Newton's Approximation in Nonlinear Circuits

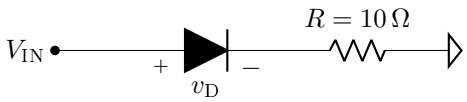
Michael You

January 30, 2019

With linear circuits, we can often get by pretty easily with Ohm's law, KVL, KCL, etc... But once we start talking about non-linear circuit elements, such as a diode, it seems like we are at a lost for tools to use.

1 The Problem

Let's consider the following circuit



Suppose that

- $V_{\text{IN}} = 8\,\text{V}$
- $i_{\text{D}} = I_s\left(e^{v_{\text{D}}/v_{\text{T}}} - 1\right)$, where $I_s = 1 \times 10^{-24}\,\text{A}$, $v_{\text{T}} = 25\,\text{mV}$

Solve for the current through the diode.

2 A Naïve Approach

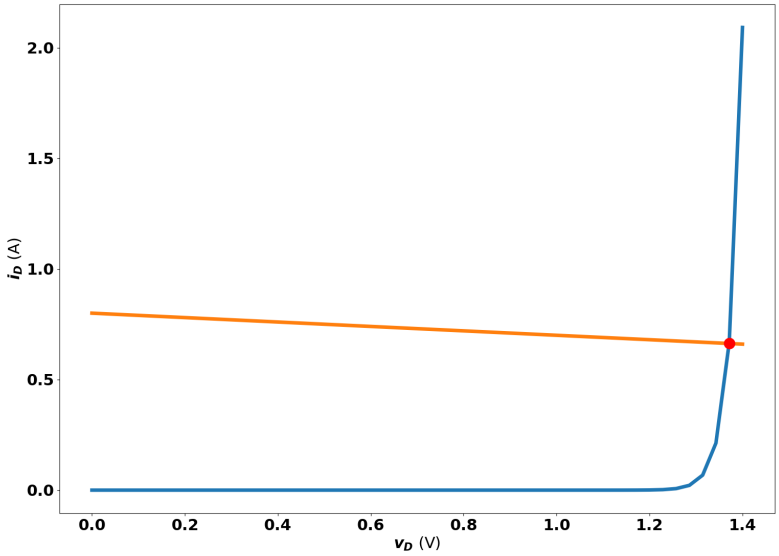
Hey, KCL works most of the time, so let's try it...

$$\begin{aligned} i_{\text{D}} &= \frac{V_{\text{IN}} - v_{\text{D}}}{R} \\ I_s\left(e^{v_{\text{D}}/v_{\text{T}}} - 1\right) &= \frac{V_{\text{IN}} - v_{\text{D}}}{R} \\ I_s e^{v_{\text{D}}/v_{\text{T}}} + \frac{v_{\text{D}}}{R} &= \frac{V_{\text{IN}}}{R} + I_s \end{aligned}$$

Oh man, it doesn't look like v_{D} is easy to solve here...so what do we do here?

3 Graph it!

If you recall from Algebra, if we have two unknowns and two equations, we can find the intersection of the two curves to find their solution. We don't know $v_\mathrm{D}, i_\mathrm{D}$, and we have equations for each, so why not graph them?



So yeah, this works pretty well, and we can find a solution at $(v_\mathrm{D}, i_\mathrm{D}) = (1.371\,\mathrm{V}, 0.663\,\mathrm{A})$, but it can cause a lot of unnecessary strain on your calculator because it has to plot so many points. In addition, how did the calculator find the points in the first place?

4 Newton's Method

In fact, even the calculator had to use some sort of numerical method to approximate the intersection of the two curves! One popular and fairly reliable method is:

Definition 1. Newton's Method is a first-order numerical approximation method to estimate the root of a function. In particular, given

$$f(x) = 0 \tag{1}$$
$$x_0 = \text{initial guess} \tag{2}$$

iterate

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

until $|f(x_n)| < \epsilon$, for some sufficiently small $\epsilon > 0$ that defines the desired accuracy.

For our problem, we can define

$$f(v_\mathrm{D}) = I_S\left(e^{v_\mathrm{D}/v_\mathrm{T}} - 1\right) - \frac{V_\mathrm{IN} - v_\mathrm{D}}{R} = 0 \tag{4}$$

We also need the derivative,

$$\frac{\mathrm{d}f(v_\mathrm{D})}{\mathrm{d}v_\mathrm{D}} = \frac{I_S}{v_\mathrm{T}} e^{v_\mathrm{D}/v_\mathrm{T}} + \frac{1}{R} \tag{5}$$

Now, we just need to iterate with the following iterative sequence:

$$\begin{aligned} v_{\mathrm{D}_0} &= v_{\mathrm{D}_\mathrm{initial}} \\ v_{\mathrm{D}_{i+1}} &= v_{\mathrm{D}_i} - \frac{f(v_{\mathrm{D}_i})}{f'(v_{\mathrm{D}_i})} \end{aligned} \tag{6}$$

Choosing an initial guess of $v_{\mathrm{D}_0} = 1.3\,\mathrm{V}$ gives us a final convergence value of

$$\boxed{(v_\mathrm{D}, i_\mathrm{D}) = (1.371\,\mathrm{V}, 0.663\,\mathrm{A})} \tag{7}$$

A Implementation of Newton's Method

In Python because MATLAB is dying...

```
'''
18220 Newton's Approximation Demo

Michael You
'''
import numpy as np
import matplotlib.pyplot as plt
from math import log, exp

# CONSTANTS
CONST_IS = 1e-24
CONST_VT = 0.025
CONST_VIN = 8
CONST_R = 10

CONST_THRESHOLD = 1e-6
CONST_INIT_GUESS = 1.3

def func_iD_1 (vD, R=CONST_R, VIN=CONST_VIN, IS=CONST_IS, VT=CONST_VT):
  return IS*(exp(vD/VT) - 1)

def func_iD_2 (vD, R=CONST_R, VIN=CONST_VIN, IS=CONST_IS, VT=CONST_VT):
  return (VIN - vD)/(R)

def func_iD (vD, R=CONST_R, VIN=CONST_VIN, IS=CONST_IS, VT=CONST_VT):
  return (
    IS*(exp(vD/VT) - 1) - (VIN - vD)/(R)
  )

def func_iD_D (vD, R=CONST_R, VIN=CONST_VIN, IS=CONST_IS, VT=CONST_VT):
  return (
    (IS/VT)*exp(vD/VT) + (1/(R))
  )

def newtons(func, funcD, initGuess, THRESHOLD=CONST_THRESHOLD):
  prevVal = initGuess
  currVal = prevVal - func(prevVal)/funcD(prevVal)

  while abs(currVal - prevVal) > THRESHOLD:
    print('Iterating', currVal)
    prevVal = currVal
    currVal = prevVal - func(prevVal)/funcD(prevVal)

  return currVal

if __name__ == "__main__":
  print('### Beginning Newton\'s Approximation ###')
  print('Init Guess, v_D =', CONST_INIT_GUESS)

  ans = newtons(func_iD, func_iD_D, CONST_INIT_GUESS)
```

```
print('v_D converged at', ans, end='\n\n')
print('Solution (v_D, i_D) = ({0}, {1})'.format(ans, func_iD_1(ans)), end='\n\n')

### PLOT CURVES AND INTERSECTION ###
print('### PLOTTING ###')
CONST_LINEWIDTH = 5

vD = np.linspace(0, 1.4)
f1 = np.vectorize(lambda vD: func_iD_1(vD))
f2 = np.vectorize(lambda vD: func_iD_2(vD))
iD1 = f1(vD)
iD2 = f2(vD)

plt.rcParams.update({'font.weight': 'bold'})
plt.rcParams.update({'font.size': 22})

plt.figure()
plt.plot(vD, iD1, linewidth=CONST_LINEWIDTH)
plt.plot(vD, iD2, linewidth=CONST_LINEWIDTH)
plt.plot(ans, func_iD_1(ans), marker='o', markersize=15, color="red")
plt.xlabel('$v_D$ (V)')
plt.ylabel('$i_D$ (A)')

plt.show()
```

newtons.py