Wetenschappelijk Programmeren: Oefening 2

Ruben Van Assche

27 oktober 2016

1 Opgave

In deze opgave moest uit een tabel gegeven data een polynomische interpolatie en natuurlijke en kubische spline gehaald worden. Deze moest dan geplot worden om vervolgens de beste interpolatie te selecteren.

2 Omzetting Data

De originele x-as inde tabel werd voorgesteld door data(dd/mm/jjjj), dit is niet de beste manier om onze x-as voor te stellen. We verkiezen hier een x-as die start bij 0 en elke datum hier op plot door het aantal dagen tussen data te tellen. Wanneer we deze berkening maken bekomen we volgende tabel:

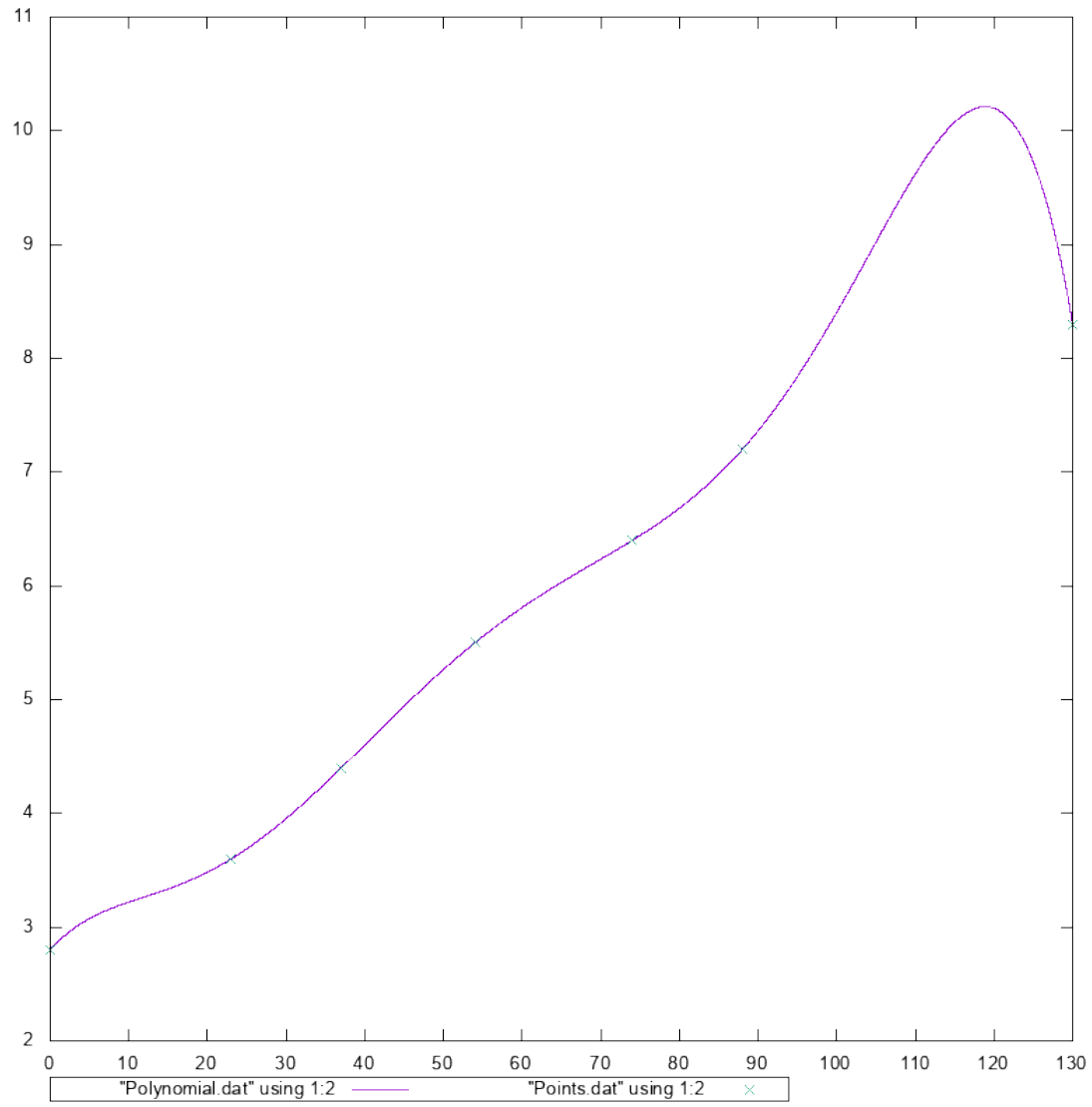
Datum	**Gewicht Baby**	**# Dagen tussen vorige datum**	**Nieuwe x-waarde**
27/10/2001	2.8	0	0
19/11/2001	3.6	23	23
03/12/2001	4.4	14	37
20/12/2001	5.5	17	54
09/01/2002	6.4	20	74
23/01/2002	7.2	14	88
06/03/2002	8.3	42	130

3 Plotten

Voor het plotten van elke interpolant wordt deze eerst berekend, vervolgens wordt er op het interval $[0, 130]$ telkens met een stap van 0.1 een evaluatie van de interpolant gedaan, de behorende y waarde en de x waarde worden vervolgens opgeslagen in een bestand. Dit bestand wordt dan geplot d.m.v. GNUPlot.

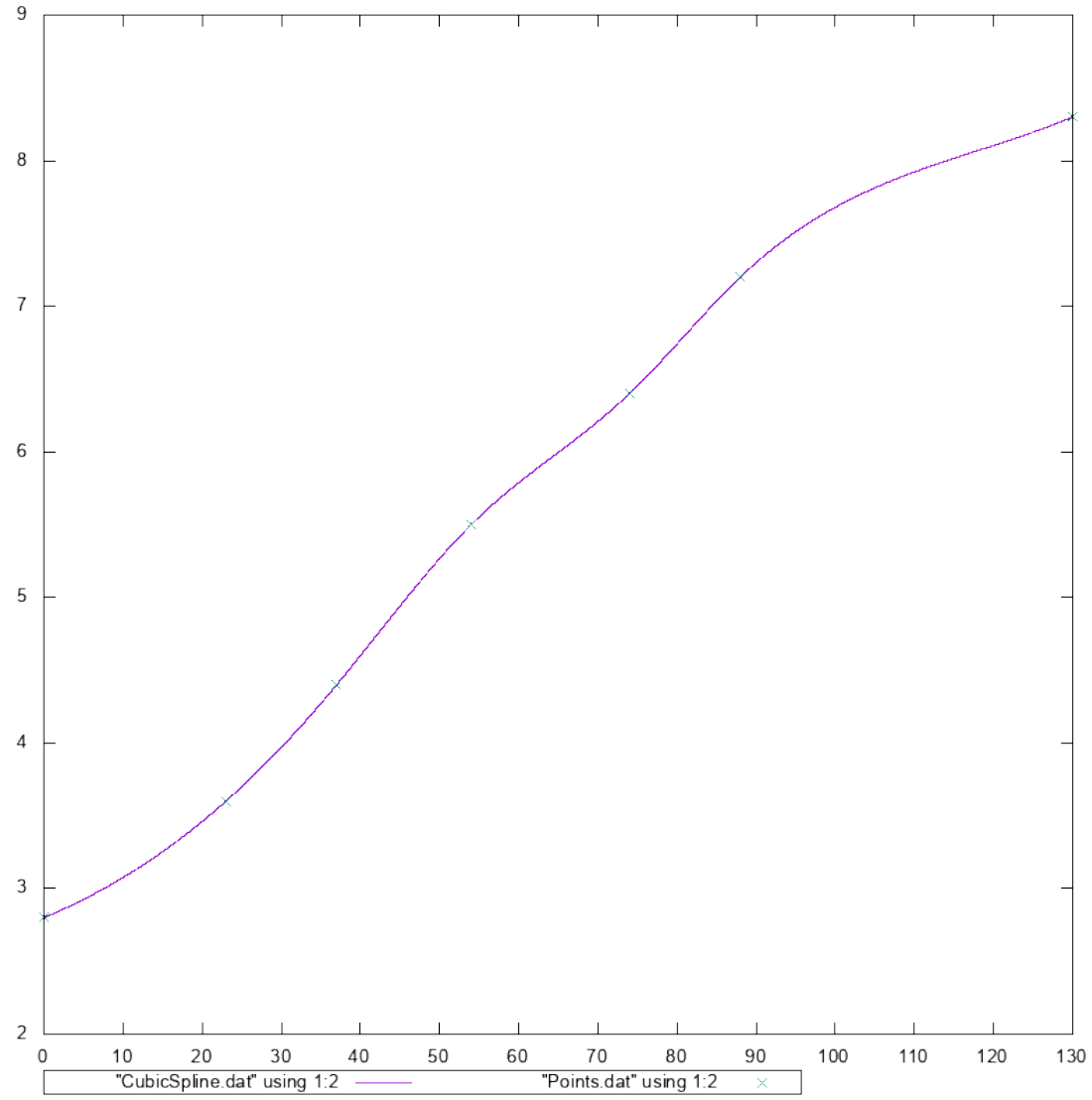
In de volgende secties overloop ik elke interpolatie methode en probeer te bespreken waarom de bepaalde methode dan wel of niet goed is voor deze set datapunten.

4 Polynomische Spline



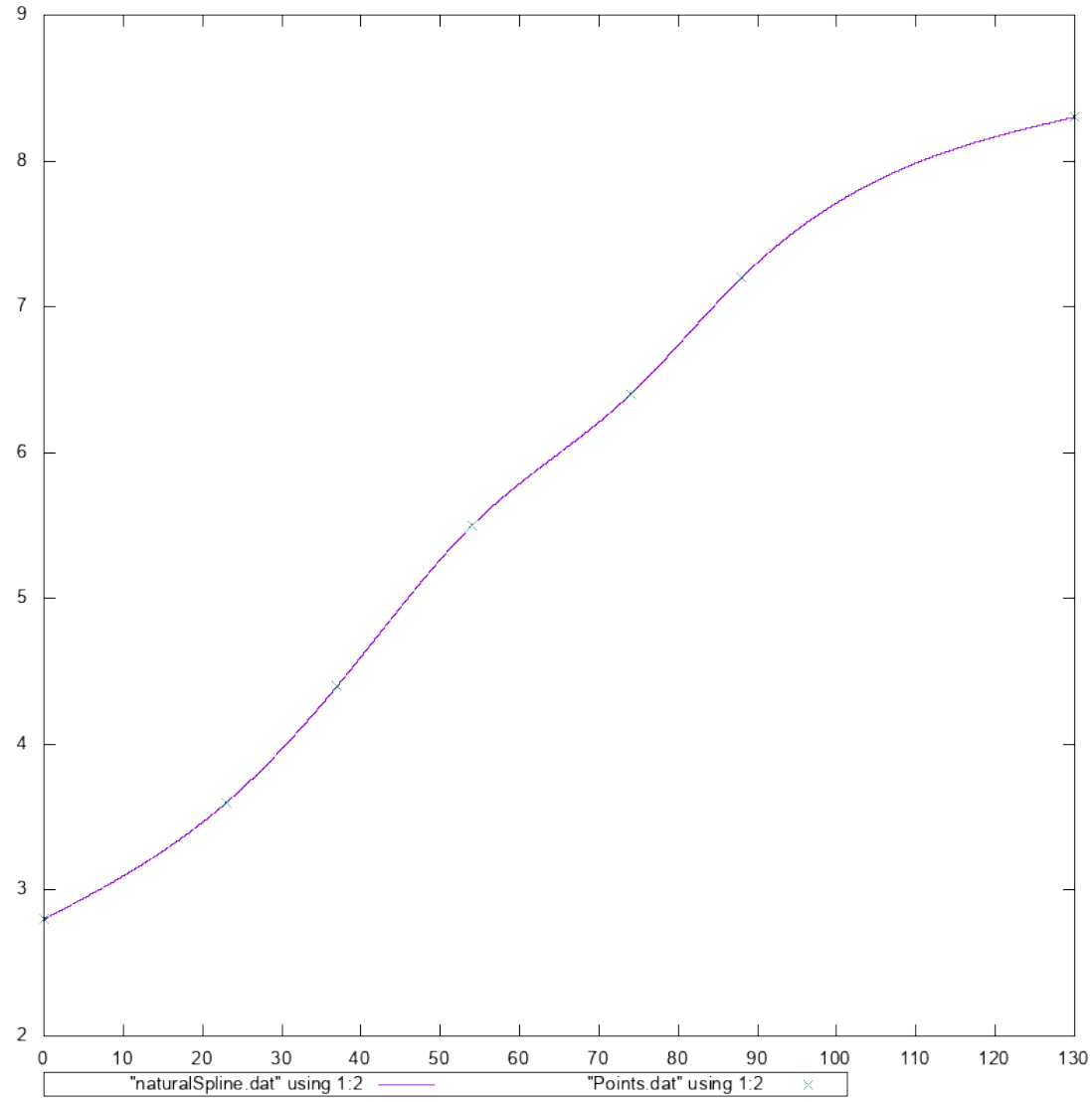
Het eerste wat bij de polynomische interpolant opvalt is dat deze niet geschikt is om de dataset te interpoleren. Het Runghe fenomeen treed hier duidelijk op, doordat er te veel datapunten voor de interpolant zijn(7) begint de polynomische interpolant te oscilleren. Dit zien we duidelijk op het einde van de functie $[90, 130]$ waar de functie een oscillatie maakt inplaats van mooi door te lopen.

5 Kubische Spline



Deze interpolatie ziet er al veel beter uit dan de Polynomische interpolatie. De kubische spline loopt veel vloeiender door en lijkt de perfectie te benaderen. Maar we zien toch nog een paar foutjes op het einde $[90, 130]$ waar de functie toch wat minder concaaf is als verwacht. Dit komt omdat GSL bij de kubische spline een gelijke y waarde voor begin en eindpunt vraagt anders treden er kleine foutjes zoals deze op.

6 Natuurlijke Spline



Om onze data te plotten maken we gebruik van de natuurlijk spline, deze heeft een mooi vloeiend verloop over de gehele functie dit omdat de tweede afgeleide van het begin en eindpunt 0 is.

7 Conclusie

Waar de polynomische functie faalt om de data te interpoleren, slaagt de kubische spline er zeker al in om een mooi resultaat te laten zien alhoewel ik toch de natuurlijke spline verkies welke net iets mooier loopt.

8 Code

Main.cpp - De code voor het berekenen van de punten door de interpolant

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_spline.h>
#include <iostream>
#include <vector>
#include <utility>
#include <fstream>
#include <string>


void writePointsFile(std::vector< std::pair<double, double> >& points, std::str
    std::ofstream file;
    file.open(filename);

    file << "#  X   Y" << std::endl;
    for(auto i: points){
        file << i.first << "   " << i.second << std::endl;
    }

    file.close();
}

void cubicSplineInterpolation(double x[], double y[], int size){
    gsl_interp_accel *acc = gsl_interp_accel_alloc();
    gsl_spline* spline = gsl_spline_alloc(gsl_interp_cspline_periodic, size);
    gsl_spline_init(spline, x, y, size);

    std::vector< std::pair<double, double> > points;

    for(double xi = x[0]; xi <= x[size-1]; xi += 0.01){
        double yi = gsl_spline_eval(spline, xi, acc);

        auto pair = std::make_pair(xi,yi);
        points.push_back(pair);
    }

    writePointsFile(points, "CubicSpline.dat");

}

void naturalSplineInterpolation(double x[], double y[], int size){
    gsl_interp_accel *acc = gsl_interp_accel_alloc();
    gsl_spline* spline = gsl_spline_alloc(gsl_interp_cspline, size);
```

```
    gsl_spline_init(spline, x, y, size);

    std::vector< std::pair<double, double> > points;

    for(double xi = x[0]; xi <= x[size-1]; xi += 0.01){
        double yi = gsl_spline_eval(spline, xi, acc);

        auto pair = std::make_pair(xi,yi);
        points.push_back(pair);
    }

    writePointsFile(points, "naturalSpline.dat");

}

void polynomialInterpolation(double x[], double y[], int size){
    gsl_interp_accel *acc = gsl_interp_accel_alloc();
    gsl_interp* interpolant = gsl_interp_alloc (gsl_interp_polynomial, size);
    gsl_interp_init(interpolant, x, y ,size);

    std::vector< std::pair<double, double> > points;

    for(double xi = x[0]; xi <= x[size-1]; xi += 0.01){
        double yi = gsl_interp_eval(interpolant, x, y, xi, acc);

        auto pair = std::make_pair(xi,yi);
        points.push_back(pair);
    }

    writePointsFile(points, "Polynomial.dat");
}

int main (void){
    // Set COUT Precision
    std::cout.precision(20);

    std::cout << "Building Points..." << std::endl;

    double x[7] = {0,23,37,54,74,88,130};
    double y[7] = {2.8, 3.6, 4.4, 5.5, 6.4, 7.2, 8.3};
    int size = 7;

    std::vector< std::pair<double, double> > points;

    for(int i = 0; i < 7; i++){
        auto pair = std::make_pair(x[i],y[i]);
        points.push_back(pair);

        std::cout << "x: " << x[i] << ", y: " << y[i] << std::endl;
    }
```

```
    writePointsFile(points, "Points.dat");


    std::cout << "Building Interpolants ..." << std::endl;
    polynomialInterpolation(x, y, size);
    naturalSplineInterpolation(x, y, size);
    cubicSplineInterpolation(x, y, size);



}
```

graphs.gnuplot - De code voor tekenen berekenen van de interpolant

```
set terminal png  transparent enhanced font "arial,10" fontscale 1.0 size 800,
set output 'Polynomial.png'
set key bmargin left horizontal Right noreverse enhanced autotitles box linetyp
set samples 10000
set xtics 10
set ytics 1
plot "Polynomial.dat" using 1:2 with lines, \
"Points.dat" using 1:2 with points

set terminal png  transparent enhanced font "arial,10" fontscale 1.0 size 800,
set output 'naturalSpline.png'
set key bmargin left horizontal Right noreverse enhanced autotitles box linetyp
set samples 10000
set xtics 10
set ytics 1
plot "naturalSpline.dat" using 1:2 with lines, \
"Points.dat" using 1:2 with points

set terminal png  transparent enhanced font "arial,10" fontscale 1.0 size 800,
set output 'CubicSpline.png'
set key bmargin left horizontal Right noreverse enhanced autotitles box linetyp
set samples 10000
set xtics 10
set ytics 1
plot "CubicSpline.dat" using 1:2 with lines, \
"Points.dat" using 1:2 with points
```