Basic Circuit Discretizations with PSCAD

Assignment 1b

Due: 2021/01/29

1 Introduction

As a follow-up to the previous assignment, we now use a software suite known as PSCAD to simulate the same circuit as before. PSCAD is an electromagnetic transient simulator, allowing the user to model complex power systems and analyze their behaviour.

2 Setup

PSCAD provides a master library of components as well as primitives to help model the circuit. The "Single Phase Voltage Source Model" component was used as the source and it was made ideal by setting the source impedance to 0Ω with a ramp-up time of $0s$ (instantaneous voltage). The ideal resistor and inductors were used to model the passive components and a multimeter was added to the top of the inductor. The multimeter's instantaneous current and voltage readings were connected to plots which were placed directly on the schematic sheet. Figure 1 illustrates how the PSCAD schematic sheet was set up for this assignment.

Once all of the components were hooked together and instrumented, the build button was clicked. This generated a Fortran program that models the entire circuit. A listing of this program can be found in Section 5.2.

Clicking the run button ran the simulation and outputted the results on the plots. The data was extracted by right-clicking each plot to save it to the clipboard. It was then formatted and added to the Python program for plotting against the hand-derived approximations.

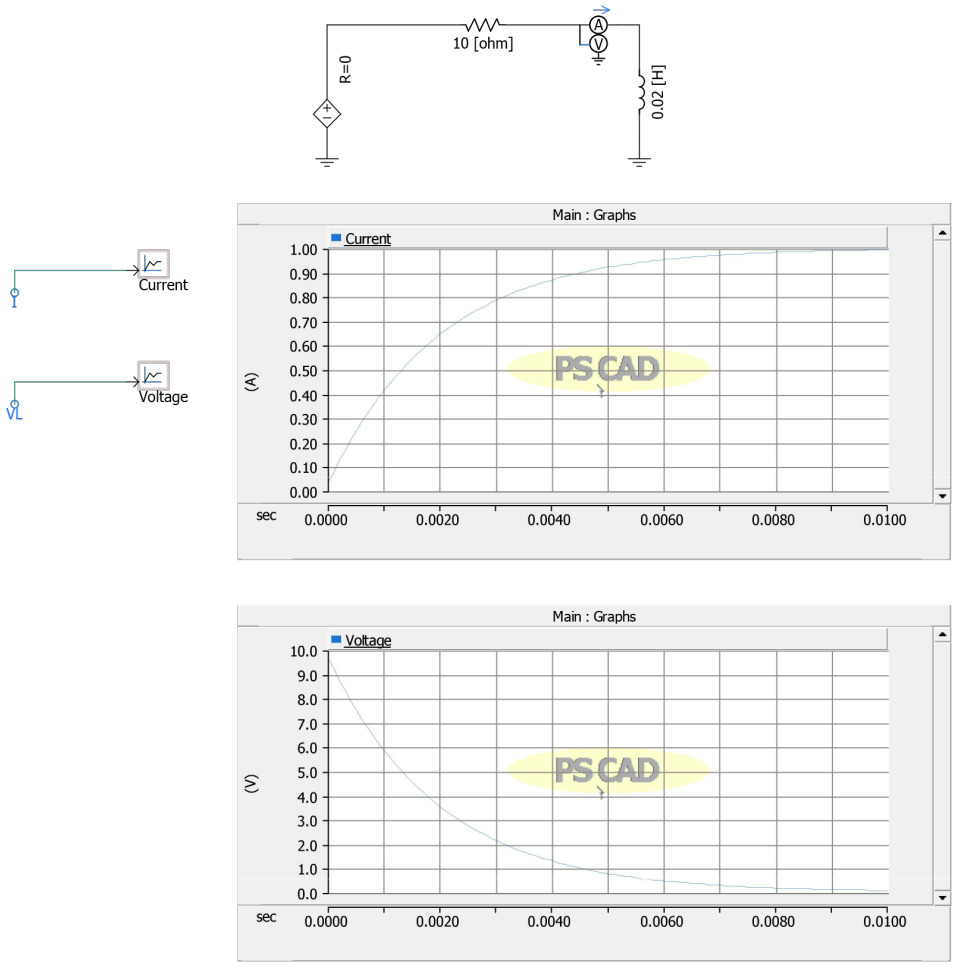

Figure 1: PSCAD schematic and resulting plots for $\Delta t = 100\mu s$ circuit analysis.

3 Simulation

Recall that in assignment 1a, the following approximations were derived and plotted:

a) Using the trapezoidal rule with $\Delta t_1 = 0.1ms$

b) Using the backward Euler rule with $\Delta t_1 = 0.1ms$

c) Using the trapezoidal rule with $\Delta t_2 = 0.8ms$

d) Using the backward Euler rule with $\Delta t_2 = 0.8ms$

In this assignment, we append the PSCAD solutions generated and compare it to the manually derived approximations on the same plots. Two sets of plots are generated, at time steps $\Delta t_1 = 0.1ms$ and $\Delta t_2 = 0.8ms$. The following two figures, 2 and 3, show PSCAD compared to the manually derived approximations for these two time steps respectively.

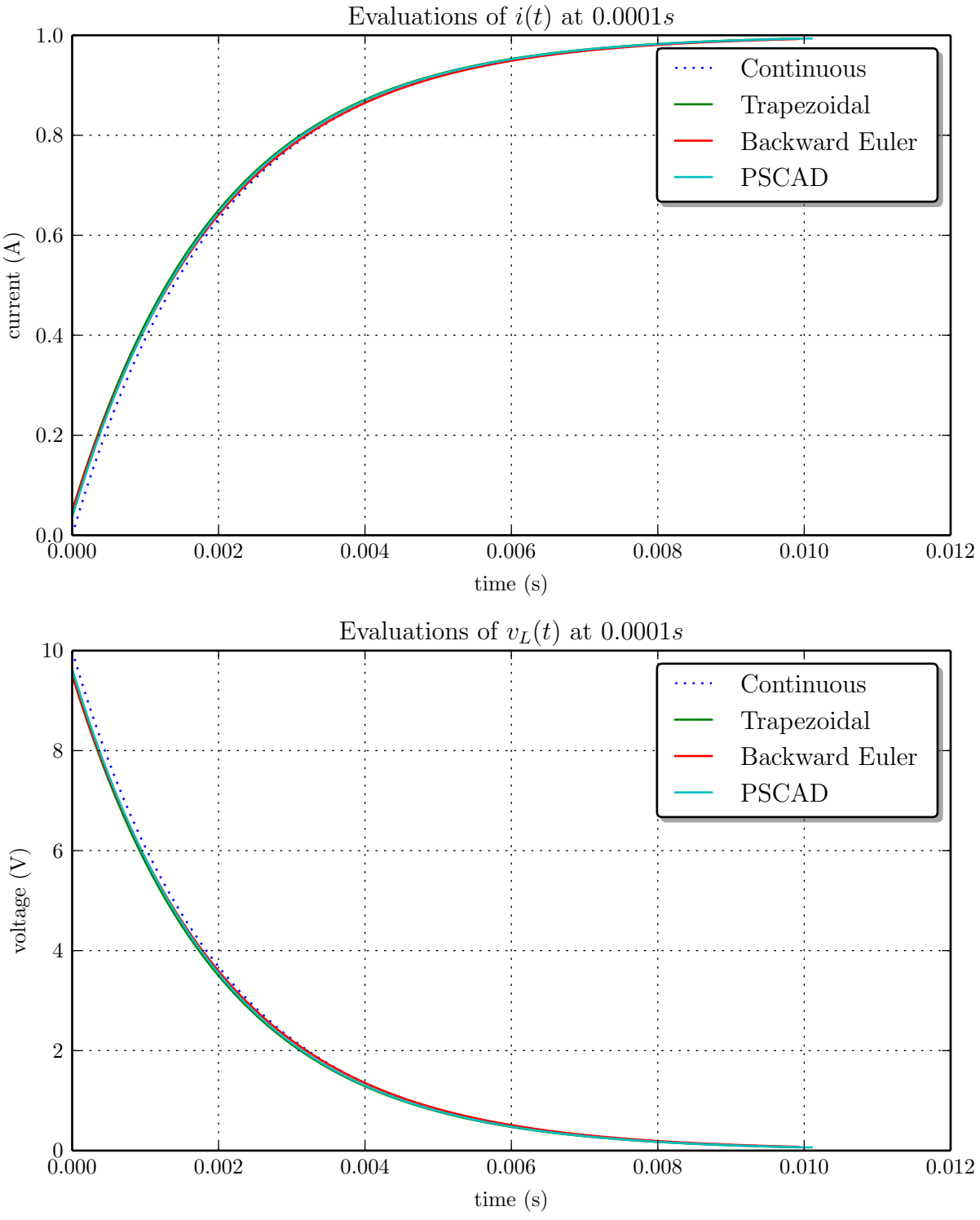

Figure 2: Comparison of Approximations and PSCAD at $\Delta t_1 = 0.1ms$

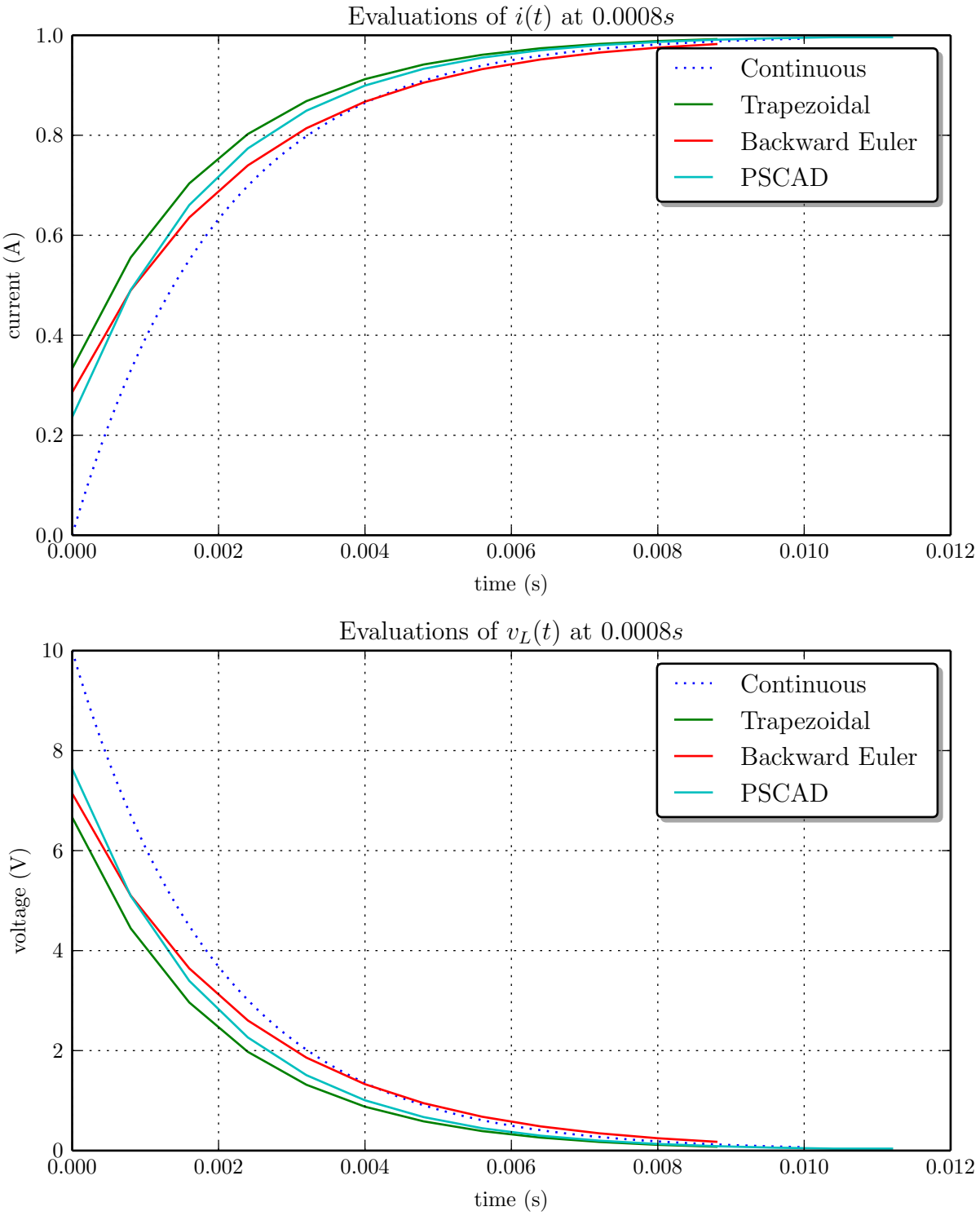

Figure 3: Comparison of Approximations and PSCAD at $\Delta t_2 = 0.8ms$

4 Conclusion

PSCAD offers very similar performance to the manually derived approximations, but with a much easier interface. Much more complex circuits can be simulated in a fraction of the time it would take to manually derive the differential equations. The following observations were made about the results in this assignment.

- At a time step of $\Delta t_1 = 0.1ms$, all three solutions are almost indistinguishable from ideal at the plotted time scale, showing how a small time step improves the performance of all three techniques considerably.
- The solutions for PSCAD are very close in shape and magnitude to the trapezoidal results that were generated in the previous assignment. This likely means that the underlying mechanism for simulation is likely similar to the trapezoidal approximation.
- Conversely, PSCAD does not have the same shape as the backward Euler solution, so it's not likely this technique.
- Interestingly enough, PSCAD appears to offer a more precise solution than the manually derived trapezoidal one for the same time step. Perhaps the time step used in PSCAD is different internally (e.g. slightly smaller than what is specified). Or, perhaps PSCAD makes different assumptions about the beginning of time of the circuit. In the PSCAD circuit, at $0^- s$ the voltage is 10 V and undefined in the manually derived equations (times before 0 are not taken into account). Perhaps this value is used somewhere in the PSCAD calculation that wasn't evident.

5 Code Listings

5.1 Python Code Listing

The following is the code written in Python to perform the calculations derived for this homework assignment as well as generate the plots used in this report. The argument parser was removed in this version due to the same parameters being used for every iteration. Also, the results from PSCAD were copy pasted directly into the Python source to keep things simple. The results are transposed to match the generated results from the manually derivated solutions.

```
import argparse
import math
import matplotlib
import matplotlib.pyplot as plt

# Matplotlib export settings
matplotlib.use("pgf")
matplotlib.rcParams.update({
    "pgf.texsystem": "pdflatex",
    "font.size": 10 ,
    "font.family": "serif",  # use serif/main font for text elements
    "text.usetex": True,     # use inline math for ticks
    "pgf.rcfonts": False     # don't setup fonts from rc parameters
})

# 0.8 ms Current
# ----------------------------------------
# Domain, Current
pscad_dat_current_0p8 = [
    [0.0,0.236111111111],
    [0.0008,0.490740740741],
    [0.0016,0.66049382716],
    [0.0024,0.77366255144],
    [0.0032,0.849108367627],
    [0.004,0.899405578418],
    [0.0048,0.932937052279],
    [0.0056,0.955291368186],
    [0.0064,0.970194245457],
    [0.0072,0.980129496971],
    [0.008,0.986752997981],
    [0.0088,0.991168665321],
    [0.0096,0.994112443547],
    [0.0104,0.996074962365],
    [0.0112,0.996074962365],
]
# transpose
pscad_dat_current_0p8 = map(list, zip(*pscad_dat_current_0p8))

# 0.8 ms Voltage
# ----------------------------------------
# Domain, Voltage
pscad_dat_voltage_0p8 = [
    [0.0,7.63888888889],
    [0.0008,5.09259259259],
    [0.0016,3.3950617284],
    [0.0024,2.2633744856],
    [0.0032,1.50891632373],
    [0.004,1.00594421582],
    [0.0048,0.670629477214],
    [0.0056,0.447086318143],
    [0.0064,0.298057545428],
    [0.0072,0.198705030286],
    [0.008,0.13247002019],
    [0.0088,0.0883133467936],
    [0.0096,0.0588755645291],
```

```
    [0.0104,0.0392503763527],
    [0.0112,0.0392503763527],
]
# transpose
pscad_dat_voltage_0p8 = map(list, zip(*pscad_dat_voltage_0p8))

# 0.1 ms Current
# ---------------------------------------
# Domain, Current
pscad_dat_current_0p1 = [
    [0.0,0.0362879238548],
    [0.0001,0.0832982690327],
    [0.0002,0.128015426641],
    [0.0003,0.170551259488],
    [0.0004,0.211012173659],
    [0.0005,0.2494993847],
    [0.0006,0.286109170812],
    [0.0007,0.320933113699],
    [0.0008,0.354058327665],
    [0.0009,0.385567677535],
    [0.001,0.415539985948],
    [0.0011,0.444050230536],
    [0.0012,0.471169731486],
    [0.0013,0.49696632995],
    [0.0014,0.521504557757],
    [0.0015,0.544845798842],
    [0.0016,0.567048442801],
    [0.0017,0.588168030957],
    [0.0018,0.608257395301],
    [0.0019,0.627366790652],
    [0.002,0.645544020376],
    [0.0021,0.662834555967],
    [0.0022,0.679281650798],
    [0.0023,0.69492644832],
    [0.0024,0.709808084988],
    [0.0025,0.723963788159],
    [0.0026,0.737428969224],
    [0.0027,0.750237312189],
    [0.0028,0.762420857936],
    [0.0029,0.774010084378],
    [0.003,0.785033982701],
    [0.0031,0.795520129886],
    [0.0032,0.805494757697],
    [0.0033,0.814982818297],
    [0.0034,0.824008046673],
    [0.0035,0.832593020006],
    [0.0036,0.840759214152],
    [0.0037,0.848527057364],
    [0.0038,0.855915981395],
    [0.0039,0.862944470107],
    [0.004,0.869630105712],
    [0.0041,0.87598961275],
    [0.0042,0.882038899933],
    [0.0043,0.887793099937],
    [0.0044,0.893266607257],
    [0.0045,0.89847311422],
    [0.0046,0.903425645233],
    [0.0047,0.908136589368],
    [0.0048,0.91261773135],
    [0.0049,0.916880281041],
    [0.005,0.920934901478],
    [0.0051,0.924791735552],
    [0.0052,0.928460431379],
    [0.0053,0.931950166433],
    [0.0054,0.93526967051],
    [0.0055,0.938427247558],
    [0.0056,0.941430796458],
    [0.0057,0.944287830777],
```

```
    [0.0058,0.947005497568],
    [0.0059,0.949590595248],
    [0.006,0.952049590602],
    [0.0061,0.954388634962],
    [0.0062,0.956613579598],
    [0.0063,0.95872999035],
    [0.0064,0.960743161552],
    [0.0065,0.962658129281],
    [0.0066,0.964479683951],
    [0.0067,0.966212382294],
    [0.0068,0.967860558768],
    [0.0069,0.969428336389],
    [0.007,0.970919637053],
    [0.0071,0.972338191343],
    [0.0072,0.973687547863],
    [0.0073,0.974971082113],
    [0.0074,0.976192004937],
    [0.0075,0.97735337055],
    [0.0076,0.978458084182],
    [0.0077,0.979508909344],
    [0.0078,0.980508474741],
    [0.0079,0.981459280852],
    [0.008,0.982363706176],
    [0.0081,0.983224013192],
    [0.0082,0.984042354012],
    [0.0083,0.984820775767],
    [0.0084,0.98556122573],
    [0.0085,0.986265556182],
    [0.0086,0.986935529051],
    [0.0087,0.987572820317],
    [0.0088,0.988179024204],
    [0.0089,0.98875565717],
    [0.009,0.989304161698],
    [0.0091,0.989825909908],
    [0.0092,0.990322206985],
    [0.0093,0.99079429445],
    [0.0094,0.991243353257],
    [0.0095,0.991670506757],
    [0.0096,0.9920768235],
    [0.0097,0.992463319915],
    [0.0098,0.992830962846],
    [0.0099,0.993180671975],
    [0.01,0.993513322123],
    [0.0101,0.993513322123],
]
# transpose
pscad_dat_current_0p1 = map(list, zip(*pscad_dat_current_0p1))

# 0.1 ms Voltage
# ---------------------------------------
# Domain, Voltage
pscad_dat_voltage_0p1 = [
    [0.0,9.63712076145],
    [0.0001,9.16701730967],
    [0.0002,8.71984573359],
    [0.0003,8.29448740512],
    [0.0004,7.88987826341],
    [0.0005,7.505006153],
    [0.0006,7.13890829188],
    [0.0007,6.79066886301],
    [0.0008,6.45941672335],
    [0.0009,6.14432322465],
    [0.001,5.84460014052],
    [0.0011,5.55949769464],
    [0.0012,5.28830268514],
    [0.0013,5.0303367005],
    [0.0014,4.78495442243],
    [0.0015,4.55154201158],
```

```
    [0.0016,4.32951557199],
    [0.0017,4.11831969043],
    [0.0018,3.91742604699],
    [0.0019,3.72633209348],
    [0.002,3.54455979624],
    [0.0021,3.37165444033],
    [0.0022,3.20718349202],
    [0.0023,3.0507355168],
    [0.0024,2.90191915012],
    [0.0025,2.76036211841],
    [0.0026,2.62571030776],
    [0.0027,2.49762687811],
    [0.0028,2.37579142064],
    [0.0029,2.25989915622],
    [0.003,2.14966017299],
    [0.0031,2.04479870114],
    [0.0032,1.94505242303],
    [0.0033,1.85017181703],
    [0.0034,1.75991953327],
    [0.0035,1.67406979994],
    [0.0036,1.59240785848],
    [0.0037,1.51472942636],
    [0.0038,1.44084018605],
    [0.0039,1.37055529893],
    [0.004,1.30369894288],
    [0.0041,1.2401038725],
    [0.0042,1.17961100067],
    [0.0043,1.12206900063],
    [0.0044,1.06733392743],
    [0.0045,1.0152688578],
    [0.0046,0.965743547665],
    [0.0047,0.918634106316],
    [0.0048,0.873822686496],
    [0.0049,0.831197189593],
    [0.005,0.790650985223],
    [0.0051,0.75208264448],
    [0.0052,0.715395686213],
    [0.0053,0.680498335666],
    [0.0054,0.647303294902],
    [0.0055,0.615727524419],
    [0.0056,0.585692035423],
    [0.0057,0.557121692231],
    [0.0058,0.529945024318],
    [0.0059,0.504094047522],
    [0.006,0.479504093984],
    [0.0061,0.456113650375],
    [0.0062,0.433864204015],
    [0.0063,0.412700096502],
    [0.0064,0.392568384478],
    [0.0065,0.373418707186],
    [0.0066,0.355203160494],
    [0.0067,0.337876177056],
    [0.0068,0.321394412321],
    [0.0069,0.30571663611],
    [0.007,0.290803629471],
    [0.0071,0.27661808657],
    [0.0072,0.263124521371],
    [0.0073,0.250289178865],
    [0.0074,0.238079950628],
    [0.0075,0.2264662945],
    [0.0076,0.215419158183],
    [0.0077,0.204910906564],
    [0.0078,0.194915252585],
    [0.0079,0.185407191484],
    [0.008,0.176362938241],
    [0.0081,0.167759868082],
    [0.0082,0.159576459883],
    [0.0083,0.151792242328],
```

```
    [0.0084,0.144387742702],
    [0.0085,0.13734443818],
    [0.0086,0.130644709488],
    [0.0087,0.124271796831],
    [0.0088,0.118209757961],
    [0.0089,0.112443428304],
    [0.009,0.106958383021],
    [0.0091,0.101740900922],
    [0.0092,0.0967779301457],
    [0.0093,0.0920570555044],
    [0.0094,0.087566467431],
    [0.0095,0.0832949324344],
    [0.0096,0.0792317649986],
    [0.0097,0.0753668008523],
    [0.0098,0.0716903715424],
    [0.0099,0.0681932802477],
    [0.01,0.0648667787722],
    [0.0101,0.0648667787722],
]
# transpose
pscad_dat_voltage_0p1 = map(list, zip(*pscad_dat_voltage_0p1))

# concatenate lists together
pscad_dat_current = [pscad_dat_current_0p1, pscad_dat_current_0p8]
pscad_dat_voltage = [pscad_dat_voltage_0p1, pscad_dat_voltage_0p8]

# Main function
def main(args):
    # Circuit parameters
    i_0 = 0
    v_0 = 10
    inductance_L = 0.02 # 20 mH
    resistance_R = 10 # 10 Ohms
    voltage_E = 10 # 10 Volts
    simulation_time = 0.01 # 10 ms
    time_deltas = [0.0001, 0.0008] # 100 us, 800 us time deltas

    print("Simulation time = %g" % simulation_time)
    print("Will run for the following time deltas:")
    for delta in time_deltas:
        print("Time delta = %g" % delta)

    cont_results = [[],[],[]]
    trap_results = [[[],[],[]]]
    back_results = [[[],[],[]]]
    # indices for the results tuple
    TIME_INDEX = 0
    VALUE_INDEX = 1
    CURRENT_INDEX = 1
    VOLTAGE_INDEX = 2

    # Calculate results using continuous solution
    print("Simulating using continuous solution")
    # Calculate 3000 points for the continuous solution
    for step in range(3000):
        time = 0.0 + (step * simulation_time/3000)
        # calculate continuous solution
        # current
        i_continuous_next = (10/resistance_R) - ((10/resistance_R)*math.exp(-(resistance_R/
            inductance_L)*time))
        # inductor voltage
        v_continuous_next = 10 * math.exp((-(resistance_R)/(inductance_L))*time)
        # append the result
        cont_results[TIME_INDEX].append(0+(step*simulation_time/3000))
        cont_results[CURRENT_INDEX].append(i_continuous_next)
        cont_results[VOLTAGE_INDEX].append(v_continuous_next)

    # Calculate results using trapezoidal discretization
```

```
    print("Simulating␣using␣trapezoidal␣discretization")
    for delta_index, delta in enumerate(time_deltas):
        sim_steps = int(simulation_time/delta)
        for step in range(sim_steps):
            if step == 0:
                # Use initial conditions
                i_prev = i_0
                v_prev = v_0
            else:
                # Use previous value
                i_prev = trap_results[delta_index][CURRENT_INDEX][step - 1]
                v_prev = trap_results[delta_index][VOLTAGE_INDEX][step - 1]
            # perform trapezoidal discretization step
            # current
            i_approx_next = ((i_prev*((2*inductance_L) - (resistance_R*delta))) + (20 *
                delta))/((resistance_R*delta)+(2*inductance_L))
            # voltage
            v_approx_next = (((2*inductance_L)/(delta))*(i_approx_next - i_prev)) - v_prev
            # append the result
            trap_results[delta_index][TIME_INDEX].append(0+(step*delta))
            trap_results[delta_index][CURRENT_INDEX].append(i_approx_next)
            trap_results[delta_index][VOLTAGE_INDEX].append(v_approx_next)
        # append a new set of results for the next delta
        trap_results.append([[],[],[]])

    # Calculate results using backwards euler discretization
    print("Simulating␣using␣backward␣euler␣discretization")
    for delta_index, delta in enumerate(time_deltas):
        sim_steps = int(simulation_time/delta)
        for step in range(sim_steps):
            if step == 0:
                # Use initial conditions
                i_prev = i_0
            else:
                # Use previous value
                i_prev = back_results[delta_index][CURRENT_INDEX][step - 1]
            # perform backward eurler discretization step
            # current
            i_approx_next = ((inductance_L*i_prev)+(10*delta))/((resistance_R*delta) +
                inductance_L)
            # voltage
            v_approx_next = ((inductance_L)/(delta))*(i_approx_next - i_prev)
            # append the result
            back_results[delta_index][TIME_INDEX].append(0+(step*delta))
            back_results[delta_index][CURRENT_INDEX].append(i_approx_next)
            back_results[delta_index][VOLTAGE_INDEX].append(v_approx_next)
        # append a new set of results for the next delta
        back_results.append([[],[],[]])

    # Plots for publication
    # Comparisons between all techniques
    for delta_index, delta in enumerate(time_deltas):
        # Current plot
        fig, ax = plt.subplots(2)
        ax[0].plot(cont_results[TIME_INDEX], cont_results[CURRENT_INDEX], linestyle='dotted'
            , label='Continuous')
        ax[0].plot(trap_results[delta_index][TIME_INDEX], trap_results[delta_index][
            CURRENT_INDEX], label='Trapezoidal')
        ax[0].plot(back_results[delta_index][TIME_INDEX], back_results[delta_index][
            CURRENT_INDEX], label='Backward␣Euler␣␣␣␣')
        ax[0].plot(pscad_dat_current[delta_index][TIME_INDEX], pscad_dat_current[delta_index
            ][VALUE_INDEX], label='PSCAD')
        ax[0].set(xlabel='time␣(s)', ylabel='current␣(A)', title='Evaluations␣of␣$i(t)$␣at␣$
            %gs$' % delta)
        ax[0].legend(loc='best', fancybox=True, shadow=True)
        ax[0].grid()
        # Voltage plot
        ax[1].plot(cont_results[TIME_INDEX], cont_results[VOLTAGE_INDEX], linestyle='dotted'
```

```
            , label='Continuous')
        ax[1].plot(trap_results[delta_index][TIME_INDEX], trap_results[delta_index][
            VOLTAGE_INDEX], label='Trapezoidal')
        ax[1].plot(back_results[delta_index][TIME_INDEX], back_results[delta_index][
            VOLTAGE_INDEX], label='Backward Euler    ')
        ax[1].plot(pscad_dat_voltage[delta_index][TIME_INDEX], pscad_dat_voltage[delta_index
            ][VALUE_INDEX], label='PSCAD')
        ax[1].set(xlabel='time (s)', ylabel='voltage (V)', title='Evaluations of $v_L(t)$ at
            $%gs$' % delta)
        ax[1].legend(loc='best', fancybox=True, shadow=True)
        ax[1].grid()
        fig.set_size_inches(6.5,8)
        fig.tight_layout()
        # plt.show()
        fig.savefig('compare_plot_' + str(delta).replace('.','p') + '.pgf')
        fig.savefig('compare_plot_' + str(delta).replace('.','p') + '.png')

if __name__ == '__main__':
    # the following sets up the argument parser for the program
    parser = argparse.ArgumentParser(description='Assignment 1a solution generator')

    args = parser.parse_args()

    main(args)
```

5.2 Fortran Code Listing

The following is the code generated by PSCAD to simulate the circuit for this assignment.

```
!=========================================================================
! Generated by  : PSCAD v4.6.3.0
!
! Warning:  The content of this file is automatically generated.
!           Do not modify, as any changes made here will be lost!
!-------------------------------------------------------------------------
! Component     : Main
! Description   :
!-------------------------------------------------------------------------


!=========================================================================

      SUBROUTINE MainDyn()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 'emtstor.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's2.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'fnames.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'matlab.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------
```

```
!---------------------------------------
! Variable Declarations
!---------------------------------------


! Subroutine Arguments

! Electrical Node Indices

! Control Signals
      REAL     I, VL

! Internal Variables
      REAL     RVD1_1, RVD1_2, RVD1_3, RVD1_4

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER ISTOF, IT_0                       ! Storage Indices
      INTEGER SS, INODE, IBRCH                  ! SS/Node/Branch/Xfmr


!---------------------------------------
! Local Indices
!---------------------------------------

! Dsdyn <-> Dsout transfer index storage

      NTXFR = NTXFR + 1

      TXFR(NTXFR,1) = NSTOL
      TXFR(NTXFR,2) = NSTOI
      TXFR(NTXFR,3) = NSTOF
      TXFR(NTXFR,4) = NSTOC

! Define electric network subsystem number

      SS        = NODE(NNODE+1)

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      ISTOF     = NSTOF
      NSTOF     = NSTOF + 2
      NPGB      = NPGB + 2
      INODE     = NNODE + 2
      NNODE     = NNODE + 4
      IBRCH     = NBRCH(SS)
      NBRCH(SS) = NBRCH(SS) + 3
      NCSCS     = NCSCS + 0
      NCSCR     = NCSCR + 0

!---------------------------------------
! Transfers from storage arrays
!---------------------------------------

      I        = STOF(ISTOF + 1)
      VL       = STOF(ISTOF + 2)


!---------------------------------------
! Electrical Node Lookup
!---------------------------------------
```

```
!---------------------------------------
! Configuration of Models
!---------------------------------------

      IF ( TIMEZERO ) THEN
         FILENAME = 'Main.dta'
         CALL EMTDC_OPENFILE
         SECTION = 'DATADSD:'
         CALL EMTDC_GOTOSECTION
      ENDIF
!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 1:[source_1] Single Phase Voltage Source Model 2 'Source1'
!  DC source: Type: Ideal
      RVD1_1 = RTCF(NRTCF)
      RVD1_2 = RTCF(NRTCF+1)
      RVD1_3 = RTCF(NRTCF+2)
      RVD1_4 = RTCF(NRTCF+3)
      NRTCF = NRTCF + 4
      CALL EMTDC_1PVSRC(SS, (IBRCH+2),RVD1_4,0,RVD1_1,RVD1_2,RVD1_3)

!---------------------------------------
! Feedbacks and transfers to storage
!---------------------------------------

      STOF(ISTOF + 1) = I
      STOF(ISTOF + 2) = VL


!---------------------------------------
! Transfer to Exports
!---------------------------------------

!---------------------------------------
! Close Model Data read
!---------------------------------------

      IF ( TIMEZERO ) CALL EMTDC_CLOSEFILE
      RETURN
      END

!=======================================================================

      SUBROUTINE MainOut()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 'emtstor.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's2.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'fnames.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'matlab.h'
      INCLUDE 'rtconfig.h'
```

```
!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------

      REAL    EMTDC_VVDC    !

!---------------------------------------
! Variable Declarations
!---------------------------------------


! Electrical Node Indices
      INTEGER  NT_3

! Control Signals
      REAL     I, VL

! Internal Variables
      INTEGER  IVD1_1

! Indexing variables
      INTEGER ICALL_NO                              ! Module call num
      INTEGER ISTOL, ISTOI, ISTOF, ISTOC, IT_0      ! Storage Indices
      INTEGER IPGB                                  ! Control/Monitoring
      INTEGER SS, INODE, IBRCH                      ! SS/Node/Branch/Xfmr


!---------------------------------------
! Local Indices
!---------------------------------------

! Dsdyn <-> Dsout transfer index storage

      NTXFR = NTXFR + 1

      ISTOL = TXFR(NTXFR,1)
      ISTOI = TXFR(NTXFR,2)
      ISTOF = TXFR(NTXFR,3)
      ISTOC = TXFR(NTXFR,4)

! Define electric network subsystem number

      SS        = NODE(NNODE+1)

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      IPGB      = NPGB
      NPGB      = NPGB + 2
      INODE     = NNODE + 2
      NNODE     = NNODE + 4
      IBRCH     = NBRCH(SS)
      NBRCH(SS) = NBRCH(SS) + 3
      NCSCS     = NCSCS + 0
      NCSCR     = NCSCR + 0

!---------------------------------------
! Transfers from storage arrays
!---------------------------------------

      I        = STOF(ISTOF + 1)
      VL       = STOF(ISTOF + 2)
```

```
!---------------------------------------
! Electrical Node Lookup
!---------------------------------------

      NT_3  = NODE(INODE + 2)

!---------------------------------------
! Configuration of Models
!---------------------------------------

      IF ( TIMEZERO ) THEN
         FILENAME = 'Main.dta'
         CALL EMTDC_OPENFILE
         SECTION = 'DATADSO:'
         CALL EMTDC_GOTOSECTION
      ENDIF
!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 10:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 5
      I = (-CBR((IBRCH+1), SS))
      VL = EMTDC_VVDC(SS, NT_3, 0)

! 20:[pgb] Output Channel 'Current'

      PGB(IPGB+1) = I

! 30:[pgb] Output Channel 'Voltage'

      PGB(IPGB+2) = VL

!---------------------------------------
! Feedbacks and transfers to storage
!---------------------------------------

      STOF(ISTOF + 1) = I
      STOF(ISTOF + 2) = VL


!---------------------------------------
! Close Model Data read
!---------------------------------------

      IF ( TIMEZERO ) CALL EMTDC_CLOSEFILE
      RETURN
      END

!=======================================================================

      SUBROUTINE MainDyn_Begin()

!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'rtconfig.h'
```

```
!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------


!---------------------------------------
! Variable Declarations
!---------------------------------------


! Subroutine Arguments

! Electrical Node Indices

! Control Signals

! Internal Variables

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER IT_0                              ! Storage Indices
      INTEGER SS, INODE, IBRCH                  ! SS/Node/Branch/Xfmr


!---------------------------------------
! Local Indices
!---------------------------------------


! Define electric network subsystem number

      SS        = NODE(NNODE+1)

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      INODE     = NNODE + 2
      NNODE     = NNODE + 4
      IBRCH     = NBRCH(SS)
      NBRCH(SS) = NBRCH(SS) + 3
      NCSCS     = NCSCS + 0
      NCSCR     = NCSCR + 0

!---------------------------------------
! Electrical Node Lookup
!---------------------------------------


!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 1:[source_1] Single Phase Voltage Source Model 2 'Source1'
      CALL E_1PVSRC_CFG(0,0,6,10.0,60.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0)

      RETURN
      END

!=======================================================================

      SUBROUTINE MainOut_Begin()
```

```
!---------------------------------------
! Standard includes
!---------------------------------------

      INCLUDE 'nd.h'
      INCLUDE 'emtconst.h'
      INCLUDE 's0.h'
      INCLUDE 's1.h'
      INCLUDE 's4.h'
      INCLUDE 'branches.h'
      INCLUDE 'pscadv3.h'
      INCLUDE 'radiolinks.h'
      INCLUDE 'rtconfig.h'

!---------------------------------------
! Function/Subroutine Declarations
!---------------------------------------


!---------------------------------------
! Variable Declarations
!---------------------------------------


! Subroutine Arguments

! Electrical Node Indices
      INTEGER  NT_3

! Control Signals

! Internal Variables
      INTEGER  IVD1_1

! Indexing variables
      INTEGER ICALL_NO                          ! Module call num
      INTEGER IT_0                              ! Storage Indices
      INTEGER SS, INODE, IBRCH                  ! SS/Node/Branch/Xfmr


!---------------------------------------
! Local Indices
!---------------------------------------


! Define electric network subsystem number

      SS        = NODE(NNODE+1)

! Increment and assign runtime configuration call indices

      ICALL_NO  = NCALL_NO
      NCALL_NO  = NCALL_NO + 1

! Increment global storage indices

      INODE     = NNODE + 2
      NNODE     = NNODE + 4
      IBRCH     = NBRCH(SS)
      NBRCH(SS) = NBRCH(SS) + 3
      NCSCS     = NCSCS + 0
      NCSCR     = NCSCR + 0

!---------------------------------------
! Electrical Node Lookup
!---------------------------------------

      NT_3  = NODE(INODE + 2)
```

```
!---------------------------------------
! Generated code from module definition
!---------------------------------------


! 10:[multimeter] Multimeter
      IVD1_1 = NRTCF
      NRTCF  = NRTCF + 5

! 20:[pgb] Output Channel 'Current'

! 30:[pgb] Output Channel 'Voltage'

      RETURN
      END
```