreichbaren Punkte erzielt werden.

2 Allgemeine Bewertungskriterien

- Die Funktionen besitzen aussagekräftige Docstrings. — 1 P. T201
- Der Python-Codestil entspricht den PEP8-Empfehlungen.[2] — 2 P. T202

2.1 Nach PEP8-Stilfehlern suchen

Das Flake8-Programm kann wie folgt installiert werden:

1. "Anaconda Prompt" **als Administrator öffnen** und folgende Zeile ausführen:
2. `conda install -c anaconda flake8`

Überprüfen Sie ihren Python-Code nach Stilfehlern, indem Sie das Flake8-Programm[3] wie folgt in der "Anaconda Prompt" aufrufen:

```
flake8 python_datei.py
```

[1] *.py Dateien werden vom OST Mailserver blockiert, ihre PID können Sie aus der Python Klassenliste vom Unterrichtsportal entnehmen.

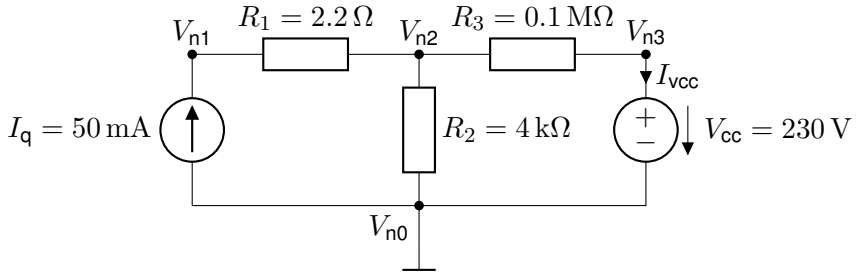
[2] Der Codestil wird mit der Flake8-Software geprüft, `http://flake8.pycqa.org/en/latest/`

[3] Die Anaconda Prompt muss beim Filepfad des zu prüfenden Pythonskripts geöffnet sein. Zum Filepfad navigieren Sie mit dem Befehl `cd <path>`

3 Einleitung

Grössere elektrische Netzwerke werden üblicherweise nicht von Hand berechnet, sondern auf einem Computer simuliert. Dafür hat sich in den vergangenen 50 Jahren SPICE[4] als Standard etabliert, welcher von verschiedenen Herstellern implementiert und erweitert wurde. Einem solchen Simulator wird das Netzwerk als Textfile, der sogenannten Netzliste, übergeben. Ein Beispiel ist in Abb. 1 abgebildet: Die Netzliste (b) beinhaltet die Schaltung (a).

Ziel dieser Aufgabe ist es, einen eigenen solchen Simulator zu implementieren. Ihr Simulator soll nur die folgenden drei Netzwerkelemente unterstützen: Widerstand (R), ideale Spannungsquelle (V) und ideale Stromquelle (I).



(a)

```
R1 1 2 2.2
R2 2 0 4k
R3 2 3 0.1MEG
Iq 0 1 DC 50e-3
Vcc 3 0 DC 230
```

(b) `netlist.cir`

Abbildung 1: Elektrische Schaltung (a) mit zugehöriger Netzliste (b).

Jede Zeile in der Netzliste beschreibt jeweils ein einzelnes Element im Netzwerk. Die verwendete Syntax für die drei Elementtypen ist in Tab. 1 aufgelistet.

Tabelle 1: Syntax der Netzlistenelemente. `<...>`: Pflichtfelder, `[...]`: optional.

Elementtyp	**Syntax**	**Beispiele**

Widerstand	`R<name> <n+> <n-> <value>`	`R1 0 4 220`
		`R2 4 3 1e3`
		`R35 2 3 5.1k`
		`R401 2 1 2.2MEG`

Spannungsquelle	`V<name> <n+> <n-> [DC] <value>`	`V1 1 0 3.3`
		`V2 1 2 DC 12`
		`VCC 2 3 5.5`

Stromquelle	`I<name> <n+> <n-> [DC] <value>`	`I1 4 5 DC 0.2`
		`I2 5 0 2.5m`
		`Iq 4 1 7.5u`

Die Syntax befolgt die folgenden allgemeinen Punkte:

- Eine Zeile beginnt mit einem der drei Buchstaben: `R`, `V` oder `I`, wobei auch Kleinbuchstaben erlaubt sind.
- `<name>` enthält eine oder mehrere Ziffern und/oder Buchstaben.

[4]SPICE = Simulation Program with Integrated Circuit Emphasis

- `<n+>` enthält die ganzzahlige Nummer des Startknotens des jeweiligen Elements.
- `<n->` enthält die ganzzahlige Nummer des Endknotens des jeweiligen Elements.
- `[DC]` ist ein optionaler Parameter, welcher bei den Quellen (`V`, `I`) vorkommen kann.
- `<value>` enthält den Elementwert, welcher auf verschiedene Weise spezifiziert werden kann. Zum Beispiel kann der Widerstandswert von 4.7 kΩ wie folgt angegeben werden:
  - als Ganzzahl oder Floatzahl (4700, 4700.0),
  - in wissenschaftlicher Notation (4.7e3, 4.7E3)
  - oder mit einem Präfix (4.7k, 4.7K).

  Dabei ist sowohl die Gross- als auch die Kleinschreibung möglich. Eine Liste der möglichen Präfixe ist in Tab. 2 ersichtlich.

Tabelle 2: Einheitenpräfixe

F	P	N	U	M	K	MEG	G	T
femto	pico	nano	micro	milli	kilo	mega	giga	tera
10^{-15}	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{3}	10^{6}	10^{9}	10^{12}

4 Funktion `netlist_parser(fname)`

4 P.
T401
T402
T403

Die Funktion liest die angegebene Textdatei ein, welche eine gültige Netzliste beinhaltet, und liefert ein Dictionary mit allen Elementinformationen. Der vollständige Elementname (z.B. `"R1"`, `"Iq"`, `"Vcc"`) soll als Schlüsselwort übernommen werden, wobei der Anfangsbuchstabe in Grossschreibung und der Rest in Kleinschreibung sein soll. Zu diesem Schlüsselwort wird wiederum ein Dictionary hinterlegt, welches die folgenden Schlüsselwörter und Werte besitzt:

`"n+"`:	`"Vn"` + Nummer des Startknotens als String
`"n-"`:	`"Vn"` + Nummer des Endknotens als String
`"value"`:	Elementwert als float-Zahl

Hinweis: Der optionale [DC]-Parameter wird ignoriert.

Zum Beispiel wird aus der Netzliste, welche in der bereitgestellten Datei `netlist.cir` enthalten ist, folgendes Dictionary zurückgegeben:

```
>>> elements = netlist_parser("netlist.cir")
>>> print(elements)
{"R1": {"n+": "Vn1", "n-": "Vn2", "value": 2.2},
 "R2": {"n+": "Vn2", "n-": "Vn0", "value": 4000.0},
 "R3": {"n+": "Vn2", "n-": "Vn3", "value": 100000.0},
 "Iq": {"n+": "Vn0", "n-": "Vn1", "value": 0.05},
 "Vcc": {"n+": "Vn3", "n-": "Vn0", "value": 230.0}}
```

1 P.
T404
T405

In der Netzliste können auch Kommentare vorhanden sein, welche ignoriert werden sollen. Die Funktion soll zwei Arten von Kommentaren unterscheiden und ignorieren:

```
* Eine Zeile, welche mit einem Asterisk beginnt, wird ignoriert.
R1 1 2 2.2 ; Der Text ab einem Semikolon wird ignoriert.
```

```
>>> elements = netlist_parser("netlist_with_comments.cir")
```

5 Funktion `inventory(elements)`

Diese Funktion liest das angegebene Dictionary mit den Elementinformationen ein und gibt die Elementnamen als formatierter String zurück. Dabei sollen die Elementnamen nach Typ gruppiert und in beliebiger Reihenfolge mit einem Komma separiert werden. Im folgenden Beispiel sind die Details ersichtlich:

2 P.
T501
T502

```
>>> s = inventory(elements)
>>> print(s)
Resistors: R1, R2, R3
Voltage Sources: Vcc
Current Sources: Iq
```

6 Funktion `mna_build(elements)`

Die mathematische Methode auf der SPICE basiert, ist das sogenannte *modifizierte Knotenpotentialverfahren*[5]. Aus der Netzliste wird eine grosse Matrix zusammengestellt, die angewendet auf einen Vektor unbekannter (d.h. gesuchter) Ströme und Spannungen die Quellenspannungen- und Ströme ergibt. Für die Schaltung in Abb. 1 resultiert das folgende Gleichungssystem:

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{y} \tag{1}$$

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_3} & 0 \\ 0 & -\frac{1}{R_3} & \frac{1}{R_3} & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_{\mathrm{n1}} \\ V_{\mathrm{n2}} \\ V_{\mathrm{n3}} \\ I_{\mathrm{vcc}} \end{bmatrix} = \begin{bmatrix} I_{\mathrm{q}} \\ 0 \\ 0 \\ V_{\mathrm{cc}} \end{bmatrix} \tag{2}$$

Mittels Matrixinversion, d.h. $\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{y}$, können dann die Unbekannten berechnet werden.

Die `mna_build(elements)`-Funktion nimmt das Dictionary mit den Elementinformationen entgegen und gibt die folgenden drei Objekte zurück:

1 P.
T601

- Die Liste `unknowns` mit den Namen der Unbekannten im Lösungsvektors $\boldsymbol{x}$.
- Die Matrix $\boldsymbol{M}$ mit den Widerstandskehrwerten als 2D-NumPy-Array.
- Der Quellenvektor $\boldsymbol{y}$ mit den bekannten Quellenwerten als 1D-NumPy-Array.

Zur obigen Beispielschaltung gibt die Funktion folgende Daten zurück:

```
>>> unknowns, M, y = mna_build(elements)
>>> print(unknowns)
["Vn1", "Vn2", "Vn3", "Ivcc"]
>>> print(M)
[[ 4.54545455e-01 -4.54545455e-01 0.00000000e+00 0.00000000e+00]
 [-4.54545455e-01 4.54805455e-01 -1.00000000e-05 0.00000000e+00]
```

[5] Englisch: Modified Nodal Analysis.
Mehr Infos unter: `https://www.swarthmore.edu/NatSci/echeeve1/Ref/mna/MNA2.html`

```
 [ 0.00000000e+00 -1.00000000e-05 1.00000000e-05 1.00000000e+00]
 [ 0.00000000e+00 0.00000000e+00 1.00000000e+00 0.00000000e+00]]
>>> print(y)
[5.0e-02 0.0e+00 0.0e+00 2.3e+02]
```

Die Liste `unknowns` besteht aus Strings mit den Namen der Unbekannten des Lösungsvektors $\boldsymbol{x}$. Der Lösungsvektor $\boldsymbol{x}$ enthält sowohl die unbekannten Knotenspannungen (z.B. $V_{\mathrm{n1}}, V_{\mathrm{n2}}, \ldots$) als auch die unbekannten Ströme (z.B. I_{vcc}) der vorhandenen Spannungsquellen (z.B. V_{cc}). Die Namen der Knotenspannungen beginnen mit `"Vn"`, gefolgt von der Knotennummer. Die Namen der unbekannten Quellenströme setzen sich aus dem Grossbuchstaben `"I"` und dem jeweiligen Namen der Spannungsquelle (z.B. `"VCC"`) in Kleinschreibung zusammen. Die Knotenspannung V_{n0} entspricht dem Wert 0 und wird weder im Vektor $\boldsymbol{x}$ noch in der Liste `unknowns` aufgelistet. Die Reihenfolge der Namen in `unknowns` kann theoretisch beliebig gewählt werden. Diese hat aber natürlich einen Einfluss, wie die Werte in $\boldsymbol{M}$ und $\boldsymbol{y}$ abgelegt werden müssen.

1 P. T602

Es sollen N unbekannte Spannungen und Ströme gelöst werden, wobei N der Anzahl Knoten (ohne Null-Potential V_{n0}) plus Anzahl idealer Spannungsquellen entspricht. N ist somit die Anzahl unbekannter Variablen (in `unknowns`). Damit das System lösbar ist, muss folglich die Matrix die Dimensionen $N \times N$ haben und auch der Quellenvektor muss N Einträge aufweisen.

Die Elementwerte für $\boldsymbol{M}$ und $\boldsymbol{y}$ können mithilfe von sogenannten Stempeln systematisch abgefüllt werden. Dabei werden die Werte des Stempels jeweils den Werten in der Matrix oder im Vektor addiert. Für jeden Elementtyp existiert ein eigener Stempel.

- **Widerstandsstempel**: Bei einem Widerstand R werden nur die Werte der Matrix $\boldsymbol{M}$ verändert. Zuerst werden die Nummern der Start- (`n+`) und Endknoten (`n-`) gelesen und deren Position (Index) in der Liste `unknowns` bestimmt. Danach wird der Kehrwert des Widerstandswerts (z.B. $1/R_1$) in den entsprechenden Positionen in $\boldsymbol{M}$ (mit unterschiedlichem Vorzeichen) dazuaddiert. Ein Beispiel wird in Abb. 2 gezeigt. Falls eines der Start- oder Endknoten dem Nullknoten (V_{n0}) entspricht, dann fallen die entsprechenden Stempelpositionen (d.h. die ganzen Reihen oder Spalten) weg.

1 P. T603

$$\text{Teil von } \boldsymbol{M}: \quad \begin{matrix} \texttt{n+} & \texttt{n-} \\ \downarrow & \downarrow \end{matrix} \begin{bmatrix} +\frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & +\frac{1}{R} \end{bmatrix} \begin{matrix} \leftarrow \texttt{n+} \\ \leftarrow \texttt{n-} \end{matrix}$$

(a) allgemein

$$\boldsymbol{M} = \begin{matrix} V_{\mathrm{n1}} & V_{\mathrm{n2}} & V_{\mathrm{n3}} & I_{\mathrm{vcc}} \\ \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \begin{bmatrix} +\frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & +\frac{1}{R_1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow V_{\mathrm{n1}} \\ \leftarrow V_{\mathrm{n2}} \\ \leftarrow V_{\mathrm{n3}} \\ \leftarrow I_{\mathrm{vcc}} \end{matrix}$$

(b) Beispiel für R_1 (`n+` $\rightarrow V_{\mathrm{n1}}$, `n-` $\rightarrow V_{\mathrm{n2}}$)

Abbildung 2: Stempel für Widerstände.

- **Stromquellenstempel**: Bei einer idealen Stromquelle I werden nur die Werte des Quellenvektors $\boldsymbol{y}$ verändert. Es muss bei den entsprechenden Positionen (je nach Start- und Endknoten) in $\boldsymbol{y}$ der Stromwert I dazuaddiert oder davon subtrahiert werden, wie in Abb. 3 gezeigt wird. Auch hier fallen die Einträge, welche dem Nullknoten entsprechen würden, weg.

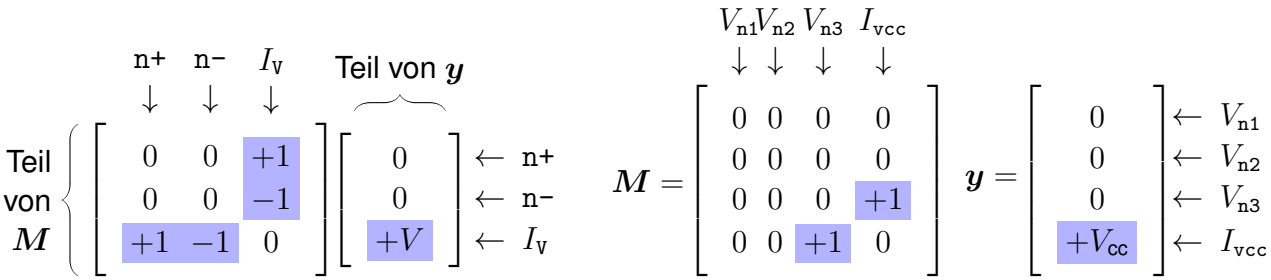
1 P. T604

(a) allgemein (b) Beispiel für I_q ($\mathtt{n+} \rightarrow V_\mathrm{n0}$, $\mathtt{n-} \rightarrow V_\mathrm{n1}$)

Abbildung 3: Stempel für Stromquellen.

- **Spannungsquellenstempel**: Bei einer idealen Spannungsquelle V werden sowohl die Werte der Matrix $\boldsymbol{M}$ als auch die Werte des Quellenvektors $\boldsymbol{y}$ verändert, wie in Abb. 4 gezeigt. Dabei werden in der Matrix $\boldsymbol{M}$ bei der entsprechenden Spalte und Zeile eine Eins dazu addiert oder subtrahiert, je nach Start- und Endknoten. Im Vektor $\boldsymbol{y}$ wird bei der entsprechenden Position der Spannungswert dazu addiert. Wiederum fallen die Einträge, welche dem Nullknoten entsprechen würden, weg.

1 P. T605



(a) allgemein (b) Beispiel für V_cc ($\mathtt{n+} \rightarrow V_\mathrm{n3}$, $\mathtt{n-} \rightarrow V_\mathrm{n0}$)

Abbildung 4: Stempel für Spannungsquellen.

7 Funktion `mna_solve(unknowns, M, y)`

Diese Funktion berechnet mithilfe der Matrixinversion den Lösungsvektor

2 P. T701 T702

$$\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{y} \tag{3}$$

und gibt ein Dictionary zurück, welches die Namen aus der Liste `unknowns` mit den entsprechenden Lösungswerten aus $\boldsymbol{x}$ verknüpft, z.B.

```
>>> solution = mna_solve(unknowns, M, y)
>>> print(solution)
{"Vn1": 201.26384615385973,
 "Vn2": 201.15384615385975,
```

```
 "Vn3": 230.0,
 "Ivcc": -0.00028846153846140 27}
```

8 Funktion `mna_report(elements, solution)`

4 P.
T801
T802
T803
T804

Diese Funktion berechnet die Spannungen und Ströme aller Elemente und gibt diese als Dictionary zurück. Der Elementname wird als Schlüsselwort übernommen und zu einem Dictionary verknüpft, welches die jeweilige Spannung (`V`) und den jeweiligen Strom (`I`) enthält, wie im folgenden Beispiel gezeigt:

```
>>> report = mna_report(elements, solution)
>>> print(report)
{"R1": {"V": 0.10999999999998522, "I": 0.04999999999999328},
 "R2": {"V": 201.15384615385975, "I": 0.05028846153846494},
 "R3": {"V": -28.846153846140254, "I": -0.00028846153846140253},
 "Iq": {"V": -201.26384615385973, "I": 0.05},
 "Vcc": {"V": 230.0, "I": -0.00028846153846140 27}}
```

Die Spannungen und Ströme der Elemente lassen sich mit den Formeln aus Tab. 3 ermitteln. Falls bei einem Element der Startknoten (`n+`) oder der Endknoten (`n-`) dem Nullknoten (`Vn0`) entspricht, dann soll dafür der Wert $\texttt{Vn0} = 0$ eingesetzt werden.

Tabelle 3: Berechnungsformeln für die Spannungen und Ströme der Elemente.

Elementtyp	**Spannung**	**Strom**	**Beispiele**
Widerstand	$V = V_{\texttt{n+}} - V_{\texttt{n-}}$	$I = \dfrac{V_{\texttt{n+}} - V_{\texttt{n-}}}{R}$	für `"R1"`: $V = V_{\text{n1}} - V_{\text{n2}}$ $I = \dfrac{V_{\text{n1}} - V_{\text{n2}}}{R_1}$
Spannungsquelle	$V = V$	$I = I_{\texttt{V}}$	für `"Vcc"`: $V = V_{\text{cc}}$ $I = I_{\text{vcc}}$
Stromquelle	$V = V_{\texttt{n+}} - V_{\texttt{n-}}$	$I = I$	für `"Iq"`: $V = V_{\text{n0}} - V_{\text{n1}} = -V_{\text{n1}}$ $I = I_{\text{q}}$