>

April 2016

Pulse Values from Long Term Measurement

Univ.-Prof. Dr.-Ing. habil. Josef *BETTEN*
RWTH Aachen University
Mathematical Models in Materials Science and Continuum Mechanics
Augustinerbach 4-20
D-52056 A a c h e n , Germany

<betten@mmw.rwth-aachen.de>

Abstract

Long term measurement of heart rate furnishes important information of cardiac anomalies. During a period of about 24 hours pulse rate of a patient at the University Hospital Aachen has been measured. Thus, we have a lot of *pulse values* important for a medical doctor treating sick patients. To analyse these "data" the Maple Program 2015 is very useful. At first the given "data" have been interpolated by *cubic spline functions.* Then, these functions have been approximated by *non linear regession* and also by *FOURIER-Series.* The measurement started at 9 am and ended on the next day at 6 am. Thus, we have 22 pulse valuses within the 21 hours. For the following analyse it's most convenient to map the x-coordinates [9...30] to the abcissa [0...2*Pi] according to: X = 2*Pi*(i - 1)/21 for i=1..22.

Pulse Values and Cubic Spline Interpolation

```
> restart:   with(Statistics):
> X:=array([seq(2*Pi*(i-1)/21,i=1..22)]);
```

$$X := \left[0, \frac{2\pi}{21}, \frac{4\pi}{21}, \frac{2\pi}{7}, \frac{8\pi}{21}, \frac{10\pi}{21}, \frac{4\pi}{7}, \frac{2\pi}{3}, \frac{16\pi}{21}, \frac{6\pi}{7}, \frac{20\pi}{21}, \frac{22\pi}{21}, \frac{8\pi}{7}, \frac{26\pi}{21}, \frac{4\pi}{3}, \frac{10\pi}{7}, \frac{32\pi}{21}, \frac{34\pi}{21}, \frac{12\pi}{7}, \frac{38\pi}{21}, \frac{40\pi}{21}, 2\pi\right]$$

```
> whattype(X);
```

symbol

```
> Y:=array([63,59,61,63,65,65,60,62,61,59,57,57,57,62,47,48,51,55,
  60,52,44,47]);
```

$$Y := [63, 59, 61, 63, 65, 65, 60, 62, 61, 59, 57, 57, 57, 62, 47, 48, 51, 55, 60, 52, 44, 47]$$

```
> whattype(Y);
```

symbol

```
> YM:=evalf(Mean(Y),4);  # mean value of Y # dimension = bpm
```

$$YM := 57.05$$

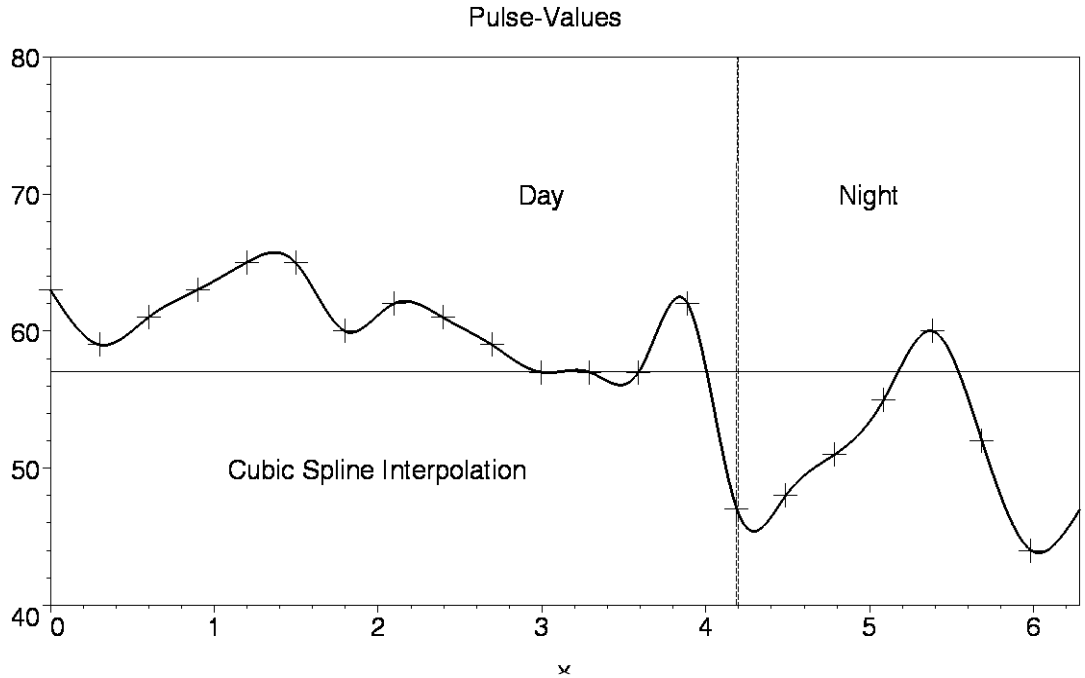
```
> DATA:=seq([X[i],Y[i]],i=1..22);
```

$$DATA := [0, 63], \left[\frac{2\pi}{21}, 59\right], \left[\frac{4\pi}{21}, 61\right], \left[\frac{2\pi}{7}, 63\right], \left[\frac{8\pi}{21}, 65\right], \left[\frac{10\pi}{21}, 65\right], \left[\frac{4\pi}{7}, 60\right],$$
$$\left[\frac{2\pi}{3}, 62\right], \left[\frac{16\pi}{21}, 61\right], \left[\frac{6\pi}{7}, 59\right], \left[\frac{20\pi}{21}, 57\right], \left[\frac{22\pi}{21}, 57\right], \left[\frac{8\pi}{7}, 57\right], \left[\frac{26\pi}{21}, 62\right], \left[\frac{4\pi}{3}, 47\right],$$
$$\left[\frac{10\pi}{7}, 48\right], \left[\frac{32\pi}{21}, 51\right], \left[\frac{34\pi}{21}, 55\right], \left[\frac{12\pi}{7}, 60\right], \left[\frac{38\pi}{21}, 52\right], \left[\frac{40\pi}{21}, 44\right], [2\pi, 47]$$

```
> whattype(DATA);
```

exprseq

```
> with(CurveFitting):
> Sp(x):=Spline([DATA],x,degree=3):
> alias(th=thickness,co=color,H=Heaviside):
> p[1]:=plot([DATA],x=0..2*Pi,40..80,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,axes=boxed,
  title="Pulse-Values"):
> p[2]:=plot(Sp(x),x=0..2*Pi,th=3,co=black):
> p[3]:=plots[textplot]({[3,70,`Day`],[5,70,`Night`],
  [2,50,`Cubic Spline Interpolation`]}):
> p[4]:=plot(YM,x=0..2*Pi,linestyle=3,th=1,co=black):
> p[5]:=plot(80*H(x-4.189)-40*H(x-4.199),x=0..2*Pi,
  linestyle=3,th=1,co=black):
> plots[display](seq(p[k],k=1..5));
```



In this Figure 22 pulse-values (+++) are interpolated by *cubic spline.* The vertcal line at x = 4*Pi/3, i.e. 23 o'clock, divides the period [0...2*Pi] into two parts (Day & Night). The horizontal line at 57.05 indicates the meanvalue YM of the pulse-values. Furthermore, the measured pulse rate is important for a medical doctor to check the heart health and the fitnes level. The normal pulse rates at rest are about (60 - 100) bpm for adults and (40 - 60) bpm for well-trained athlets. The values during Day should be higher than in the nighttime.

In the following the *cubic spline* is approximated by **Nonlinear Regression:**

```
> f:=(t,a,b,c,d,e,F)->a+b*t*sin(c*t+d)*cos(e*t+F)^2;
```

$$f := (t, a, b, c, d, e, F) \rightarrow a + b\,t\,\sin(c\,t + d)\,\cos(e\,t + F)^2$$

```
> y:=unapply(evalf(NonlinearFit(f(t,a,b,c,d,e,F),X,Y,t)),t);
```

$$y := t \rightarrow 60.6177917901184 - 3.55843771528764\,t \sin(0.628630870556338\,t - 7.84446550487827)\cos(1.93136367997716\,t - 2.36238873287308)^2$$

```
> y(x):=subs(t=x,y(t));
```

$$y(x) := 60.6177917901184 - 3.55843771528764\,x \sin(0.628630870556338\,x - 7.84446550487826)\cos(1.93136367997716\,x - 2.36238873287308)^2$$

```
> y(0):=evalf(subs(x=0,y(x)));
```

$$y(0) := 60.6177917901184$$

```
> y(0)[data]:=63;
```

$$y(0)_{data} := 63$$

```
> y(2*Pi):=evalf(subs(x=2*Pi,y(x)));
```

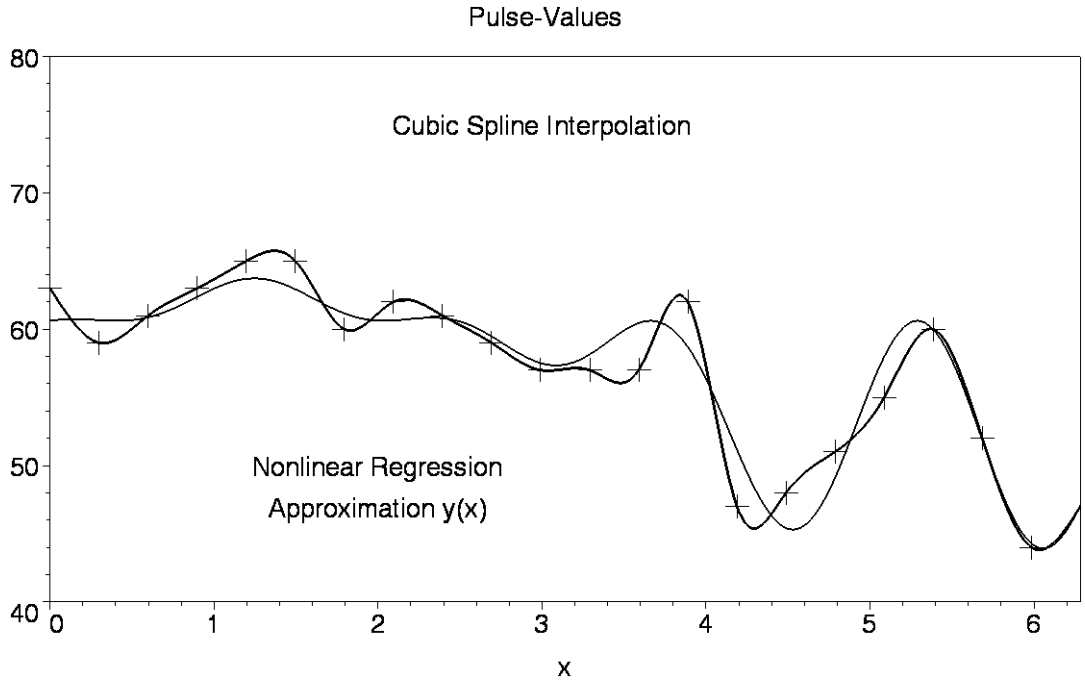
$$y(2\,\pi) := 47.1050433944327$$

```
> y(2*Pi)[data]:=47;
```

$$y(2\,\pi)_{data} := 47$$

The result of the *Nonlinear Regression* with its *Approximation* **y(x)** is represented in the next Figure:

```
> alias(th=thickness,co=color):
> p[1]:=plot([DATA],x=0..2*Pi,40..80,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,axes=boxed,
  title="Pulse-Values"):
> p[2]:=plot(Sp(x),x=0..2*Pi,th=3,co=black):
> p[3]:=plot(y(x),x=0..2*Pi,th=2,co=black):
> p[4]:=plots[textplot]({[3,75,`Cubic Spline Interpolation`],
  [2,50,`Nonlinear Regression`],[2,47,`Approximation y(x)`]}):
> plots[display](seq(p[k],k=1..4));
```



The quality of the approximation y(x) with respect to the cubic spline or to the given pulse-values can be checked by the *error norms* L[2] or l[2], respectively.

```
> L[2]:=
  sqrt((1/(2*Pi))*Int((SPLINE-APPROX)^2,x=0..2*Pi))=
  evalf(sqrt((1/(2*Pi))*int((Sp(x)-y(x))^2,x=0..2*Pi)))/YM;
```

$$L_2 := \frac{1}{2}\sqrt{2}\sqrt{\frac{1}{\pi}\int_0^{2\pi} (SPLINE - APPROX)^2\, dx} = 0.02934655276$$

```
> with(linalg):
> for i from 1 to 22 do
  v[i]:=evalf(subs(x=DATA[i][1],y(x))-DATA[i][2]) od:
> V:=vector([seq(v[i],i=1..22)]);
```

$V := [-2.38220820988157, 1.66475872791110, -0.131970019738494, -0.627688982867983, -1.30877883007925, -2.06856503490148, 1.15867243788686, -1.35971455517728, -0.187139060881871, 0.473448472020975, 0.518937263714959, 1.15202553060854, 3.45736736351700, -3.30261707123917, 4.16910917516865, -2.57974794155758, -1.89518693126377, 2.97275415919216, -0.0109951552355960, -0.117458093442199, 0.301276130214685, 0.105043394432700]$

```
> l[2]:=
  (1/sqrt(number_of_points))*Norm(V,2)=
  evalf((1/sqrt(22))*norm(V,2))/YM;
```

$$l_2 := \frac{\mathrm{Norm}(V, 2)}{\sqrt{number\_of\_points}} = 0.0332870367049969$$

The above *error norms* show that the approximation y(x) is a suitable nonlinear model function fit to given data.

Note: The above Approximation **y(x)** has been caculated using *floating-point arithmetic* ***default*** in order to find *high precise results.* Furthermore, we have tested the calculation, for instance, with *digits* = *four* and *five* and found **not acceptable** results!

>

Approximation of the Splinefunction by *FOURIER-Series*

```
> restart:
> with(Statistics):
> X:=array([seq(2*Pi*(i-1)/21,i=1..22)]);
```

$$X := \left[0, \frac{2\pi}{21}, \frac{4\pi}{21}, \frac{2\pi}{7}, \frac{8\pi}{21}, \frac{10\pi}{21}, \frac{4\pi}{7}, \frac{2\pi}{3}, \frac{16\pi}{21}, \frac{6\pi}{7}, \frac{20\pi}{21}, \frac{22\pi}{21}, \frac{8\pi}{7}, \frac{26\pi}{21}, \frac{4\pi}{3}, \frac{10\pi}{7}, \frac{32\pi}{21}, \frac{34\pi}{21}, \frac{12\pi}{7}, \frac{38\pi}{21}, \frac{40\pi}{21}, 2\pi\right]$$

```
> whattype(X);
```

symbol

```
> Y:=array([63,59,61,63,65,65,60,62,61,59,57,57,57,62,47,48,51,55,
  60,52,44,47]);
```

$$Y := [63, 59, 61, 63, 65, 65, 60, 62, 61, 59, 57, 57, 57, 62, 47, 48, 51, 55, 60, 52, 44, 47]$$

```
> whattype(Y);
```

symbol

```
> YM:=evalf(Mean(Y),4);  # mean value of Y # dimension = bpm
```

$$YM := 57.05$$

```
> DATA:=seq([X[i],Y[i]],i=1..22);
```

$$DATA := [0, 63], \left[\frac{2\pi}{21}, 59\right], \left[\frac{4\pi}{21}, 61\right], \left[\frac{2\pi}{7}, 63\right], \left[\frac{8\pi}{21}, 65\right], \left[\frac{10\pi}{21}, 65\right], \left[\frac{4\pi}{7}, 60\right], \left[\frac{2\pi}{3}, 62\right], \left[\frac{16\pi}{21}, 61\right], \left[\frac{6\pi}{7}, 59\right], \left[\frac{20\pi}{21}, 57\right], \left[\frac{22\pi}{21}, 57\right], \left[\frac{8\pi}{7}, 57\right], \left[\frac{26\pi}{21}, 62\right], \left[\frac{4\pi}{3}, 47\right], \left[\frac{10\pi}{7}, 48\right], \left[\frac{32\pi}{21}, 51\right], \left[\frac{34\pi}{21}, 55\right], \left[\frac{12\pi}{7}, 60\right], \left[\frac{38\pi}{21}, 52\right], \left[\frac{40\pi}{21}, 44\right], [2\pi, 47]$$

```
> with(CurveFitting):
> # The cubic splinefunction fit to data is given by:
> Sp(x):=Spline([DATA],x,degree=3):
> # This function is not printed because of its length.
  However,primting is possible, if we insert a semicolon (;)
  instead of the doppel point (:) at the of the command for Sp(x).
> FOURIER_series(x):=
  a[0]/2+sum(a[k]*cos(k*x)+b[k]*sin(k*x),k=1..infinity);
```

$$\text{FOURIER\_series}(x) := \frac{1}{2} a_0 + \left(\sum_{k=1}^{\infty} (a_k \cos(k\,x) + b_k \sin(k\,x))\right)$$

```
> a[k]:=(1/Pi)*Int(f(x)*cos(k*x),x=0..2*Pi);
```

$$a_k := \frac{1}{\pi}\int_0^{2\pi} \mathrm{f}(x)\cos(k\,x)\,dx$$

```
> a[0]:=simplify(subs(k=0,a[k]));
```

$$a_0 := \frac{1}{\pi}\int_0^{2\pi} \mathrm{f}(x)\,dx$$

```
> b[k]:=(1/Pi)*Int(f(x)*sin(k*x),x=0..2*Pi);
```

$$b_k := \frac{1}{\pi}\int_0^{2\pi} \mathrm{f}(x)\sin(k\,x)\,dx$$

```
> with(CurveFitting): F(x):=Sp(x):
> A[0]:=evalf(subs(f(x)=F(x),a[0]));
```

$$A_0 := 114.1960062$$

```
> A[k]:=simplify(value(subs(f(x)=F(x),a[k]))):
> A[k]:=subs({sin(k*Pi)=0.,(cos(k*Pi))^2=1.},%):
> A[k]:=evalf(%):
> for i in [seq(k,k=1..5)] do A[i]:=subs(k=i,A[k]) od:
> for i in [seq(k,k=1..15)] do P[i]:=subs(k=i,A[k]) od:
> for i in [seq(k,k=1..100)] do R[i]:=subs(k=i,A[k]) od:
> B[k]:=simplify(value(subs(f(x)=F(x),b[k]))):
> B[k]:=subs({sin(k*Pi)=0.,(cos(k*Pi))^2=1.},%):
> for i in [seq(k,k=1..5)] do B[i]:=subs(k=i,B[k]) od:
> for i in [seq(k,k=1..15)] do Q[i]:=subs(k=i,B[k]) od:
> for i in [seq(k,k=1..100)] do S[i]:=subs(k=i,B[k]) od:
> # The coefficients A[i]...S[i] are not printed because of their
  length.However, it can be done by inserting semicolons (;)
  instead of doppelpoints (:) at the ends of the commands. The
  FOURIER-Series with k = 5, 15, 100  are expressed by  y(x),
  z(x), T(x), respectively, as follows:
>
> y(x):=evalf(A[0]/2+sum(A[k]*cos(k*x)+B[k]*sin(k*x),k=1..5));
```

$$\begin{aligned}\mathrm{y}(x) := {} & 57.09800310 - 1.037475982\cos(x) + 5.750747994\sin(x) - 1.043625943\cos(2.\,x) \\ & + 1.017438123\sin(2.\,x) - 2.581563326\cos(3.\,x) - 0.3011718296\sin(3.\,x) \\ & - 2.144147642\cos(4.\,x) + 2.555986247\sin(4.\,x) + 1.715299509\cos(5.\,x) \\ & + 1.882993266\sin(5.\,x)\end{aligned}$$

```
> y(0):=simplify(subs(x=0,y(x)));
```

$$\mathrm{y}(0) := 52.00648972$$

```
> y(0)[data]:=63;
```

$$\mathrm{y}(0)_{data} := 63$$

```
> y(2*Pi):=simplify(subs(x=2*Pi,y(x)));
```

$$\mathrm{y}(2\,\pi) := 52.00648974$$

```
> y(2*Pi)[data]:=47;
```

$$\mathrm{y}(2\,\pi)_{data} := 47$$

```
>
```

```
> z(x):=evalf(A[0]/2+sum(P[k]*cos(k*x)+Q[k]*sin(k*x),k=1..15));
```

$$z(x) := -1.0374754\cos(x) + 5.750747994\sin(x) + 57.09800310 + 0.04588158316\cos(15.\,x) + 0.05707791714\cos(14.\,x) + 0.2450638226\cos(13.\,x) - 0.01220520060\cos(12.\,x) + 0.1780472153\cos(11.\,x) + 0.2354260321\cos(10.\,x) - 0.1540426313\cos(9.\,x) + 1.400542814\cos(8.\,x) + 0.1770108345\cos(7.\,x) + 0.0385764726\cos(6.\,x) + 0.3026995316\sin(14.\,x) + 0.3339772552\sin(15.\,x) + 0.5473660021\sin(12.\,x) + 0.5061307261\sin(13.\,x) + 1.104154831\sin(10.\,x) + 0.05669948876\sin(11.\,x) - 0.1608339671\sin(8.\,x) + 0.1772926149\sin(9.\,x) + 1.065751796\sin(6.\,x) + 1.704896570\sin(7.\,x) + 1.715299510\cos(5.\,x) + 1.882993268\sin(5.\,x) - 2.144147645\cos(4.\,x) + 2.555986247\sin(4.\,x) - 1.04362594\cos(2.\,x) + 1.017438123\sin(2.\,x) - 2.581563325\cos(3.\,x) - 0.3011718296\sin(3.\,x)$$

```
> z(0):=simplify(subs(x=0,z(x)));
```

$$z(0) := 54.21786916$$

```
> z(0)[data]:=63;
```

$$z(0)_{data} := 63$$

```
> z(2*Pi):=simplify(subs(x=2*Pi,z(x)));
```

$$z(2\,\pi) := 54.21786923$$

```
> z(2*Pi)[data]:=47;
```

$$z(2\,\pi)_{data} := 47$$

```
>
> T(x):=evalf(A[0]/2+sum(R[k]*cos(k*x)+S[k]*sin(k*x),k=1..100)):
> # This equation is not printed because of its length with more
  than 100 terms. However, printing is possible, if we insert a
  semicolon (;) instead of the doppelpoint (:) at the end of the
  command for T(x).
> T(0):=simplify(subs(x=0,T(x)));
```

$$T(0) := 54.88027315$$

```
> T(0)[data]:=63;
```

$$T(0)_{data} := 63$$

```
> T(2*Pi):=simplify(subs(x=2*Pi,T(x)));
```

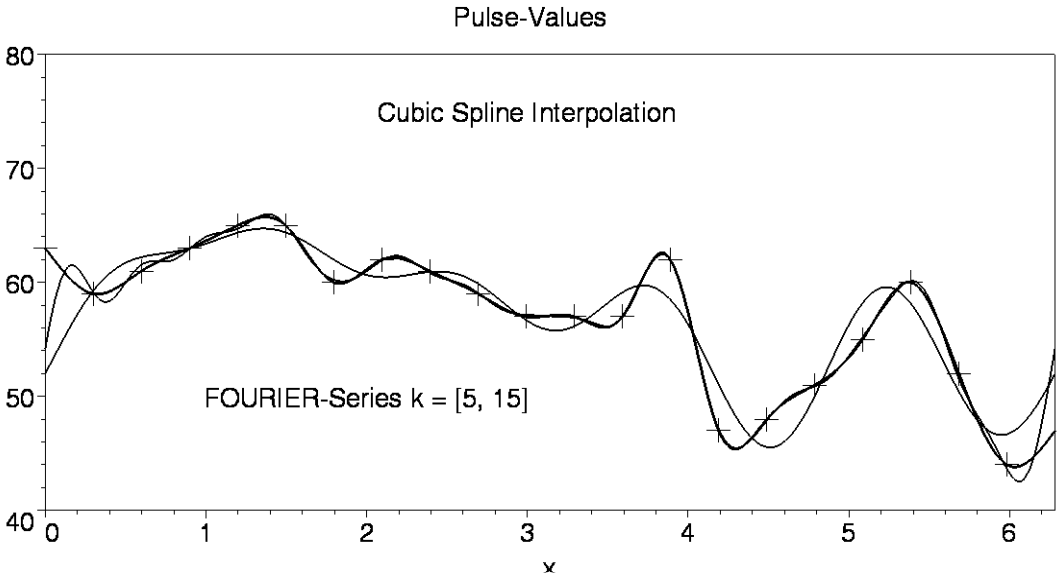
$$T(2\,\pi) := 54.88027358$$

```
> T(2*Pi)[data]:=47;
```

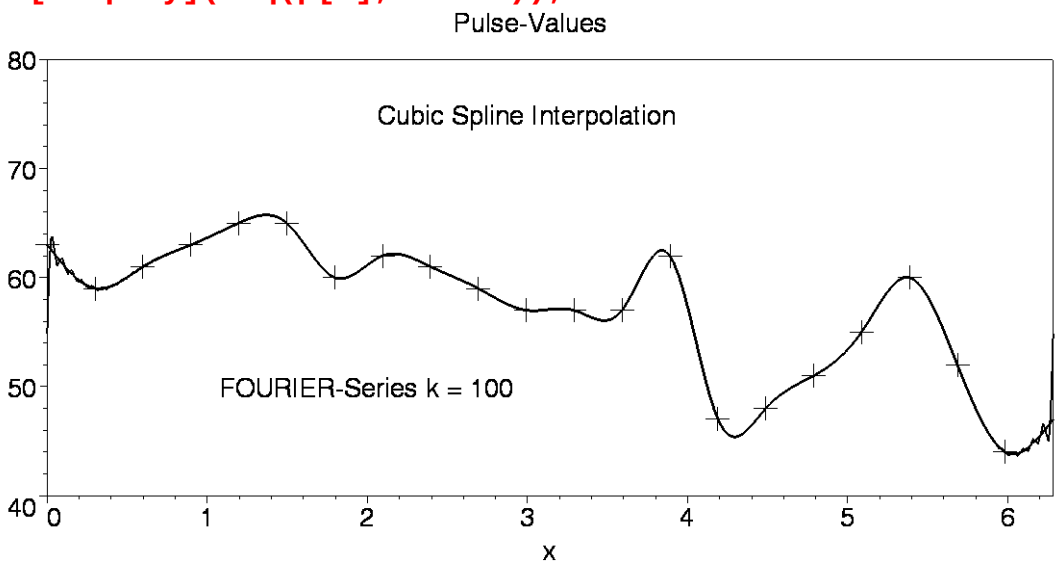
$$T(2\,\pi)_{data} := 47$$

```
> # The following Figure illustrades the approximations of the
  splinefunction by FOURIER-Series with k = [5, 15]:
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,40..80,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,title="Pulse-Values"):
> p[3]:=plot({y(x),z(x)},x=0..2*Pi,th=2,co=black):
> p[4]:=plots[textplot]({[3,75,`Cubic Spline Interpolation`],
  [2,50,`FOURIER-Series k = [5, 15]`]}):
```

```
> plots[display](seq(p[k],k=1..4));
```




```
> # The next Figure shows the FOURIER-Series with k = 100:
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,40..80,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,title="Pulse-Values"):
> p[3]:=plot(T(x),x=0..2*Pi,th=2,co=black):
> p[4]:=plots[textplot]({[3,75,`Cubic Spline Interpolation`],
  [2,50,`FOURIER-Series k = 100`]}):
> plots[display](seq(p[k],k=1..4));
```



In this Figure we cannot observe differencies between the splinefunction and the FOURIER-Series. Only immediately at the beginning x = 0 and at the end x = 2*Pi:

```
> T(0):=simplify(subs(x=0,T(x)));
  T(0.021):=simplify(subs(x=0.021,T(x)));
```

$$T(0) := 54.88027315$$

$$T(0.021) := 63.06636791$$

```
> T(0)[data]:=63;
```

$$T(0)_{data} := 63$$

```
> T(2*Pi):=simplify(subs(x=2*Pi,T(x)));
  T(1.9943*Pi):=simplify(subs(x=1.9943*Pi,T(x)));
```

$$T(2\,\pi) := 54.88027358$$

$$T(6.265278230) := 47.03280626$$

```
> T(2*Pi)[data]:=47;
```

$$T(2\,\pi)_{data} := 47$$

Because of the above anomaly at x = 0 and x = 2*Pi, let us discuss the following *error norms* L[2] and l[2] between x = 2*Pi/21 = 0.299 and x = 40*Pi/21 = 5.984 instead x = [0...2*Pi]. Thus, we consider only 20 pulse-values instead of the complete number of 22.
The *error norms* L[2](k = 5, 15, 100), of the approximations y(x), z(x), and T(x) with respect to the cubicfunction can thus be expressed as:

```
> L[2][k=5]:=
  sqrt((1/5.685)*Int((SP(xi)-y(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-y(x))^2,x=0.299..5.984))/YM;
```

$$L_{2_{k=5}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (SP(\xi) - y(\xi))^2\, d\xi} = 0.02973261534$$

```
> L[2][k=15]:=
  sqrt((1/5.685)*Int((SP(xi)-z(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-z(x))^2,x=0.299..5.984))/YM;
```

$$L_{2_{k=15}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (SP(\xi) - z(\xi))^2\, d\xi} = 0.004180071437$$

```
> L[2][k=100]:=
  sqrt((1/5.685)*Int((SP(xi)-T(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-T(x))^2,x=0.299..5.984))/YM;
```

$$L_{2_{k=100}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (SP(\xi) - T(\xi))^2\, d\xi} = 0.0006802844784$$

```
>
```

The three values above show the *error norms* L[2] of the *FOURIER-Series* y(x), z(x), and T(x) relative to the *cubic splinefunction.* Furthermore one should discuss the *error norms* with respect to the given measured data, as follows:

```
> with(linalg):
> for i from 2 to 21 do
  u[i]:=subs(x=DATA[i][1],y(x))-DATA[i][2] od:
> U:=evalf(vector([seq(u[i],i=2..21)]),4);
```

$U := [0.268, 1.223, -0.025, -0.636, -0.615, 1.995, -1.547, -0.064, 0.895, -0.3071, -0.8599,$
$2.110, -3.518, 4.081, -2.449, -0.965, 3.229, -1.855, -1.589, 2.697]$

```
> l[2][k=5]:=
  (1/sqrt(number_of_points))*Norm(U,2)=
  evalf((1/sqrt(20))*norm(U,2),4)/YM;
```

$$l_{2_{k=5}} := \frac{\mathrm{Norm}(U, 2)}{\sqrt{number\_of\_points}} = 0.03381244522$$

```
> for i from 2 to 21 do
  v[i]:=subs(x=DATA[i][1],z(x))-DATA[i][2] od:
> V:=evalf(vector([seq(v[i],i=2..21)]),4);
```

$V := [0.138, 0.557, -0.047, -0.291, 0.045, 0.236, -0.063, -0.158, 0.073, 0.1649, -0.1109, -0.123, 0.047, 0.154, -0.170, -0.105, 0.263, 0.113, -0.479, -0.384]$

```
> l[2][k=15]:=
  (1/sqrt(number_of_points))*Norm(V,2)=
  evalf((1/sqrt(20))*norm(V,2),4)/YM;
>
```

$$l_{2_{k=15}} := \frac{\mathrm{Norm}(V, 2)}{\sqrt{number\_of\_points}} = 0.004087642419$$

```
> for i from 2 to 21 do
  w[i]:=subs(x=DATA[i][1],T(x))-DATA[i][2] od:
> W:=evalf(vector([seq(w[i],i=2..21)]),4);
```

$W := [-0.034, 0.079, 0.041, -0.019, -0.043, 0.006, 0.032, 0.002, -0.039, -0.0201, -0.0119, -0.007, -0.040, -0.021, -0.020, 0.041, 0.022, 0.001, -0.056, 0.010]$

```
> l[2][k=100]:=
  (1/sqrt(number_of_points))*Norm(W,2)=
  evalf((1/sqrt(20))*norm(W,2),4)/YM;
```

$$l_{2_{k=100}} := \frac{\mathrm{Norm}(W, 2)}{\sqrt{number\_of\_points}} = 0.0005851007888$$

```
>
```

Note: Because of its "piecevise character" of the splinefunction the precision of the L[2] error norm cannot be arrived by *Digits*, for instance, equal to four. Therefore, we have caculeted the L[2] error norms above with the default value of ten. However, the calculation of the l[2] error norms with *Digits = 4* is possible without loss of *precision.*

The *error norms* may sometimes be improved without changing the number of terms, if we utilize the method of *smoothing.* As an example let us apply this method to the FOURIER-Series y(x) with k = 5 terms. Introducing the *smoothing factor* g(k,N) we arrive at the *modified* Fourier-Series G(x,n,N) as:

```
> g(k,N):=N*sin(Pi*k/N)/Pi/k;    # smoothing factor for
  FOURIER-Series
```

$$g(k, N) := \frac{N \sin\left(\frac{\pi k}{N}\right)}{\pi k}$$

```
> G(x,n,N):=
  A[zero]/2+sum(g(kappa,Nu)*(A[kappa]*cos(kappa*x)+
```

```
  B[kappa]*sin(kappa*x)),kappa =1..n);  # smoothing function
```

$$G(x, n, N) := \frac{1}{2} A_{zero} + \left(\sum_{\kappa=1}^{n} g(\kappa, N) \left(A_\kappa \cos(\kappa x) + B_\kappa \sin(\kappa x) \right) \right)$$

```
> # Example for y(x) with n = 5 and N = n + 1:
> g(k,6):=subs(N=6,g(k,N));
```

$$g(k, 6) := \frac{6 \sin\left(\frac{\pi k}{6}\right)}{\pi k}$$

```
> G(x,n=5,N=6):=
  evalf(subs({n=5,Nu=6,kappa=k,g(kappa,Nu)=g(k,6),
  A[zero]=A[0],A[kappa]=A[k],b[kappa]=B[k]},G(x,n,N)));
```

$$G(x, n = 5, N = 6) := 57.09800310 - 0.9907161038 \cos(x) + 5.491559915 \sin(x) - 0.8630716053 \cos(2.\ x) + 0.8414145698 \sin(2.\ x) - 1.643474286 \cos(3.\ x) - 0.1917319392 \sin(3.\ x) - 0.8865979059 \cos(4.\ x) + 1.056891807 \sin(4.\ x) + 0.3275980719 \cos(5.\ x) + 0.3596252237 \sin(5.\ x)$$

```
> G(0,n=5,N=6):=simplify(subs(x=0,G(x,n=5,N=6)));
```

$$G(0, n = 5, N = 6) := 53.04174128$$

```
> y(0):=simplify(subs(x=0,y(x)));
```

$$y(0) := 52.00648972$$

```
> y(0)[data]:=63;
```

$$y(0)_{data} := 63$$

```
> G(2*Pi,n=5,N=6):=simplify(subs(x=2*Pi,G(x,n=5,N=6)));
```

$$G(2\,\pi, n = 5, N = 6) := 53.04174126$$

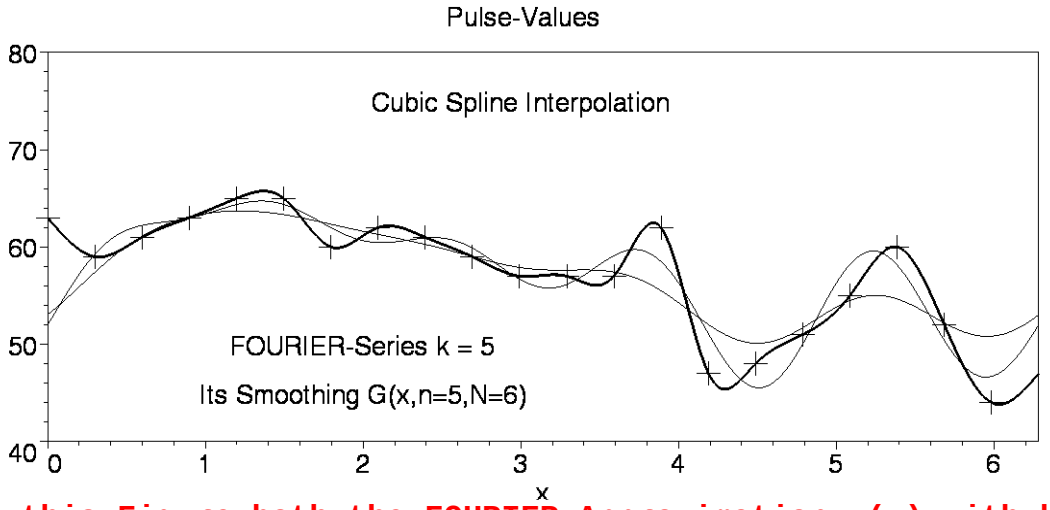
```
> y(2*Pi):=simplify(subs(x=2*Pi,y(x)));
```

$$y(2\,\pi) := 52.00648974$$

```
> y(2*Pi)[data]:=47;
```

$$y(2\,\pi)_{data} := 47$$

```
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,40..80,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,
  style=point,symbol=cross,symbolsize=50):
> p[3]:=plot({y(x),G(x,n=5,N=6)},x=0..2*Pi,
  th=1,co=black,title="Pulse-Values"):
> p[4]:=plots[textplot]({[3,75,`Cubic Spline Interpolation`],
  [2,50,`FOURIER-Series k = 5`],[2,45,`Its Smoothing
  G(x,n=5,N=6)`]}):
> plots[display](seq(p[k],k=1..4));
```




```
> # In this Figure both the FOURIER-Approximation y(x) with k = 5
  and its smoothing G(x,n=5,N=6) are represented by thin lines and
  differ from the cubic spline interpolation (thick line) more at
  the beginning x = 0 and at the end
> #  x = 2*Pi. Thus, we consider the data between x = 2*Pi/21 =
  0.299 and         x = 40*Pi/21 = 5.984 instead of  x =
  [0...2*Pi] in order to calculate the error norms M[2] and m[2]
  for the smoothing with respect to the cubic spline and the given
  data, respectively.
> M[2][G_k=5]:=
  sqrt((1/5.685)*Int((SP(xi)-G(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-G(x,n=5,N=6))^2,x=0.299..5.984))/YM;
```

$$M_{2_{G\_k=5}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (\mathrm{SP}(\xi) - \mathrm{G}(\xi))^2 \, d\xi} = 0.04361025500$$

```
>
> L[2][y_k=5]:=0.02973261539;  # by comparison with M[2][G_k=5]
```

$$L_{2_{y\_k=5}} := 0.02973261539$$

```
> with(linalg):
> for i from 2 to 21 do
  q[i]:=subs(x=DATA[i][1],G(x,n=5,N=6))-DATA[i][2] od:
> Q:=evalf(vector([seq(q[i],i=2..21)]));
```

$$Q := [-1.600718285, 0.118110312, 0.100398833, -1.296254782, -1.672640897, 2.344476781, -0.708082836, -0.731428092, 0.008441527, 0.8974699744, 0.6244444103, 0.431036868, -6.503707734, 5.029927996, 2.102381119, 0.703805299, -0.511270106, -5.342454724, 0.193510065, 6.828878082]$$

```
> m[2][G_k=5]:=
  (1/sqrt(number_of_points))*Norm(Q,2)=
  evalf((1/sqrt(20))*norm(Q,2))/YM;
```

$$m_{2_{G\_k=5}} := \frac{\mathrm{Norm}(Q, 2)}{\sqrt{number\_of\_points}} = 0.05003553206$$

```
>
```

```
> l[2][y_k=5]:=0.03381244522;  # by coparison with m[2][G_k=5]
```

$$l_{2_{y\_k=5}} := 0.03381244522$$

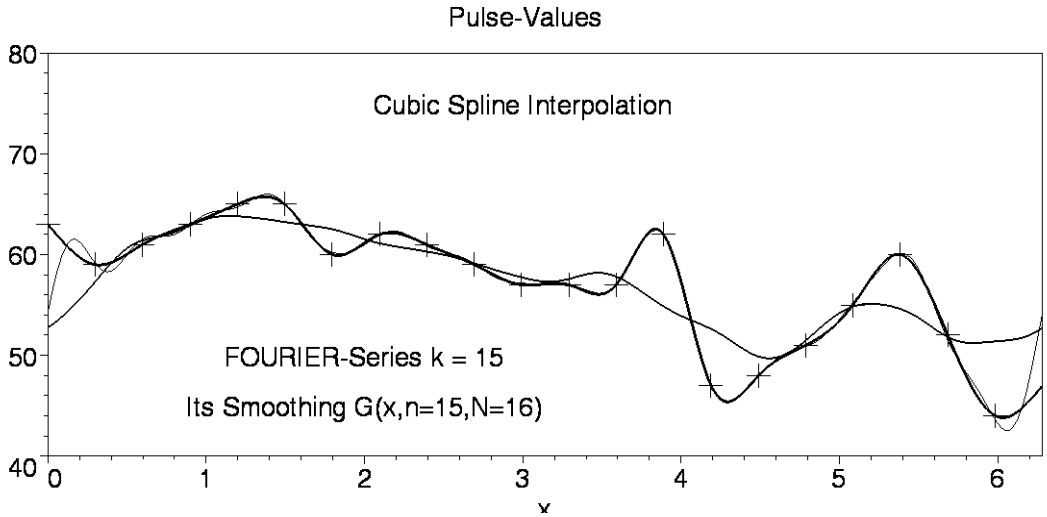
```
> # We see, in this example of the approximation with k = 5 terms
  the error norms L[2], l[2] are less than the corresponding
  values M[2], m[2]. In other examples we observe less error norms
  of the smoothings than the corresponding values due to the
  original approximations.
>
> # The next example is concerned with the smoothing of the
  FOURIER-Series z(x) with k = 15 and N = 16:
> g(k,16):=subs(N=16,g(k,N));    # smoothing factor
```

$$g(k, 16) := \frac{16 \sin\left(\frac{\pi k}{16}\right)}{\pi k}$$

```
> G(x,n=15,N=16):=
  evalf(subs({n=15,Nu=16,kappa=k,g(kappa,Nu)=g(k,6),
  A[zero]=A[0],A[kappa]=A[k],B[kappa]=B[k]},G(x,n,N)));
```

$$G(x, n = 15, N = 16) := -0.9907161288 \cos(x) + 5.491559855 \sin(x) + 57.09800310$$
$$+ 0.005841824634 \cos(15.\, x) + 0.006743293932 \cos(14.\, x) + 0.01800143943 \cos(13.\, x)$$
$$- 0.01545659691 \cos(11.\, x) - 0.03893915225 \cos(10.\, x) + 0.03268886121 \cos(9.\, x)$$
$$- 0.2895598960 \cos(8.\, x) - 0.02414755710 \cos(7.\, x) + 0.03576149974 \sin(14.\, x)$$
$$+ 0.04252330470 \sin(15.\, x) + 0.03717840312 \sin(13.\, x) - 0.1826257387 \sin(10.\, x)$$
$$- 0.004922183990 \sin(11.\, x) + 0.03325215555 \sin(8.\, x) - 0.03762266114 \sin(9.\, x)$$
$$- 0.2325794711 \sin(7.\, x) + 0.3275980719 \cos(5.\, x) + 0.3596252233 \sin(5.\, x)$$
$$- 0.8865979059 \cos(4.\, x) + 1.056891806 \sin(4.\, x) - 0.8630716019 \cos(2.\, x)$$
$$+ 0.8414145716 \sin(2.\, x) - 1.643474285 \cos(3.\, x) - 0.1917319432 \sin(3.\, x)$$

```
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,40..80,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,
  style=point,symbol=cross,symbolsize=50):
> p[3]:=plot(z(x),x=0..2*Pi,th=1,co=black,title="Pulse-Values"):
> p[4]:=plot(G(x,n=15,N=16),x=0..2*Pi,th=2,co=black):
> p[5]:=plots[textplot]({[3,75,`Cubic Spline Interpolation`],
  [2,50,`FOURIER-Series k = 15`],[2,45,`Its Smoothing
  G(x,n=15,N=16)`]}):
> plots[display](seq(p[k],k=1..5));
```




```
>
> # Error Norm of the smoothing with respect to the
  splinefunction:
> M[2][G_k=15]:=
  sqrt((1/5.685)*Int((SP(xi)-G(xi))^2,x=0.299..5.984))=
  evalf(sqrt(((1/5.685))*int((Sp(x)-G(x,n=15,N=16))^2,x=0.299..5.9
  84)))/YM;
```

$$M_{2_{G\_k=15}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (\mathrm{SP}(\xi) - \mathrm{G}(\xi))^2 \, dx} = 0.04649703844$$

```
> # Error Norm of the smoothing with respect to the given data:
> with(linalg):
> for i from 2 to 21 do
  r[i]:=evalf(subs(x=DATA[i][1],G(x,n=15,N=16))-DATA[i][2]) od:
> R:=vector([seq(r[i],i=2..21)]);
```

R := [-1.689240123, 0.4062067270, -0.1245478528, -1.195870476, -1.767450212, 2.515720582, -0.8807586212, -0.6939730671, 0.1358380948, 0.7443021160, 0.5812004139, 0.8126073702, -7.143995359, 5.623023666, 1.887850136, 0.4772601670, -0.154350460, -5.400189975, -0.1914917503, 7.379010261]

```
> m[2][G_k=15]:=
  (1/sqrt(number_of_points))*Norm(R,2)=
  evalf((1/sqrt(20))*norm(R,2))/YM;
```

$$m_{2_{G\_k=15}} := \frac{\mathrm{Norm}(R, 2)}{\sqrt{number\_of\_points}} = 0.05357869152$$

The next part of this worsheet is concerned with the

Dimensionless Representation

where the given pulse values have been divided by its mean value YM = 57.05.

```
> restart:
> with(Statistics):
```

```
> X:=array([seq(2*Pi*(i-1)/21,i=1..22)]);
```

$$X := \left[0, \frac{2\pi}{21}, \frac{4\pi}{21}, \frac{2\pi}{7}, \frac{8\pi}{21}, \frac{10\pi}{21}, \frac{4\pi}{7}, \frac{2\pi}{3}, \frac{16\pi}{21}, \frac{6\pi}{7}, \frac{20\pi}{21}, \frac{22\pi}{21}, \frac{8\pi}{7}, \frac{26\pi}{21}, \frac{4\pi}{3}, \frac{10\pi}{7}, \frac{32\pi}{21}, \frac{34\pi}{21}, \frac{12\pi}{7}, \frac{38\pi}{21}, \frac{40\pi}{21}, 2\pi\right]$$

```
> whattype(X);
```

symbol

```
> Y:=array([63,59,61,63,65,65,60,62,61,59,57,57,57,62,47,48,51,55,
  60,52,44,47]);
```

$$Y := [63, 59, 61, 63, 65, 65, 60, 62, 61, 59, 57, 57, 57, 62, 47, 48, 51, 55, 60, 52, 44, 47]$$

```
> whattype(Y);
```

symbol

```
> YM:=evalf(Mean(Y));   # mean value of Y  # dimension = bpm
```

$$YM := 57.0454545454545$$

```
> Z:=array(evalf([seq(Y[i],i=1..22)/YM],4));   #  dimensionless
  pulse values
```

$$Z := [1.104, 1.034, 1.069, 1.104, 1.139, 1.139, 1.052, 1.087, 1.069, 1.034, 0.9992, 0.9992, 0.9992, 1.087, 0.8239, 0.8414, 0.8940, 0.9642, 1.052, 0.9116, 0.7713, 0.8239]$$

```
> whattype(Z);
```

symbol

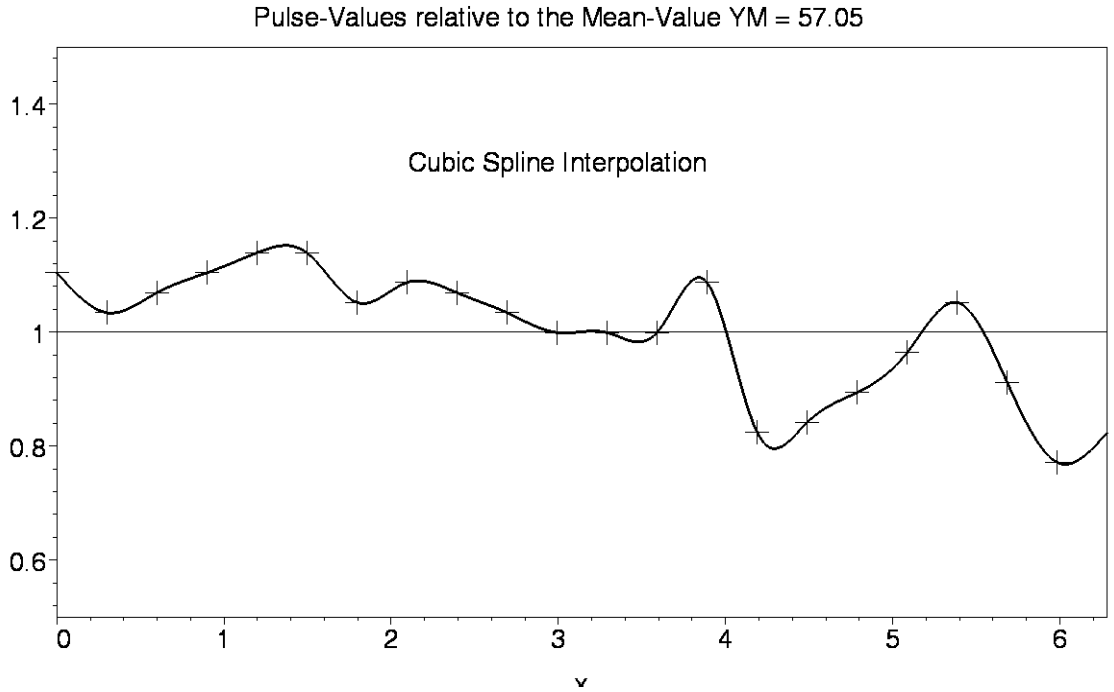
```
>
> ZM:=convert(Mean(Z),rational);
```

$$ZM := \frac{52376}{52381}$$

```
> DATA:=seq([X[i],Z[i]],i=1..22);
```

$$DATA := [0, 1.104], \left[\frac{2\pi}{21}, 1.034\right], \left[\frac{4\pi}{21}, 1.069\right], \left[\frac{2\pi}{7}, 1.104\right], \left[\frac{8\pi}{21}, 1.139\right], \left[\frac{10\pi}{21}, 1.139\right], \left[\frac{4\pi}{7}, 1.052\right], \left[\frac{2\pi}{3}, 1.087\right], \left[\frac{16\pi}{21}, 1.069\right], \left[\frac{6\pi}{7}, 1.034\right], \left[\frac{20\pi}{21}, 0.9992\right], \left[\frac{22\pi}{21}, 0.9992\right], \left[\frac{8\pi}{7}, 0.9992\right], \left[\frac{26\pi}{21}, 1.087\right], \left[\frac{4\pi}{3}, 0.8239\right], \left[\frac{10\pi}{7}, 0.8414\right], \left[\frac{32\pi}{21}, 0.8940\right], \left[\frac{34\pi}{21}, 0.9642\right], \left[\frac{12\pi}{7}, 1.052\right], \left[\frac{38\pi}{21}, 0.9116\right], \left[\frac{40\pi}{21}, 0.7713\right], [2\pi, 0.8239]$$

```
> with(CurveFitting):
> Sp(x):=Spline([DATA],x,degree=3):
> alias(th=thickness,co=color):
> p[1]:=plot([DATA],x=0..2*Pi,0.5..1.5,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,axes=boxed,
  title="Pulse-Values relative to the Mean-Value YM = 57.05"):
> p[2]:=plot(Sp(x),x=0..2*Pi,th=3,co=black):
> p[3]:=plot(1,x=0..2*Pi,linestyle=3,th=1,co=black):
> p[4]:=plots[textplot]([3,1.3,`Cubic Spline Interpolation`]):
> plots[display](seq(p[k],k=1..4));
```



Nonlinear Regression:

```
> f:=(t,a,b,c,d,e,F)->a+b*t*sin(c*t+d)^2*cos(e*t+F)^2;
```

$$f := (t, a, b, c, d, e, F) \rightarrow a + b\,t\,\sin(c\,t + d)^2\,\cos(e\,t + F)^2$$

```
> y:=unapply(evalf(NonlinearFit(f(t,a,b,c,d,e,F),X,Z,t)),t);
```

$$y := t \rightarrow 1.09266851879094 - 0.224820920578031\,t \sin(1.04483494607003\,t + 0.752812847119836)^2 \cos(1.05579891650084\,t + 0.730393401109659)^2$$

```
> y(x):=subs(t=x,y(t));
```

$$y(x) := 1.09266851879094 - 0.224820920578031\,x \sin(1.04483494607003\,x + 0.752812847119836)^2 \cos(1.05579891650084\,x + 0.730393401109659)^2$$

```
> y(0):=evalf(subs(x=0,y(x)));
```

$$y(0) := 1.09266851879094$$

```
> y(0)[data]:=63/YM;
```

$$y(0)_{data} := 1.10438247011952$$

```
> y(2*Pi):=evalf(subs(x=2*Pi,y(x)));
```

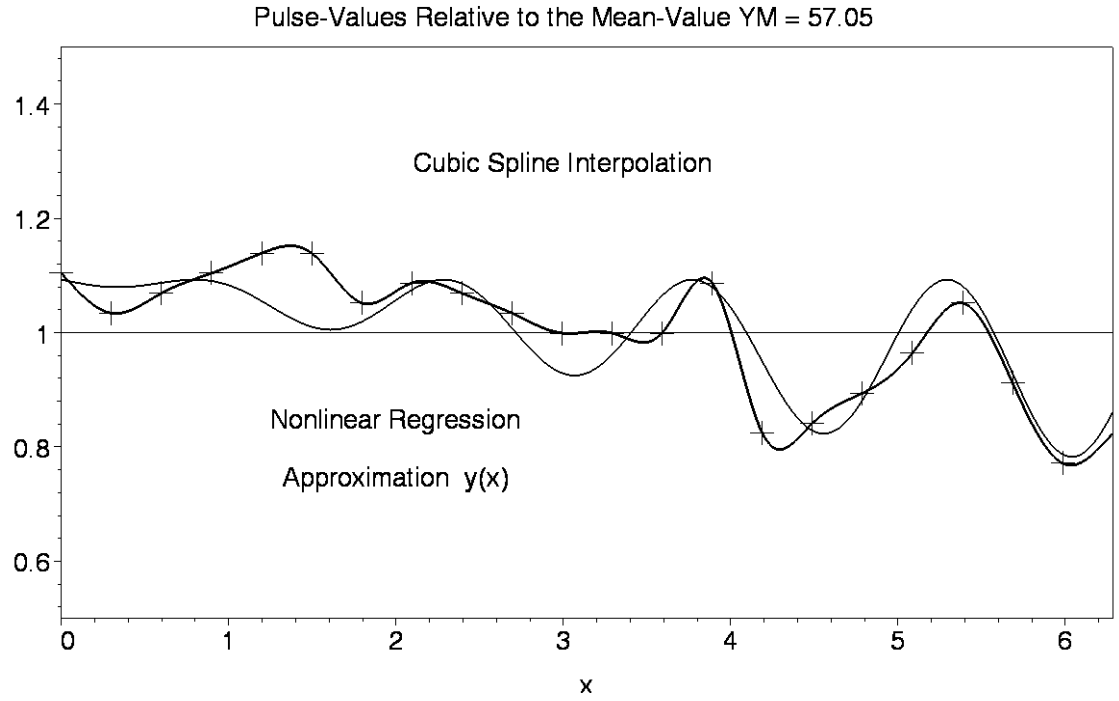
$$y(2\,\pi) := 0.861640200149388$$

```
> y(2*Pi)[data]:=47/YM;
```

$$y(2\,\pi)_{data} := 0.823904382470120$$

```
>
> alias(th=thickness,co=color):
> p[1]:=plot([DATA],x=0..2*Pi,0.5..1.5,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,axes=boxed,
  title="Pulse-Values Relative to the Mean-Value YM = 57.05"):
```

```
> p[2]:=plot(Sp(x),x=0..2*Pi,th=3,co=black):
> p[3]:=plot(y(x),x=0..2*Pi,th=2,co=black):
> p[4]:=plot(1,x=0..2*Pi,linestyle=3,th=1,co=black):
> p[5]:=plots[textplot]({[3,1.3,`Cubic Spline Interpolation`],
  [2,0.85,`Nonlinear Regression`],[2,0.75,`Approximation
  y(x)`]}):
> plots[display](seq(p[k],k=1..5));
```



error norms:

```
> L[2]:=
  sqrt((1/2/Pi)*Int((SPLINE-APPROX)^2,x=0..2*Pi))=
  evalf(sqrt((1/2/Pi)*int((Sp(x)-y(x))^2,x=0..2*Pi)));
```

$$L_2 := \frac{1}{2}\sqrt{2}\sqrt{\frac{1}{\pi}\int_0^{2\pi} (SPLINE - APPROX)^2\, dx} = 0.05356127607$$

```
> with(linalg):
> for i from 1 to 22 do
  v[i]:=evalf(subs(x=DATA[i][1],y(x))-DATA[i][2]) od:
> V:=vector([seq(v[i],i=1..22)]);
```

V := [-0.0113314812090588, 0.0457528640497927, 0.0181098702194375, -0.0136120070822395, -0.0837358516005746, -0.128779622255690, -0.0329208595532596, -0.0122682511941465, 0.0170208809194298, -0.0261279008333893, -0.0700035722952960, -0.0378621710536577, 0.0655444942894189, -0.00769729809977263, 0.125626741032895, -0.0128173799339927, -0.0124657113722093, 0.0777020071766001, 0.0295863098164297, 0.0168280871499092, 0.0157069788319250, 0.0377402001493879]

```
> l[2]:=
  (1/sqrt(number_of_points))*Norm(V,2)=
  evalf((1/sqrt(22))*norm(V,2));
```

$$l_2 := \frac{\mathrm{Norm}(V, 2)}{\sqrt{number\_of\_points}} = 0.0540184456147690$$

>

Approximation of the Splinefunction by *FOURIER*-Series in dimensionless Representation

```
> restart:
> with(Statistics):
> X:=array([seq(2*Pi*(i-1)/21,i=1..22)]);
```

$$X := \left[0, \frac{2\pi}{21}, \frac{4\pi}{21}, \frac{2\pi}{7}, \frac{8\pi}{21}, \frac{10\pi}{21}, \frac{4\pi}{7}, \frac{2\pi}{3}, \frac{16\pi}{21}, \frac{6\pi}{7}, \frac{20\pi}{21}, \frac{22\pi}{21}, \frac{8\pi}{7}, \frac{26\pi}{21}, \frac{4\pi}{3}, \frac{10\pi}{7}, \frac{32\pi}{21}, \frac{34\pi}{21}, \frac{12\pi}{7}, \frac{38\pi}{21}, \frac{40\pi}{21}, 2\pi\right]$$

```
> whattype(X);
```

symbol

```
> Y:=array([63,59,61,63,65,65,60,62,61,59,57,57,57,62,47,48,51,55,
  60,52,44,47]);
```

$$Y := [63, 59, 61, 63, 65, 65, 60, 62, 61, 59, 57, 57, 57, 62, 47, 48, 51, 55, 60, 52, 44, 47]$$

```
> whattype(Y);
```

symbol

```
> YM:=evalf(Mean(Y),4);  # mean value of Y  #  dimesion = bpm
```

$$YM := 57.05$$

```
> Z:=array(evalf([seq(Y[i],i=1..22)/YM],4));  # dimensionless
  pulse values
```

$$Z := [1.104, 1.034, 1.069, 1.104, 1.139, 1.139, 1.052, 1.087, 1.069, 1.034, 0.9992, 0.9992, 0.9992, 1.087, 0.8239, 0.8414, 0.8940, 0.9642, 1.052, 0.9116, 0.7713, 0.8239]$$

```
> whattype(Z);
```

symbol

```
> DATA:=seq([X[i],Z[i]],i=1..22);
```

$$DATA := [0, 1.104], \left[\frac{2\pi}{21}, 1.034\right], \left[\frac{4\pi}{21}, 1.069\right], \left[\frac{2\pi}{7}, 1.104\right], \left[\frac{8\pi}{21}, 1.139\right], \left[\frac{10\pi}{21}, 1.139\right], \left[\frac{4\pi}{7}, 1.052\right], \left[\frac{2\pi}{3}, 1.087\right], \left[\frac{16\pi}{21}, 1.069\right], \left[\frac{6\pi}{7}, 1.034\right], \left[\frac{20\pi}{21}, 0.9992\right], \left[\frac{22\pi}{21}, 0.9992\right], \left[\frac{8\pi}{7}, 0.9992\right], \left[\frac{26\pi}{21}, 1.087\right], \left[\frac{4\pi}{3}, 0.8239\right], \left[\frac{10\pi}{7}, 0.8414\right], \left[\frac{32\pi}{21}, 0.8940\right], \left[\frac{34\pi}{21}, 0.9642\right], \left[\frac{12\pi}{7}, 1.052\right], \left[\frac{38\pi}{21}, 0.9116\right], \left[\frac{40\pi}{21}, 0.7713\right], [2\pi, 0.8239]$$

```
> with(CurveFitting):
> # The cubic splinefunction fit to the data is given by:
> Sp(x):=Spline([DATA],x,degree=3):
> # This function is not printed because of its length. However,
```

```
  printing is possible, if we insert a semicolon (;) instead of
  the doppel point (:) at the end of the command for Sp(x).
> FOURIER_Series(x):=
  a[0]/2+sum(a[k]*cos(k*x)+b[k]*sin(k*x),k=1..infinity);
```

$$\text{FOURIER\_Series}(x) := \frac{1}{2} a_0 + \left(\sum_{k=1}^{\infty} \left(a_k \cos(k\,x) + b_k \sin(k\,x) \right) \right)$$

```
> a[k]:=(1/Pi)*Int(f(x)*cos(k*x),x=0..2*Pi);
```

$$a_k := \frac{1}{\pi} \int_0^{2\,\pi} \mathrm{f}(x) \cos(k\,x)\, dx$$

```
> a[0]:=simplify(subs(k=0,a[k]));
```

$$a_0 := \frac{1}{\pi} \int_0^{2\,\pi} \mathrm{f}(x)\, dx$$

```
> b[k]:=(1/Pi)*Int(f(x)*sin(k*x),x=0..2*Pi);
```

$$b_k := \frac{1}{\pi} \int_0^{2\,\pi} \mathrm{f}(x) \sin(k\,x)\, dx$$

```
> with(CurveFitting):
> F(x):=Sp(x):
> A[0]:=evalf(subs(f(x)=F(x),a[0]));
```

$$A_0 := 2.001661937$$

```
> A[k]:=simplify(value(subs(f(x)=F(x),a[k]))):
> A[k]:=subs({sin(k*Pi)=0.,(cos(k*Pi))^2=1.},%):
> A[k]:=evalf(%):
> for i in [seq(k,k=1..5)] do A[i]:=subs(k=i,A[k]) od:
> for i in [seq(k,k=1..15)] do P[i]:=subs(k=i,A[k]) od:
> for i in [seq(k,k=1..100)] do R[i]:=subs(k=i,A[k]) od:
> B[k]:=simplify(value(subs(f(x)=F(x),b[k]))):
> B[k]:=subs({sin(k*Pi)=0.,(cos(k*Pi))^2=1.},%):
> for i in [seq(k,k=1..5)] do B[i]:=subs(k=i,B[k]) od:
> for i in [seq(k,k=1..15)] do Q[i]:=subs(k=i,B[k]) od:
> for i in [seq(k,k=1..100)] do S[i]:=subs(k=i,B[k]) od:
> # The coefficients A[i]...S[i] are not printed because of their
  length. However,
> # it can be done by inserting semicolons (;) instead of
  doppelpoints (:) at the # ends of the commands. The
  FOURIER-Series with k = 5, 15, 100 are expressed by # y(x),
  z(x), T(x), respectively, as follows:
>
> y(x):=evalf(A[0]/2+sum(A[k]*cos(k*x)+B[k]*sin(k*x),k=1..5));
```

$$\begin{aligned} y(x) := {} & 1.000830968 - 0.01825313978 \cos(x) + 0.1006481742 \sin(x) \\ & - 0.01830805872 \cos(2.\,x) + 0.01773632949 \sin(2.\,x) - 0.04520463741 \cos(3.\,x) \\ & - 0.005376925601 \sin(3.\,x) - 0.03759917455 \cos(4.\,x) + 0.04490080094 \sin(4.\,x) \\ & + 0.02997855942 \cos(5.\,x) + 0.03300989107 \sin(5.\,x) \end{aligned}$$

```
> y(0):=simplify(subs(x=0,y(x)));
```

$$y(0) := 0.9114445169$$

```
> y(0)[data]:=63/YM;
```

$$y(0)_{data} := 1.104294479$$

```
> y(2*Pi):=simplify(subs(x=2*Pi,y(x)));
```

$$y(2\,\pi) := 0.9114445173$$

```
> y(2*Pi)[data]:=47/YM;
```

$$y(2\,\pi)_{data} := 0.8238387381$$

```
> z(x):=evalf(A[0]/2+sum(P[k]*cos(k*x)+Q[k]*sin(k*x),k=1..15));
```

$$\begin{aligned} z(x) := {} & -0.018253144\cos(x) + 0.1006481786\sin(x) + 0.004293500446\cos(13.\,x) \\ & + 0.00959191560\sin(12.\,x) - 0.0002209624171\cos(12.\,x) + 0.003122387705\cos(11.\,x) \\ & + 0.00098482511\sin(11.\,x) + 0.01934109920\sin(10.\,x) + 0.004129050911\cos(10.\,x) \\ & - 0.002720989955\cos(9.\,x) + 0.00307326010\sin(9.\,x) + 0.02453893885\cos(8.\,x) \\ & - 0.00283646628\sin(8.\,x) + 0.02993027422\sin(7.\,x) + 0.003127297193\cos(7.\,x) \\ & + 0.0007299447424\cos(6.\,x) + 0.01860477310\sin(6.\,x) + 0.0008050868405\cos(15.\,x) \\ & + 0.00584797498\sin(15.\,x) + 0.001001552415\cos(14.\,x) + 0.00529394883\sin(14.\,x) \\ & + 0.00886349836\sin(13.\,x) + 0.02997855942\cos(5.\,x) + 0.03300989099\sin(5.\,x) \\ & - 0.03759917454\cos(4.\,x) + 0.0449008008\sin(4.\,x) - 0.04520463744\cos(3.\,x) \\ & - 0.0053769260\sin(3.\,x) - 0.01830805854\cos(2.\,x) + 0.0177363298\sin(2.\,x) \\ & + 1.000830968 \end{aligned}$$

```
> z(0):=simplify(subs(x=0,z(x)));
```

$$z(0) := 0.9502503196$$

```
> z(0)[data]:=63/YM;
```

$$z(0)_{data} := 1.104294479$$

```
> z(2*Pi):=simplify(subs(x=2*Pi,z(x)));
```

$$z(2\,\pi) := 0.9502503209$$

```
> z(2*Pi)[data]:=47/YM;
```

$$z(2\,\pi)_{data} := 0.8238387381$$

```
>
> T(x):=evalf(A[0]/2+sum(R[k]*cos(k*x)+S[k]*sin(k*x),k=1..100)):
> # This equation has not be printed because of its length with
  more than 100 terms.However, printing is possible, if we insert
  a semicolon (;) instead of the doppelpoint (:) at the end of the
  command for T(x).
> T(0):=simplify(subs(x=0,T(x)));
```

$$T(0) := 0.9618525872$$

```
> T(0)[data]:=63/YM;
```

$$T(0)_{data} := 1.104294479$$

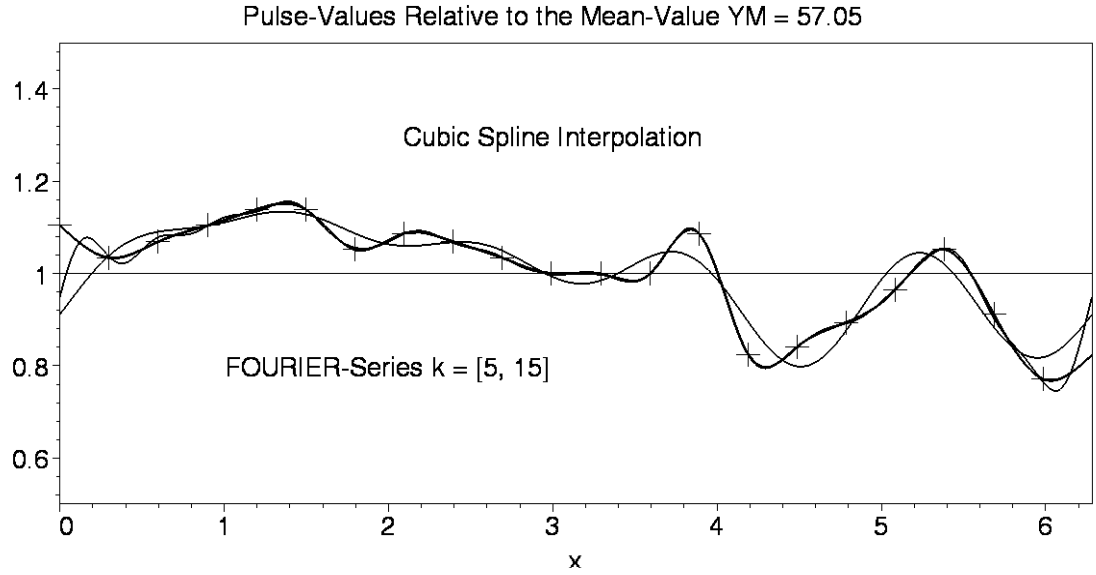
```
> T(2*Pi):=simplify(subs(x=2*Pi,T(x)));
```

$$T(2\,\pi) := 0.9618525945$$

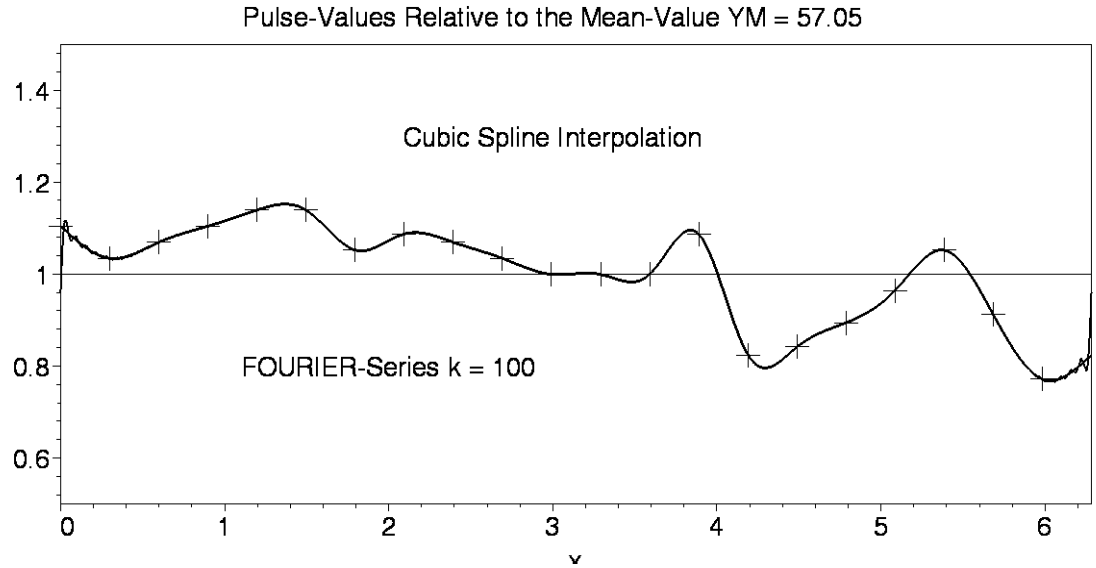
```
> T(2*Pi)[data]:=47/YM;
```

$$T(2\pi)_{data} := 0.8238387381$$

```
> # The following Figure illustrades the approximations of the
  splinefunctions by FOURIER-Series with k = [5, 15]:
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,0.5..1.5,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,
  style=point,symbol=cross,symbolsize=50,
  title="Pulse-Values Relative to the Mean-Value YM = 57.05"):
> p[3]:=plot({y(x),z(x)},x=0..2*Pi,th=2,co=black):
> p[4]:=plot(1,x=0..2*Pi,linestyle=3,th=1,co=black):
> p[5]:=plots[textplot]({[3,1.3,`Cubic Spline
  Interpolation`],[2,0.8,`FOURIER-Series k = [5, 15]`]}):
> plots[display](seq(p[k],k=1..5));
```




```
> # The next Figure represents the approximation of the
  splinefunction by a FOURIER-Series with k = 100:
> alias(th=thickness,co=color):
> p[1]:=plot(Sp(x),x=0..2*Pi,0.5..1.5,axes=boxed,th=3,co=black):
> p[2]:=plot([DATA],x=0..2*Pi,th=3,co=black,style=point,symbol=cro
  ss, symbolsize=50,title="Pulse-Values Relative to the Mean-Value
  YM = 57.05"):
> p[3]:=plot(T(x),x=0..2*Pi,th=2,co=black):
> p[4]:=plot(1,x=0..2*Pi,linestyle=3,th=1,co=black):
> p[5]:=plots[textplot]({[3,1.3,`Cubic Spline
  Interpolation`],[2,0.8,`FOURIER-Series k = 100`]}):
> plots[display](seq(p[k],k=1..5));
```



In this Figure we cannot observe differences between the splinefunction and the FOURIER-Series. Only immediately at the beginning x = 0 and at the end x = 2*Pi there are some differences:

```
> T(0):=simplify(subs(x=0,T(x)));
  T(0.0207):=simplify(subs(x=0.0207,T(x)));
```

$$T(0) := 0.9618525872$$

$$T(0.0207) := 1.104157090$$

```
> T(0)[data]:=63/YM;
```

$$T(0)_{data} := 1.104294479$$

```
> T(2*Pi):=simplify(subs(x=2*Pi,T(x)));
  T(1.9942*Pi):=simplify(subs(x=1.9942*Pi,T(x)));
```

$$T(2\,\pi) := 0.9618525945$$

$$T(6.264964071) := 0.8228604027$$

```
> T(2*Pi)[data]:=47/YM;
```

$$T(2\,\pi)_{data} := 0.8238387381$$

The *error norms* L[2](k= 5,15,100), of the approximations y(x), z(x), and T(x) with respect to the cubic splinefunction can be expressed as:

```
> L[2][k=5]:=
  sqrt((1/5.685)*Int((SP(xi)-y(xi))^2,xi=0.299..5.884))=
  sqrt((1/5.685)*int((Sp(x)-y(x))^2,x=0.299..5.984));
```

$$L_{2_{k=5}} := 0.4194061220 \sqrt{\int_{0.299}^{5.884} (SP(\xi) - y(\xi))^2\, d\xi} = 0.02973952528$$

```
> L[2][k=15]:=
  sqrt((1/5.685)*Int((SP(xi)-z(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-z(x))^2,x=0.299..5.984));
```

$$L_{2_{k=15}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (\mathrm{SP}(\xi) - \mathrm{z}(\xi))^2 \, d\xi} = 0.004174537492$$

```
> L[2][k=100]:=
  sqrt((1/5.685)*Int((SP(xi)-T(xi))^2,xi=0.299..5.984))=
  sqrt((1/5.685)*int((Sp(x)-T(x))^2,x=0.299..5.984));
```

$$L_{2_{k=100}} := 0.4194061220 \sqrt{\int_{0.299}^{5.984} (\mathrm{SP}(\xi) - \mathrm{T}(\xi))^2 \, d\xi} = 0.0006794214249$$

```
>
```

The three values above show the L[2] *error norms* of the FOURIER-Series y(x), z(x), and T(x) relative to the *cubic splinefunction.* Furthermore, one should discuss the *error norms* with respect to the given measured data, as follows:

```
> with(linalg):
> for i from 2 to 21 do
  u[i]:=subs(x=DATA[i][1],y(x))-DATA[i][2] od:
> U:=evalf(vector([seq(u[i],i=2..21)]),4);
```

U := [0.00476, 0.02145, -0.00058, -0.01125, -0.0108, 0.03477, -0.02718, -0.00102, 0.01562, -0.00552, -0.01503, 0.03711, -0.06172, 0.07168, -0.04284, -0.01674, 0.05665, -0.03253, -0.02779, 0.04708]

```
> l[2][k=5]:=
  (1/sqrt(number_of_points))*Norm(U,2)=
  evalf((1/sqrt(20))*norm(U,2),4);
```

$$l_{2_{k=5}} := \frac{\mathrm{Norm}(U, 2)}{\sqrt{number\_of\_points}} = 0.03383$$

```
> for i from 2 to 21 do
  v[i]:=subs(x=DATA[i][1],z(x))-DATA[i][2] od:
> V:=evalf(vector([seq(v[i],i=2..21)]),4);
```

V := [0.00241, 0.00974, -0.00089, -0.00523, 0.00074, 0.00406, -0.00120, -0.00279, 0.00126, 0.00282, -0.00197, -0.00211, 0.00088, 0.00266, -0.00300, -0.00180, 0.00469, 0.00199, -0.00840, -0.00680]

```
> l[2][k=15]:=
  (1/sqrt(number_of_points))*Norm(V,2)=
  evalf((1/sqrt(20))*norm(V,2),4);
```

$$l_{2_{k=15}} := \frac{\mathrm{Norm}(V, 2)}{\sqrt{number\_of\_points}} = 0.004099$$

```
> for i from 2 to 21 do
  w[i]:=subs(x=DATA[i][1],T(x))-DATA[i][2] od:
> W:=evalf(vector([seq(w[i],i=2..21)]),4);
```

W := [-0.00060, 0.00135, 0.00065, -0.00051, -0.0007, 0.00005, 0.00068, 0.00003, -0.00071, -0.00039, -0.00025, -0.00012, -0.00062, -0.00034, -0.00014, 0.0007, 0.00049, 0.00009, -0.00100, 0.00030]

```
> l[2][k=100]:=
```

```
(1/sqrt(number_of_points))*Norm(W,2)=
evalf((1/sqrt(20))*norm(W,2),4);
```

$$l_{2_{k=100}} := \frac{\mathrm{Norm}(W, 2)}{\sqrt{number\_of\_points}} = 0.0005874$$

Résumé: The above *error norms* - L[2] and l[2] - show that the FOURIER-Series y(x), z(x), and T(x) are *best approximations* fit to the given measured data. Furthermore, the *Nonlinear Regression* furnish *nonlinear model functions* suitable to fit the given data, too.

>