Practical Interactive Learning Tools for Scientific Computing and Data Analysis Using R

Cindy Nguyen, Dr. Reinaldo Sanchez-Arias

Department of Data Science and Business Analytics, Florida Polytechnic University

Motivation and Background

Nowadays, with data and models playing a prevalent role in science and engineering fields, learning skills in scientific computing and data analysis, is key to increase efficiency in decision making and modeling.

A ***Jupyter Notebook*** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
Jupyter notebooks were designed to assist in the exploration and understanding of methods in ***scientific computing*** and ***data analysis***, allowing the user to interactively run code with implementation of root-finding and optimization techniques and applications.

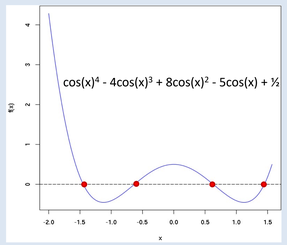

Figure 1: Four possible roots that can be found using methods implemented in our tool.

Root-Finding refers to a mathematics and computing problem of finding the zeros, or ***roots***, of a given continuous function. We explore numerical root-finding iterative methods that produce a sequence of numbers that converges towards the root of the function.

Many data-driven scientific problems involve formulating and optimizing objective functions for which a ***minimum*** or ***maximum*** is sought.
We introduce optimization methods, explain the relationship with root-finding problems, and show how to use functions in R to solve optimization problems.
Applications in ***parameter estimation*** and ***data analysis*** are motivated and showcased through interactive examples.

Components of the `rjupyternb` Tool

Root-Finding Notebooks

a. Theory and examples: Newton's Method and Bisection
b. Interest rate and power cable applications
c. Chemical, Civil, and Electrical Engineering applications

Optimization Notebooks

a. Theory and examples
b. Formulating mathematical optimization problems
c. Curve-fitting application
d. Multi-dimensional scaling application

Root-Finding Notebook Examples

- Root-finding is used to find the zeros of a continuous function. The zeros of these functions are also known as *roots*. In the collection of Jupyter notebooks we cover three methods: Newton-Raphson, Bisection, and Fixed-Point Iteration.

```
ncn_newtonraphson <- function(ftn, x0, tol = 1e-9, max.iter = 100) {

# This implementation assumes that the user provides ftn as a string that
# depends on x. The initial guess of the root is x0
# the algorithm terminates when the function value is within distance
# tol of 0, or the number of iterations exceeds max.iter

 myf <- eval(parse(text = paste(

       'f <- function(x) {

       return(c(',ftn,', eval(D(expression(',ftn,'), "x"))))

       }', sep='')))

 # initialize
 x <- x0
 fx <- myf(x)
 iter <- 0

 # continue iterating until stopping conditions are met
 while ((abs(myf(x)[1]) > tol) && (iter < max.iter)) {
   x <- x - myf(x)[1]/myf(x)[2]
   fx <- myf(x)
   iter <- iter + 1
   cat("At iteration", iter, "value of x is:", x, "\n")
 }

 # output depends on success of algorithm
 if (abs(myf(x)[1]) > tol) {
   cat("Algorithm failed to converge\n")
   return(NULL)
 } else {
   cat("Algorithm converged\n")
   return(x)
 }
}
```

Figure 2: Implementation of Newton-Raphson method for root-finding of one-dimensional functions

The `ncn_newtonraphson()` function does not require the user to input the derivative of the function, since it makes use of *symbolic differentiation* in R.

```
test_function <- expression(sin(h)^4 + 3*cos(h)^2)
D(test_function, "h")

4 * (cos(h) * sin(h)^3) - 3 * (2 * (sin(h) * cos(h)))
```

Figure 3: Symbolic differentiation in R using the `D()` function

```
ncn_newtonraphson(ftn= "cos(x)^4 - 4*cos(x)^3 + 8*cos(x)^2 - 5*cos(x) + 1/2",
                  x0 = 2)

At iteration 1 value of x is: 1.664038
At iteration 2 value of x is: 1.505978
At iteration 3 value of x is: 1.453879
At iteration 4 value of x is: 1.44793
At iteration 5 value of x is: 1.447855
At iteration 6 value of x is: 1.447855
Algorithm converged

1.44785543586895
```

Figure 4: Solution found using the Newton-Raphson implementation for the function in *Figure 1*

- Examples of one-dimensional root-finding applications are presented in the notebooks, including approximating the interest rate for a personal loan, calculating the length of a power cable, and other engineering problems, comparing the methods introduced for root-finding.

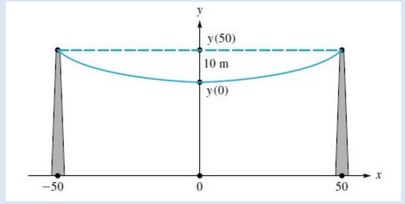

Figure 5: Electric power cable suspended from two towers that are 100 meters apart. The cable is allowed to dip 10 meters in the middle.

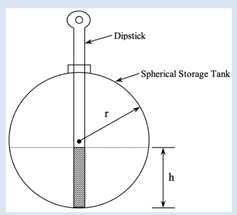

Figure 6: Spherical storage tank containing oil. The tank has a diameter of 6ft

Example: It is known that the curve assumed by a suspended cable is a catenary. When the y-axis passes through the lowest point: $y=\lambda\cosh(x/\lambda)$, with λ a parameter that can be estimated [2] by solving the nonlinear equation:

$$\lambda\cosh(50/\lambda)=\lambda+10$$

Example: Suppose you are asked to calculate the height h to which a dipstick 8 ft long would be wet with oil when immersed in the tank when it contains 6ft^3 of oil. The height h of the liquid on the spherical tank for the given volume and radius can be found by solving [3]:

$$f(h) = h^3 - 9h^2 + 3.8197 = 0$$

Optimization Notebook Examples

- Optimization in mathematics refers to finding the minimum or maximum of a function, which is often represented by a range of choices that are feasible. Different types of optimization problems are introduced in notebooks including one-dimensional and multivariate optimization, and unconstrained and constrained optimization. Examples on how to use the function `optim()` in R for optimization problems, and implementations of objective functions for applications are explained.

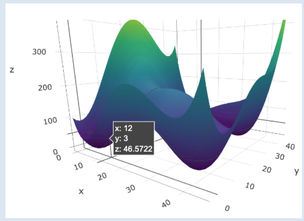

Figure 7: Interactive surface plot that the user can rotate in any direction to explore the function. The user can hover over the model to view the (*x*, *y*, *z*) coordinates and zoom in/out in regions

- *Curve Fitting for Nonlinear Regression*

Figure 8: Curve fitting example. The blue line is the solution using a sum of squares objective function (l2 fit). The red line uses the sum of absolute differences (l1 fit)

Given observations $(x_1, y_1),\ldots,(x_n, y_n)$, we aim to find the ***parameters*** for a function f such that $y_i \approx f(x_i)$. A popular model for the plant size (measured by volume of the trunk) as a function of age is the Richards curve:

$$f(x) = a(1 - e^{-bx})^c$$

Using data from a spruce tree at different ages [1], we implement objective functions to solve an optimization problem for estimating the vector of parameters $\theta = (a, b, c)^T$

- *Multidimensional Scaling (MDS)*

MDS is typically applied to a ***dissimilarity matrix***, such as one representing distances between cities. The challenge is to produce coordinates for each city such that we *minimize the squared error* between Euclidean distances of the coordinates and the entries in a distance matrix [4].

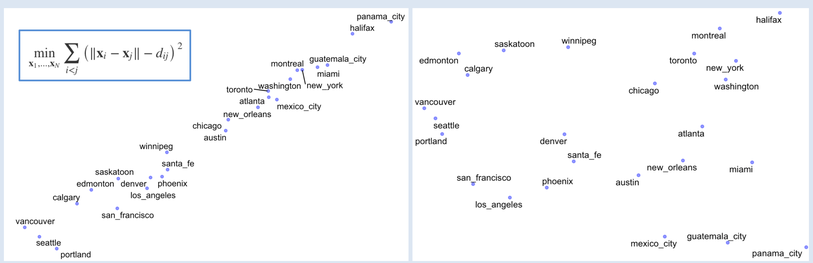

Figure 9: (left) Initial guess for coordinates $W_0 = (u_1, u_2)^T$ of $N = 25$ cities, and MDS objective function. (right) Optimal coordinates $W^* = (u_1^*, u_2^*)^T$ found by minimizing the objective function using `optim()`. Notice that in this case an optimization problem with 50 unknowns is solved.

Accessing the `rjupyternb` Tool

Title of tool: `rjupyternb`
URL: https://nanohub.org/tools/rjupyternb

Future Directions

1. Work on globalization approaches to improve Newton's method implementation (such as line search and trust-region).
2. Apply the curve fitting framework to other applications in science and engineering.
3. Incorporate additional optimization problem formulations for machine learning methods.

References

1. Jones, O., Maillardet, R., & Robinson, A. (2014). *Introduction to scientific programming and simulation using R*. CRC Press.
2. Cheney, E. W., & Kincaid, D. R. (2012). *Numerical mathematics and computing*. Cengage Learning.
3. Kaw, A., Kalu, E. (2009) *Numerical methods with applications: Abridged*. Lulu.
4. Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Series in Statistics.

This work is funded by the National Science Foundation, Network for Computational Nanotechnology Cyberplatform, Award EEC-1227110. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Cindy Nguyen, Dr. Reinaldo Sanchez-Arias
Department of Data Science and Business Analytics, Florida Polytechnic University

- Root-finding is used to find the zeros of a continuous function. The zeros of these functions are also known as *roots*. In the collection of Jupyter notebooks we cover three methods: Newton-Raphson, Bisection, and Fixed-Point Iteration.

```
ncn_newtonraphson <- function(ftn, x0, tol = 1e-9, max.iter = 100) {

# This implementation assumes that the user provides ftn as a string that
# depends on x. The initial guess of the root is x0
# the algorithm terminates when the function value is within distance
# tol of 0, or the number of iterations exceeds max.iter

  myf <- eval(parse(text = paste(

        'f <- function(x) {

        return(c(',ftn,', eval(D(expression(',ftn,'), "x"))))

        }', sep='')))

  # initialize
  x <- x0
  fx <- myf(x)
  iter <-  0

  # continue iterating until stopping conditions are met
  while ((abs(myf(x)[1]) > tol) && (iter < max.iter)) {
    x <- x - myf(x)[1]/myf(x)[2]
    fx <- myf(x)
    iter <- iter + 1
    cat("At iteration", iter, "value of x is:", x, "\n")
  }

  # output depends on success of algorithm
  if (abs(myf(x)[1]) > tol) {
    cat("Algorithm failed to converge\n")
    return(NULL)
  } else {
    cat("Algorithm converged\n")
    return(x)
  }
}
```

Figure 2: Implementation of Newton-Raphson method for root-finding of one-dimensional functions

```
test_function <- expression(sin(h)^4 + 3*cos(h)^2)
D(test_function, "h")
```

```
4 * (cos(h) * sin(h)^3) - 3 * (2 * (sin(h) * cos(h)))
```

Figure 3: Symbolic differentiation in R using the `D()` function

```
ncn_newtonraphson(ftn= "cos(x)^4 - 4*cos(x)^3 + 8*cos(x)^2 - 5*cos(x) + 1/2",
                  x0 = 2)
```

```
At iteration 1 value of x is: 1.664038
At iteration 2 value of x is: 1.505978
At iteration 3 value of x is: 1.453879
At iteration 4 value of x is: 1.44793
At iteration 5 value of x is: 1.447855
At iteration 6 value of x is: 1.447855
Algorithm converged

1.44785543586895
```

Figure 4: Solution found using the Newton-Raphson implementation for the function in *Figure 1*

This work is funded by the National Science Foundation, Network for Computational Nanotechnology Cyberplatform, Award EEC-1227110. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Cindy Nguyen, Dr. Reinaldo Sanchez-Arias
Department of Data Science and Business Analytics, Florida Polytechnic University

Example: It is known that the curve assumed by a suspended cable is a catenary. When the y-axis passes through the lowest point: $y=\lambda\cosh(x/\lambda)$, with λ a parameter that can be estimated [2] by solving the nonlinear equation:

$$\lambda\cosh(50/\lambda)=\lambda+10$$

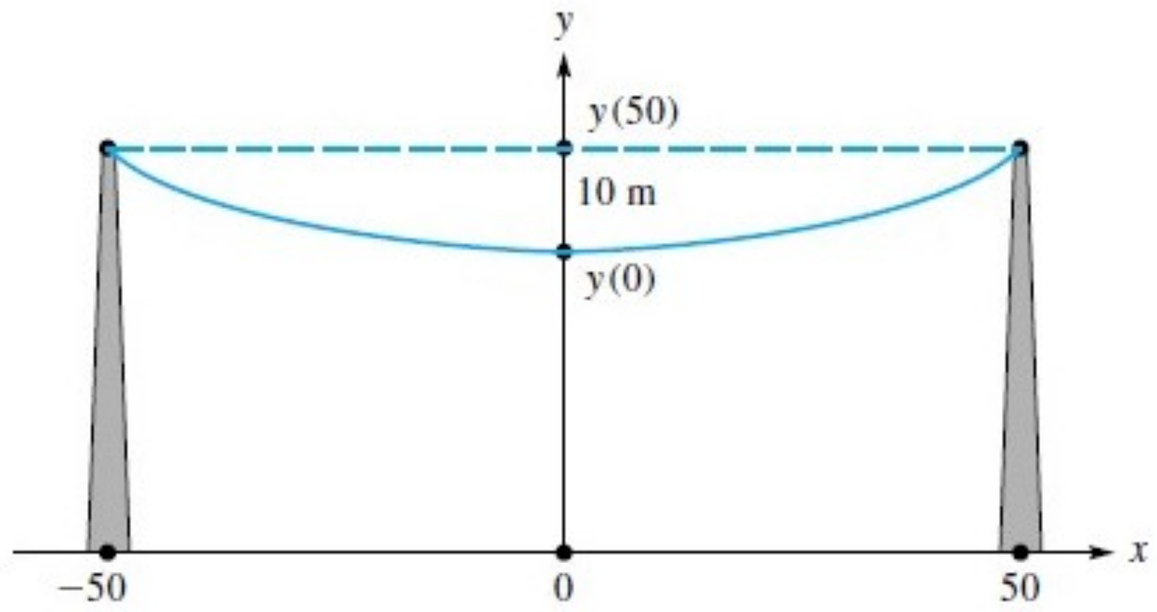

Figure 5: Electric power cable suspended from two towers that are 100 meters apart. The cable is allowed to dip 10 meters in the middle.

```
In [17]: ncn_newtonraphson(ftn= "x * cosh(50/x) - x - 10", x0 = 100)

At iteration 1 value of x is: 120.7836
At iteration 2 value of x is: 126.3517
At iteration 3 value of x is: 126.6318
At iteration 4 value of x is: 126.6324
Algorithm converged

126.632436036565
```


This work is funded by the National Science Foundation, Network for Computational Nanotechnology Cyberplatform, Award EEC-1227110. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.