

Numerical Solution of a Variation Problem

Daniel Paliura

3/29/2021

Contents

- Introduction
- Problem Formulation
- The Task
- Provided Solution
- Result
- Analytical Solution
- Comparisons

Introduction

This document was created in addition to ‘Control Theory’ subject third laboratory work ‘Numerical Solution of a Variation Problem’. This document on GitHub is [here](#). It contains the performance of the task according to variant.

Problem Formulation

Variation Problem is an optimization problem with infinite dimensions in the set of continuous differentiated functions with fixed bounds.

$$J(x(\cdot)) = \int_{\alpha}^{\beta} I(t, x(t), \dot{x}(t)) dt \rightarrow \min_{x(\cdot): x(\alpha)=a, x(\beta)=b}$$

Let approximate this problem next way:

Split interval $[\alpha, \beta]$ into N subintervals of same length by defining $N+1$ time points $t_i = \alpha + i \cdot \Delta t, i = 0 \dots N$, where $\Delta t = \frac{\beta - \alpha}{N}$. Define $N-1$ variables as $x_i = x(t_i), i = 1 \dots N-1$ and $x_0 = a, x_N = b$ are constant.

Then let approximate integral from Variation Problem by the Trapezoidal Formula

$$J(x(\cdot)) \approx \sum_{i=0}^{N-1} I(t_i, x_i, \dot{x}(t_i)) \Delta t$$

And derivative $\dot{x}(t_i)$ can be also approximated with a finite difference: $\dot{x}(t_i) \approx \frac{(x_{i+1} - x_i)}{\Delta t}$. And so let

$$J(x(\cdot)) \approx J_N(x_1, \dots, x_{N-1}) = \sum_{i=0}^{N-1} I(t_i, x_i, \frac{(x_{i+1} - x_i)}{\Delta t}) \Delta t \rightarrow \min_{x_1, \dots, x_{N-1}}$$

The optimization problem at infinite dimensions led to an optimization problem at finite dimensions. Also can be solved the regularized problem:

$$J_N^\lambda(x_1, \dots, x_{N-1}) = J_N(x_1, \dots, x_{N-1}) + \lambda \sum_{i=0}^{N-1} \left(\frac{x_{i+1} - x_i}{\Delta t} \right)^2 \Delta t$$

Where $\lambda \geq 0$ is regularization parameter.

The Task

- 1) Chosen example (variant 1 as 15-2*7) of variation problem to solve: $\int_0^1 ((x')^2 + x^2) dt \rightarrow \min, x_0 = x(0) = 0, x_N = x(1) = 1$.
- 2) To choose number of approximation points N .
- 3) To code function $J_N(x_1, \dots, x_{N-1})$.
- 4) To choose optimization method (non-gradient or gradient) for minimization $J_N(x_1, \dots, x_{N-1})$.
- 5) To apply chosen method to minimize $J_N(x_1, \dots, x_{N-1})$ and build plot of function value dependence on iteration number.
- 6) To visualize got solution as part-line plot by points $(t_i, x_i), i = 0 \dots N$.
- 7) To compare visually (by plots) got numeric solution and analytic one.
- 8) To research approximation accuracy dependently on number of approximation points N .
- 9) To prepare the report.

Provided Solution

To solve problem, I wrote simple code using R language, that appears [thereon GitHub](#).

Solution provided with the function

```
# Solves variation problem where
# I = function(t, x, dx) - underintegral function:
#       time t, value x (x=x(t)), and derivative dx (dx = dx/dt (t))
# t1, t2 - limits for integral
# x1, x2 - values for x at moments t1 and t2 (x1 = x(t1), x2 = x(t2))
# N - Number of intervals for splitting the time interval [t1, t2].
# lambda - regularization parameter. If not NULL, then problem solved
#       as regularized with value lambda >= 0
# method - optimization method passed to optim() function as cognominal argument
# ... - other arguments passed to optim() function,
#       for example, control=list(trace=3)
var.prb(I, t0, t1, x0, x1, N=50, lambda = NULL, ... , method = 'BFGS')
```

This function returns S3 class named `VariationProblemSolution` and so generic methods developed for plotting and printing result.

So I chose $N=50$ and set it as the default value into the developed function. As the method, I chose the BFGS and set `method='BFGS'` as the default. But both arguments can be changed. The method can be used according to available methods provided by `optim` function from R `stats` package.

Function $J_N(x_1, \dots, x_{N-1})$ defined next way (inner code fragment):

```
J <- if(is.null(lambda)) function(X){
  X <- c(x0, X, x1)
```

```

    dX <- (X[(1:N)+1] - X[1:N])/dt
    I <- sapply(1:N, function(i) I(times[i], X[i], dX[i]))
    return(sum(I)*dt)
} else function(X){
    X <- c(x0, X, x1)
    dX <- (X[(1:N)+1] - X[1:N])/dt
    I <- sapply(1:N, function(i) I(times[i], X[i], dX[i]))
    return((sum(I) + lambda*sum(dX^2))*dt)
}

```

Result

Let's get a solution for chosen variant via written code.

```

I <- function(t, x, dx) dx^2 + x^2
t0 <- 0; t1 <- 1
x0 <- 0; x1 <- 1

sol <- var.prb(I, t0, t1, x0, x1, control=list(trace=1))

```

```

## initial value 1.323400
## iter 10 value 1.310372
## iter 20 value 1.304887
## iter 30 value 1.303386
## iter 40 value 1.303140
## iter 50 value 1.303122
## final value 1.303113
## converged

```

```
summary(sol)
```

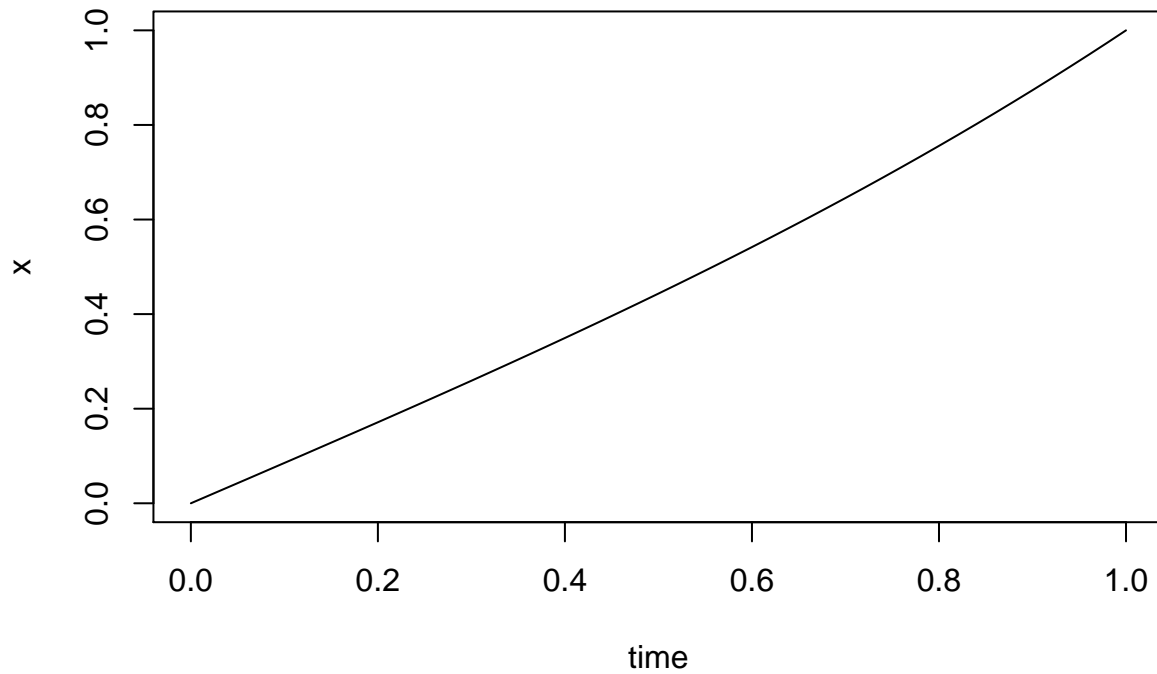
```

## Variatiion Problem solution
## Integrate I(t, x, dx) dt from 0 to 1
## Where I(t, x, dx) = dx^2 + x^2
## x = x(t): x(0) = 0, x(1) = 1
## Size of interval splitting N = 50
## J(...) depends on 49 variables (inner points)
## Regularization wasn't used
## Optimized via optim(par, fn, ...) with method='BFGS'
## Got J optimal value 1.303113
## While optimization evaluated J - 208 times and its gradient - 52 times.

```

```
plot(sol, main='Solution graphical representation')
```

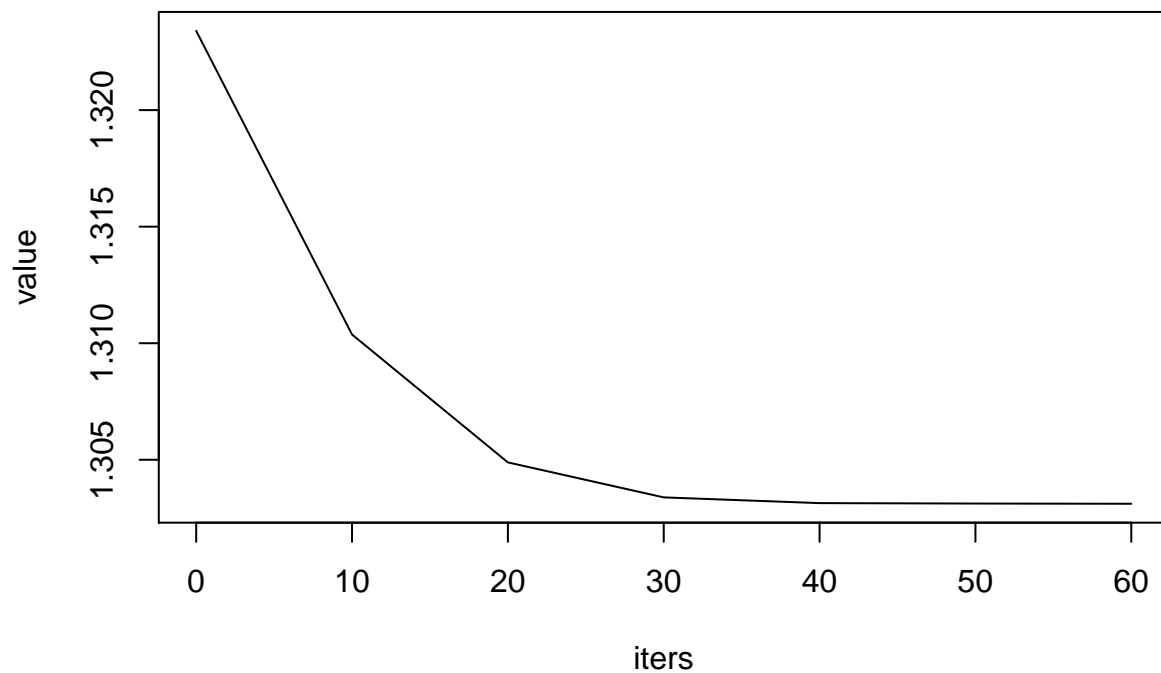
Solution graphical representation



It works easy and fine.

The plot for iterations and function J_N is below

J value dependence on iteration number



Change of J function value is decreasing while increasing of iterations number. It seems that after the 30th iteration we could stop having a pretty accurate solution. Also, I noticed, that the whole change of J is about 0.02 - it's little.

Analytical Solution

$$\frac{\partial I}{\partial x} = 2x$$

$$\frac{\partial I}{\partial \dot{x}} = 2\dot{x}$$

$$\frac{\partial^2 I}{\partial \dot{x} \partial t} = 2\ddot{x}$$

$$\left(\frac{\partial I}{\partial x} = \frac{\partial^2 I}{\partial \dot{x} \partial t}\right) \Rightarrow (2x = 2\ddot{x}) \Rightarrow (\ddot{x} - x = 0)$$

$$\lambda^2 - 1 = 0$$

$$\lambda = \pm 1$$

$$x(t) = C_1 e^t + C_2 e^{-t}, x(0) = 0, x(1) = 1$$

$$C_1 + C_2 = 0 \Rightarrow C_1 = -C_2 \quad \text{and} \quad C_1 e + \frac{C_2}{e} = 1$$

$$-C_2 e + \frac{C_2}{e} = 1 \Rightarrow C_2 = \frac{1}{e^{-1} - e}$$

$$C_1 = -C_2 = \frac{1}{e - e^{-1}}$$

$$C_1 = \frac{1}{e - e^{-1}}, \quad C_2 = \frac{1}{e^{-1} - e}$$

Finally

$$x(t) = \frac{e^t}{e - e^{-1}} + \frac{e^{-t}}{e^{-1} - e} = \frac{e^t - e^{-t}}{e - e^{-1}} = \frac{\sinh(t)}{\sinh(1)}$$

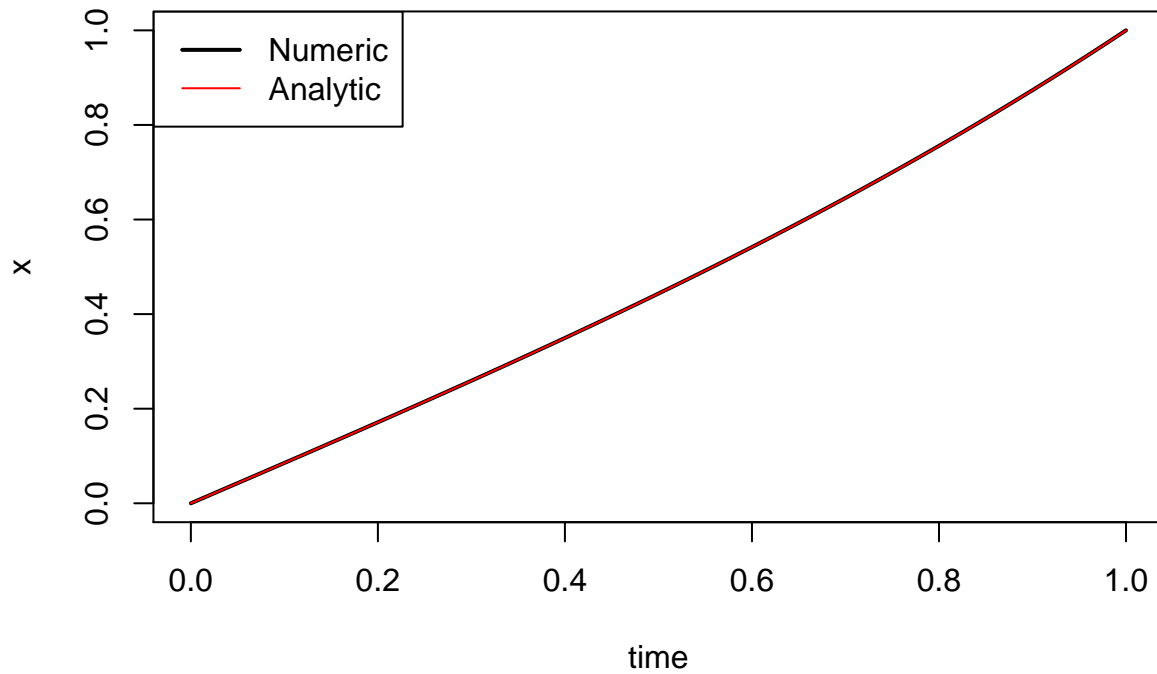
Comparisons

Set found $x(t)$ next way:

```
x <- function(t) sinh(t)/sinh(1)
time <- 0:100/100
x <- x(time)
```

Let's compare:

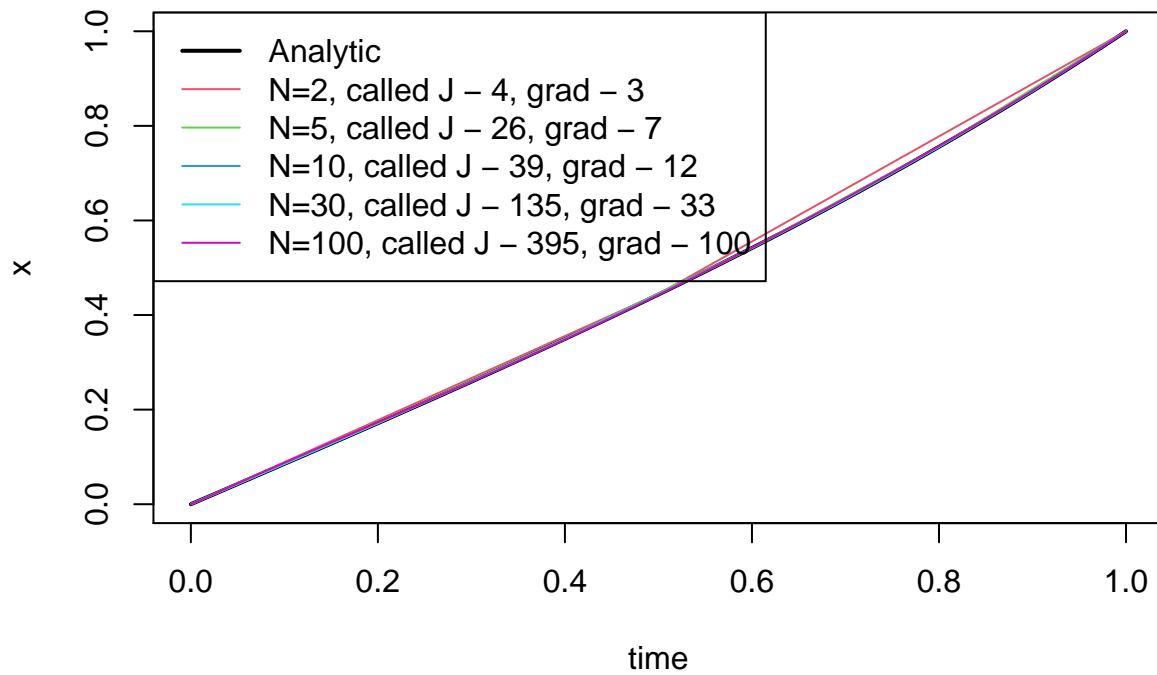
Solutions comparison



The lines are the same. So $N = 50$ is very enough to have the correct solution.

Try different N values - $N \in \{2, 5, 10, 30, 100\}$

Solutions with various N



Just a mess. Probably this example is very simple to detect which N is nicer to use. I saw no valuable difference for all $N > 2$.