Software Lab Computational Engineering Science

Convex Unconstrained Optimization

Anshika Anshika, Florian Klein, Thanh My Pham, Jan Theiß

Informatik 12: Software and Tools for Computational Engineering (STCE)
RWTH Aachen University

Contents

Preface

Subtitle

As a part of our Software Development project and in order to gain practical knowledge in the field of Mathematical Optimization ,we are required to implement an Optimization method.
Doing this project helped us strengthen our ability of solving problems together and working in group.

- Objective convex function $f = f(x(p), p)$ specification
- Minimization of f
- Computation of **gradient** and **hessian**.
- Get Number of iteration
- Should run efficiently on the RWTH Computer Cluster

- Input a Obejctive Function
- Use Linesearch to calculate stationary point x*
- Calculate Gradient (for the Gradient Descent method)
- Use Barzilai-Borwein for the calculation of step size
- Calculate value of $x^{(k+1)}$ iteratively
- checking optimality using Hesse matrix.

System Requirements(Non-Functional)

- using C++ as programming language
- development, execution on the RWTH Computer Cluster
- compilation using g++
- using make as build system
- using ADOL-C for algorithmic differentiation

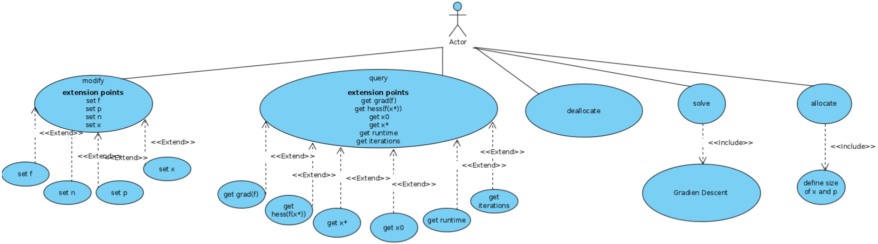


Figure: Use Case Diagramm from Jan

Activity Diagram

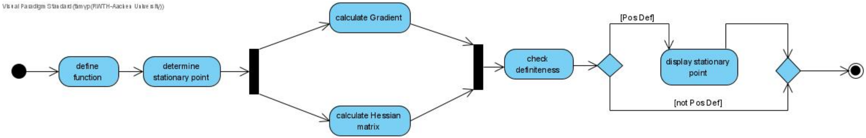


Figure: Activity Diagram

- Calculate gradient and hesse using **ADOL-C**
- Initialize vectors and matrices using **Eigen library**

Design

Class Model(s)

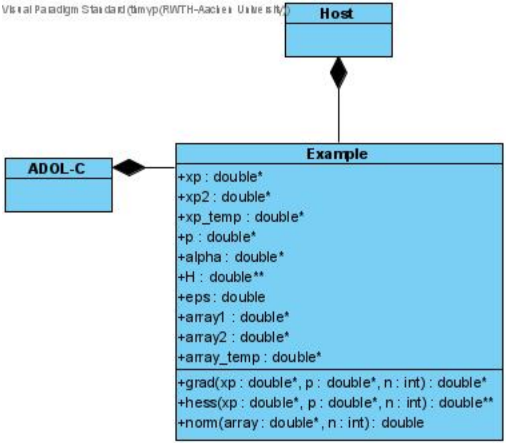


Figure: Class Diagram

Implementation

Example.hpp

```
#pragma once

class Example{
public:
//Variablen
double eps;
double* xp, xp2,xp_temp, p, alpha, array1, array2, array_temp;
double** H;

//Methoden
double* grad(double* xp, double* p, int n);

double** hess(double* xp, double* p, int n) ;

double norm(double* array, int n);

};

#include "../src/nonlinear_system_example.cpp"
```

Implementation

nonlinear_ system_ example.cpp

```
#include<adolc/adouble.h>
#include<adolc/adolc.h>
#include<iostream>

 //Gradient berechnen mit ADOL-C
double* Example::grad(double* xp, double* p, int n)
{
    double  yp = 0;
    adouble* x = new adouble[n];
    adouble  y = 0;
    adouble temp;
    trace_on(1);

    for(int i=0;i<n;i++) {
            x[i] <<= xp[i];
            temp += p[i]*x[i]*x[i]; }
    y = -exp(temp);     //<--Funktion definieren bitte hier
    y >>= yp;
    delete[] x;
    trace_off();
    double* g = new double[n];
    gradient(1,n,xp,g);
    return g;
    }
```

Implementation

nonlinear_ system_ example.cpp

```
        //Hesse berechnen mit ADOL-C
double** Example::hess(double* xp, double* p, int n)
{
    double  yp = 0;
    adouble* x = new adouble[n];
    adouble  y = 0;
    adouble temp;
    trace_on(1);

    for(int i=0;i<n;i++) {
            x[i] <<= xp[i];
            temp += p[i]*x[i]*x[i];
    }
    y = -exp(temp);      //<--Funktion definieren bitte hier
    y >>= yp;
    delete[] x;
    trace_off();
```

```
  double** H=(double**)malloc(n*sizeof(double*));
    for(int i=0;i<n;i++)
    {
        H[i]=(double*)malloc((i+1)*sizeof(double));
    }
    hessian(1,n,xp,H);

    return H;
}

double Example::norm(double* array, int n)
{
    double alpha,temp = 0;
    for(int i = 0; i<n; i++){
        temp += array[i]*array[i];
    }
    alpha = sqrt(temp);
    return alpha;
}
```

Implementation

Building (1)

```
EXE=$(addsuffix .exe, $(basename $(wildcard *.cpp)))
CPPC=g++ -g
CPPC_FLAGS=-Wall -Wextra -pedantic -Ofast -march=native
EIGEN_DIR=$(HOME)/Software/Eigen
DCO_DIR=$(HOME)/Software/dco

ADOLC_DIR=$(HOME)/adolc_base
ADOLC_INC_DIR=$(ADOLC_DIR)/include
ADOLC_LIB_DIR=$(ADOLC_DIR)/lib64

ADOLC_LIB=adolc

DCO_INC_DIR=$(DCO_DIR)/include
DCO_LIB_DIR=$(DCO_DIR)/lib
DCO_FLAGS=-DDCO_DISABLE_AUTO_WARNING
DCO_LIB=dcoc

BASE_DIR=$(HOME)/Dokumente/SP_CES/Code
LIBLS_INC_DIR=$(BASE_DIR)/LINEAR_SYSTEM/libls/include
LIBNLS_INC_DIR=$(BASE_DIR)/NONLINEAR_SYSTEM/libnls/include
LIBNLS_APPS_INC_DIR=$(BASE_DIR)/NONLINEAR_SYSTEM/libnls_apps/include
```

```
        all : $(EXE)
        ./Example.exe
        ./Example.exe
        ./Example.exe

        %.exe:  %.cpp
        $(CPPC) $(CPPC_FLAGS) $(DCO_FLAGS) -I$(EIGEN_DIR) -I$(ADOLC_INC_DIR)
        -I$(DCO_INC_DIR) -I$(LIBLS_INC_DIR) -I$(LIBNLS_INC_DIR)
        -I$(LIBNLS_APPS_INC_DIR) -Wl,--rpath -Wl,$(ADOLC_LIB_DIR)
        -L$(DCO_LIB_DIR) -L$(ADOLC_LIB_DIR) $< -o $@ -l$(DCO_LIB) -l$(ADOLC_LIB)

doc:
        cd doc && $(MAKE)

clean :
        cd doc && $(MAKE) clean
        rm -fr $(EXE)

.PHONY: all doc clean
```

```
doc
    Doxyfile
    Makefile
include
    nonlinearsystemdiffusion.hpp
    nonlinearsystemlotkavolterra.hpp
    nonlinearsystemtoy.hpp
    Example.cpp
Makefile
src
    nonlinear_ system_ diffusion.cpp
    nonlinear_ system_ lotkavolterra.cpp
    nonlinear_ system_ toy.cpp
    nonlinear_ system_ example.cpp
toy.cpp
diffusion.cpp
lotkavolterra.cpp
Example.cpp
```

Application

Example.cpp

```
#include<adolc/adouble.h>
#include<adolc/adolc.h>
#include <cassert>
#include<iostream>
#include<Eigen/Dense>
#include<random>
#include<iomanip>
#include<chrono>
#include<unistd.h>
#include<Example.hpp>
#include<string>

constexpr int MIN = -1;                //Definitionsbereich der Funktion
constexpr int MAX = 1;                 //In diesem Fall D(-1,1)
```

```
int main(){

auto start = std::chrono::steady_clock::now();
Example ex;                          //Objekt der Klasse Example erstellen
const int n = 100;                   //Anzahl unabhängiger Variablen
std::cout << "ADOL-C Tiefpunkt finden \n";

std::srand(std::time(nullptr));  //Zufallsgenerator

ex.xp = new double[n];
ex.xp2 = new double[n];
ex.xp_temp = new double[n];
ex.p = new double[n];            //Parameter
ex.alpha = new double[1];
ex.H=(double**)malloc(n*sizeof(double*));

ex.eps = 1e-6;                   //Epsilon zum anpassen für den Linesearch
ex.alpha[0] = 0.01;              //Alpha zum anpassen für den Linesearch
```

Application

Example.cpp

```
for (int i=0; i<n; i++) {
     //Zuweisung der Zufallszahlen an die x-Werte
    ex.xp[i] = MIN + (double)(rand()) / ((double)(RAND_MAX/(MAX - MIN)));
    ex.p[i] = -0.5;
}

Eigen::Vector<double,n> v;  //Erstellen eines Eigen Vektors mit den X-Werten
for (int i=0; i<n; i++) {
    v(i) = ex.xp[i];
}

std::cout << "Das ist jetzt der Vektor:" <<std::endl << v << std::endl;
```

Application

Example.cpp

```
ex.array1 = new double[n];          // Array erstellen für den Gradienten
ex.array2 = new double[n];
ex.array_temp = new double[n];

ex.array1 = ex.grad(ex.xp,ex.p,n);  //Gradienten berechnen von ersten X-Werten


Eigen::Vector<double,n> gra;        //Erstellen eines Eigen Vektors X-Werten
for (int i=0; i<n; i++) {
    gra(i) = ex.array1[i];
}

std::cout << "Das ist jetzt der Gradient:" <<std::endl << gra << std::endl;
```

Application

Example.cpp

```
int iteration = 1;

while(ex.norm(ex.array1,n)>=ex.eps){
    for(int j=0; j<n; j++){
        ex.xp2[j] = ex.xp[j]+ex.alpha[0]*(-1)*ex.array1[j];
        ex.array2 = ex.grad(ex.xp2,ex.p,n);
        ex.xp_temp[j] = ex.xp[j];
        ex.xp[j] = ex.xp2[j];
        ex.array_temp[j]=ex.array1[j];
        ex.array1[j]=ex.array2[j];
    }
    //Erstellen eines Eigen Vektors mit den X-Werten
    Eigen::Vector<double,n> grad_1;
    for (int i=0; i<n; i++) {
        grad_1(i) = ex.array_temp[i];
    }
```

```
    //Erstellen eines Eigen Vektors mit den X-Werten
 Eigen::Vector<double,n> grad_2;
    for (int i=0; i<n; i++) {
        grad_2(i) = ex.array1[i];
    }
    //Erstellen eines Eigen Vektors mit den X-Werten
    Eigen::Vector<double,n> x_1;
    for (int i=0; i<n; i++) {
        x_1(i) = ex.xp_temp[i];
    }
    //Erstellen eines Eigen Vektors mit den X-Werten
    Eigen::Vector<double,n> x_2;
    for (int i=0; i<n; i++) {
        x_2(i) = ex.xp2[i];
    }
    ex.alpha[0] = (((x_2 - x_1).transpose()) * (grad_2 - grad_1)).norm() /
                  ((grad_2 - grad_1).squaredNorm());

    iteration = iteration+1;
}
```

Application

Example.cpp

```
//Hessematrix ausrechenen zum prüfen
ex.H = ex.hess(ex.xp,ex.p,n);
//Erstellen einer Eigen Matrix die der Hesse Matrix entspricht
Eigen::Matrix<double,n,n> h;
for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++){
                h(i,j) = ex.H[i][j];
        }
}

std::cout << "Das ist jetzt die Hesse Matrix:" <<std::endl << h << std::endl;
```

Application

Example.cpp

```
//Prüfen ob positiv Definitheit durch Cholesky-Zerlegung
Eigen::LLT<Eigen::MatrixXd> lltOfA(h);
    if(lltOfA.info() == Eigen::NumericalIssue)
    {
        std::cout<<"Die Matrix ist nicht positiv definit"<<std::endl;
        return 1;
    }
    else
    {
        std::cout<<"Wie Sie sehen ist die Matrix Positiv Definit!"<<std::endl;
    }

for(int j=0; j<n; j++){
    //Ausgabe der numerisch berechneten Minimalstellen
    std::cout<<"Minimun bei x"<<j<<":"<< ex.xp[j]<<std::endl;
}
std::cout<<"Bei Iteration: "<< iteration<<std::endl;
auto end = std::chrono::steady_clock::now();
auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end-start);
std::cout << "Elapsed time in milliseconds: " << elapsed.count()*1e-6;

return 0;
}
```

Mathematical Definition

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = -exp(-0.5 \sum_{i=1}^{n} x_i^2)$$

Plots

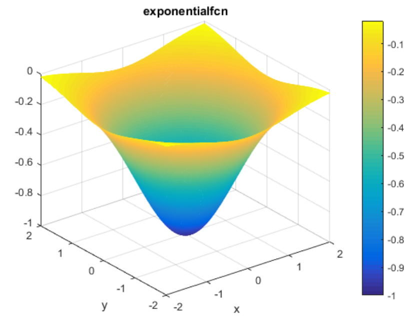
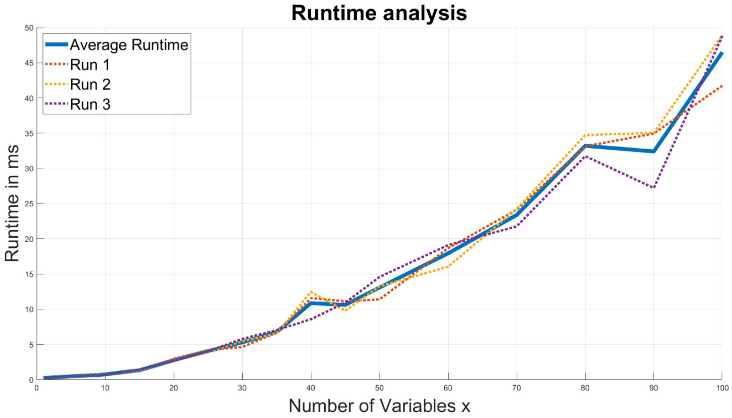
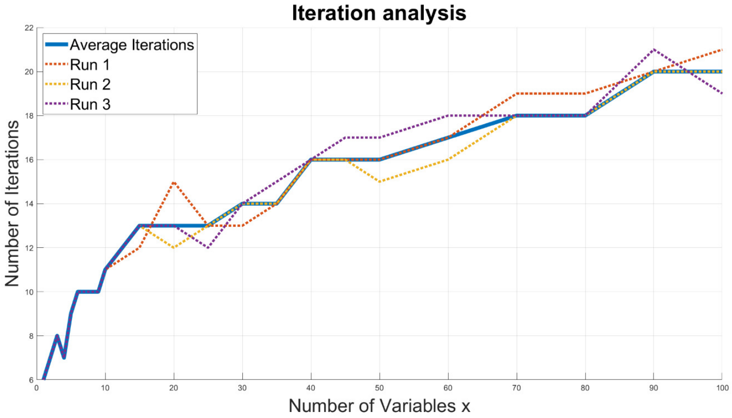


Figure: http://benchmarkfcns.xyz/benchmarkfcns/exponentialfcn.html

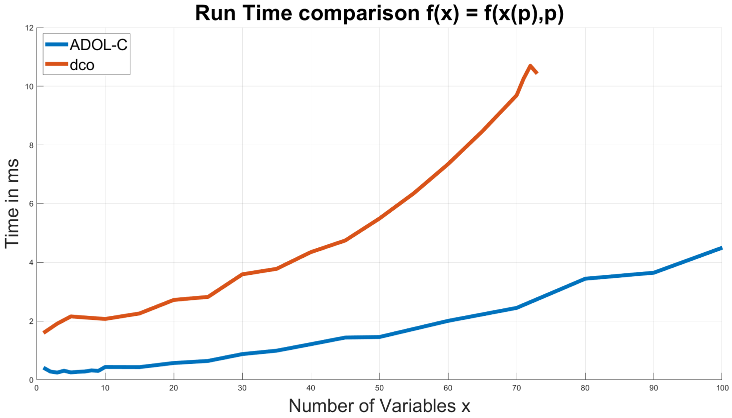


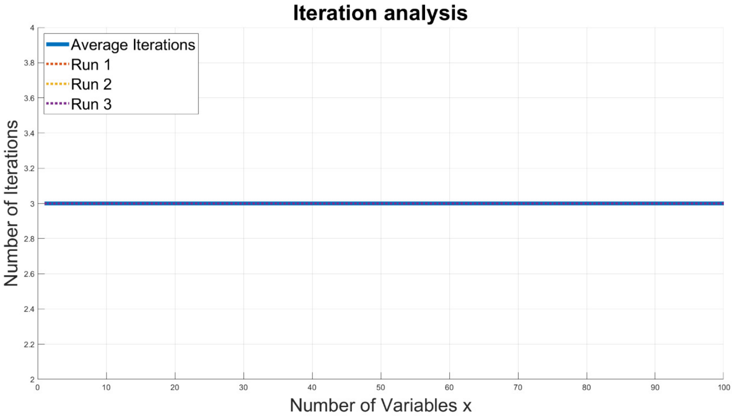
Runtime analysis
Average Runtime
Run 1
Run 2
Run 3
Runtime in ms
Number of Variables x
0
5
10
15
20
25
30
35
40
45
50
0
10
20
30
40
50
60
70
80
90
100


Iteration analysis
Average Iterations
Run 1
Run 2
Run 3
Number of Iterations
Number of Variables x
22
20
18
16
14
12
10
8
6
0
10
20
30
40
50
60
70
80
90
100

Case Study Gruppe 5

Software Tests




Iteration analysis
Average Iterations
Run 1
Run 2
Run 3
Number of Iterations
Number of Variables x
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
0
10
20
30
40
50
60
70
80
90
100

Live Software Demo

- D=(-1,1)
- $f(x) = -\exp(-0{,}5\sum_{i=0}^{n} x_i)$

Project Management

We began working on our project in the second week of May, and each of us was assigned a separate section of code. Every week, we had a meeting to ensure that everyone was on the same Page. We had several joint meetings as well because our topic was similar to that of group 5.

Summary and Conclusion

- incompatibility between **ADOL-C** and **Eigen**
- switching from typgeneric code to fixed datatypes