The svgBondGraph module

Peter Gawthrop. *peter.gawthrop@unimelb.edu.au*

May 21, 2024

Contents

Note: this is the svgBondGraph.ipynb notebook. The PDF version "The svgBondGraph module" is available here.

1 Introduction

Bond graphs can be represented graphically in scalar vector graphics (SGV) format. Such representations can be generated ab initio using an editor such as xfig or inkscape, or can be generated from legacy MTT files in fig format. This document describes how bond graphs in svg format can be converted to **BondGraphTools** format.

2 Converting from fig to svg

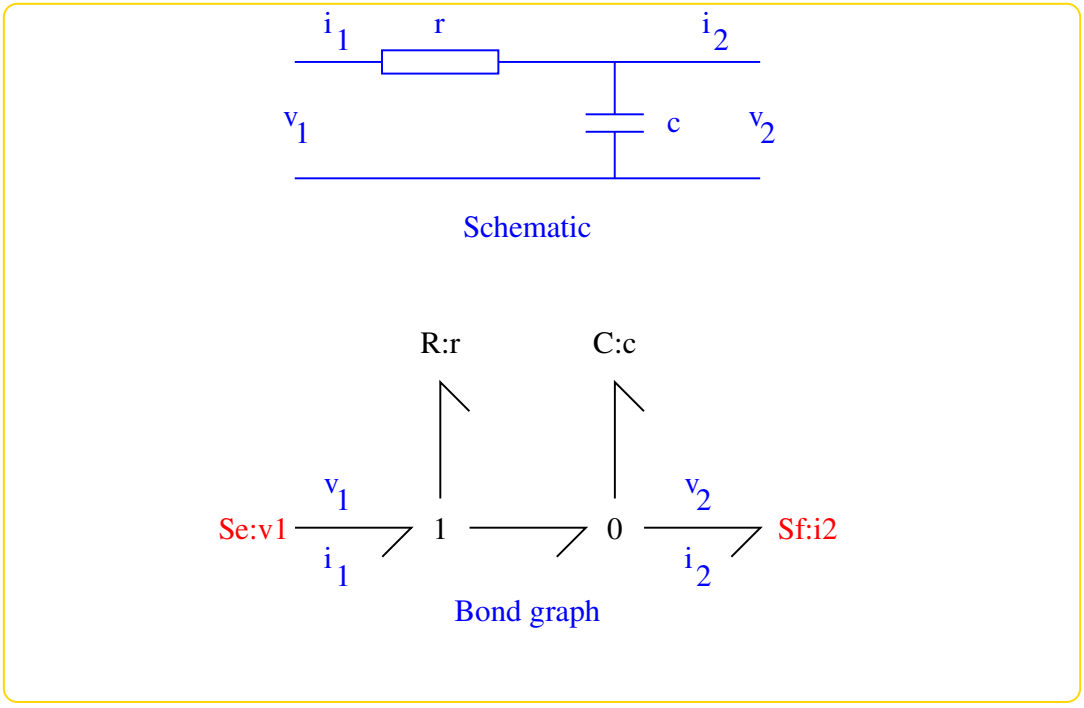
Legacy bond graph graphical representation generated by xfig are .fig files. .fig graphics can be converted to .svg graphics in various ways including via inkscape. However, fig2dev provides a simple conversion approach:

!fig2dev -Lsvg RC_abg.fig > RC_abg.svg

This gives the svg representation:

[1]:
```
import IPython.display as disp
disp.SVG('RC_abg.svg')
```

[1]:



3 Converting from graphical to computational representation

SVG files are in XML format and, as such, can be parsed using python modules such as **lxml** and **svgpathtools**. **svgBondGraph** provides the tools to convert bond graphs in svg format to bond graphs in **BondGraphTools** format.

[2]:
```
import svgBondGraph as sbg
sbg.model('RC_abg.svg')
```

{}

This generates the file ABCDE_abg.py which can be imported as usual. (Note that, by default, the BG itself is distinguished from decoration by colour: the default includes black and red)

[3]:
```
import BondGraphTools as bgt
import sympy as sp

def model():
    """ Acausal bond graph RC_abg.py
    Created by svgBondGraph at Tue Dec 18 11:03:34 2018 from RC_abg.svg

    Usage:
    import RC_abg; model = RC_abg.model()
    """

    model = bgt.new(name="RC")

    ## Junction 0:MTT1
    MTT1 = bgt.new('0')

    ## Junction 1:MTT0
    MTT0 = bgt.new('1')

    ## Component C:c
    c =  sp.symbols('c')
    c = bgt.new('C',name='c',value={'C':c})

    ## Component R:r
    r =  sp.symbols('r')
    r = bgt.new('R',name='r',value={'r':r})

    ## Component Se:v1
    v1 = bgt.new('Se',name='v1')

    ## Component Sf:i2
    i2 = bgt.new('Sf',name='i2')

    ## Component list
    components = (
      MTT1,
      MTT0,
      c,
      r,
      v1,
      i2
    )
```

```
    bgt.add(model, *components)

    ## Bonds
    bgt.connect(v1,MTT0)
    bgt.connect(MTT0,MTT1)
    bgt.connect(MTT0,r)
    bgt.connect(MTT1,c)
    bgt.connect(MTT1,i2)

    return model
```

ABCDE_abg.py can be imported and analysed using BondGraphTools. For example:

[4]:
```
import RC_abg; model = RC_abg.model(); help(RC_abg)
model.constitutive_relations
```

```
Help on module RC_abg:

NAME
    RC_abg

FUNCTIONS
    model()
        Acausal bond graph RC_abg.py
        Created by svgBondGraph at Tue May 21 08:56:53 2024 from RC_abg.svg

        Usage:
        import RC_abg; model = RC_abg.model()

FILE
    /home/peterg/WEB/Github/Tutorials/RC_abg.py
```

[4]:
```
[dx_0 - u_1 - u_0/r + x_0/(c*r)]
```

[5]:
```
model.control_vars
```

[5]:
```
{'u_0': (SS: v1, 'e'), 'u_1': (SS: i2, 'f')}
```

This can, for example, be simulated for numerical parameter values.

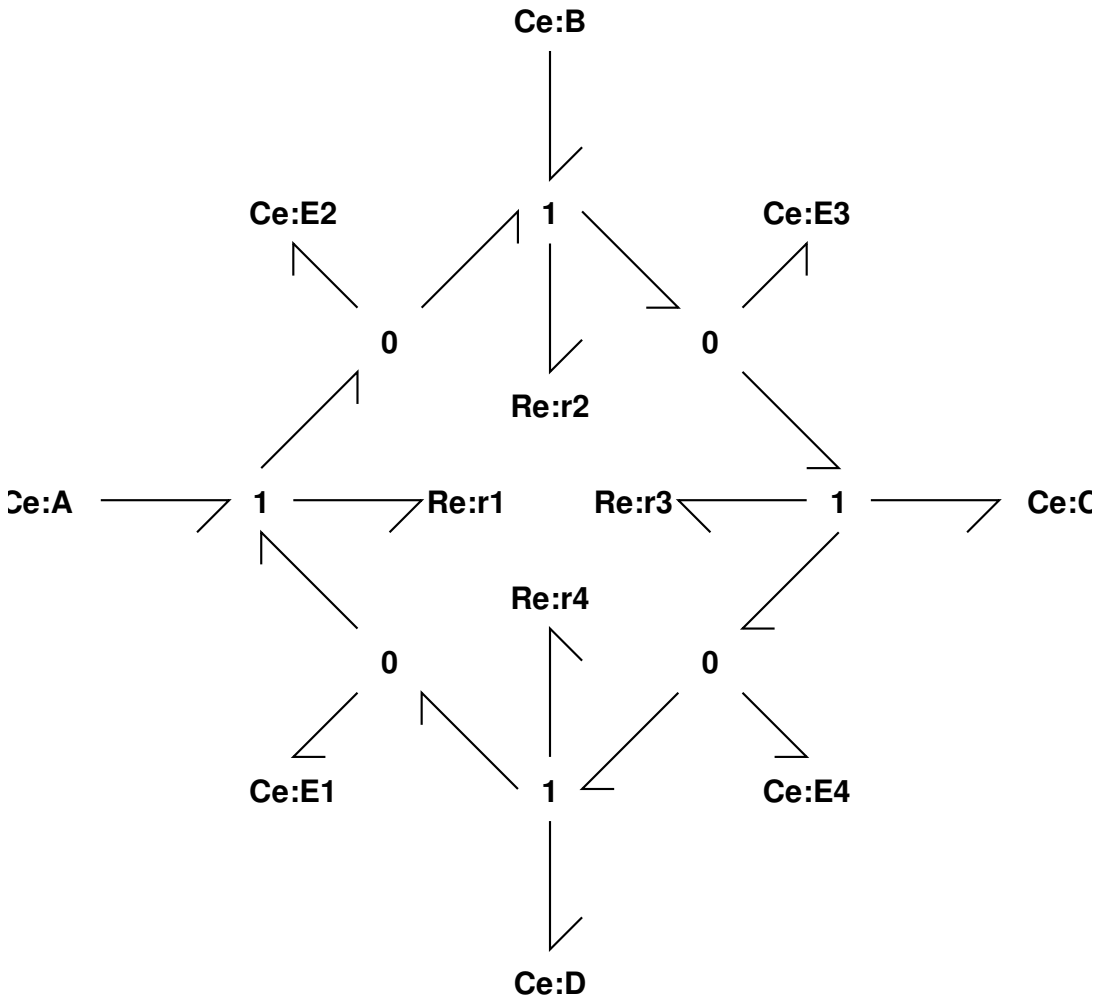
4 A biomolecular example.

!fig2dev -Lsvg ABCDE_abg.fig > ABCDE_abg.svg

This gives the svg representation:

[6]:
```
disp.SVG('ABCDE_abg.svg')
```

[6]:



[7]:
```
sbg.model('ABCDE_abg.svg')
```

```
Converting one-port r2 to two-port
Converting one-port r4 to two-port
Converting one-port r1 to two-port
Converting one-port r3 to two-port
{}
```

This generates ABCDE_abg.py which can be imported as usual and is then available for, for example, stoichiometric analysis.

[8]:
```
import ABCDE_abg
ABCDE = ABCDE_abg.model()
```

[9]:
```
ABCDE.state_vars
```

[9]:
```
{'x_0': (C: A, 'q_0'),
 'x_1': (C: B, 'q_0'),
 'x_2': (C: C, 'q_0'),
 'x_3': (C: D, 'q_0'),
 'x_4': (C: E1, 'q_0'),
```

```
 'x_5': (C: E2, 'q_0'),
 'x_6': (C: E3, 'q_0'),
 'x_7': (C: E4, 'q_0')}
```

[10]:
```
ABCDE.constitutive_relations
```

[10]:
```
[K_A*K_E1*kappa_r1*x_0*x_4 - K_E2*kappa_r1*x_5 + dx_0,
 K_B*K_E2*kappa_r2*x_1*x_5 - K_E3*kappa_r2*x_6 + dx_1,
 K_C*K_E4*kappa_r3*x_2*x_7 - K_E3*kappa_r3*x_6 + dx_2,
 K_D*K_E1*kappa_r4*x_3*x_4 - K_E4*kappa_r4*x_7 + dx_3,
 K_A*K_E1*kappa_r1*x_0*x_4 + K_D*K_E1*kappa_r4*x_3*x_4 - K_E2*kappa_r1*x_5 -
K_E4*kappa_r4*x_7 + dx_4,
 -K_A*K_E1*kappa_r1*x_0*x_4 + K_B*K_E2*kappa_r2*x_1*x_5 + K_E2*kappa_r1*x_5 -
K_E3*kappa_r2*x_6 + dx_5,
 -K_B*K_E2*kappa_r2*x_1*x_5 - K_C*K_E4*kappa_r3*x_2*x_7 + K_E3*kappa_r2*x_6 +
K_E3*kappa_r3*x_6 + dx_6,
 K_C*K_E4*kappa_r3*x_2*x_7 - K_D*K_E1*kappa_r4*x_3*x_4 - K_E3*kappa_r3*x_6 +
K_E4*kappa_r4*x_7 + dx_7]
```

[ ]:

[ ]:

References