Oplossingen NMB2: Kleinste-kwadratenbenadering (deel 2)

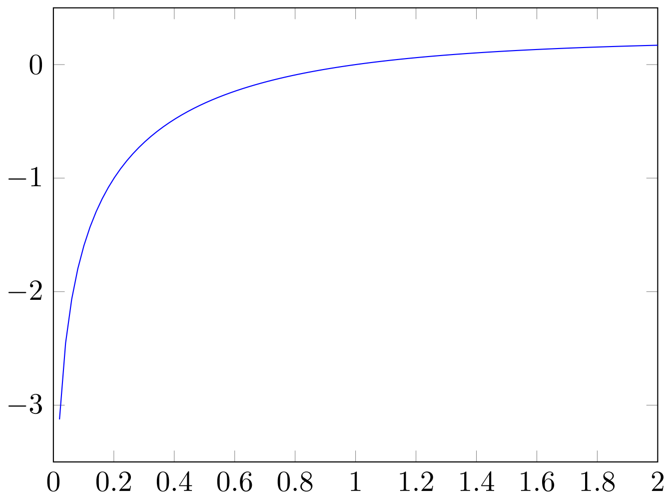
Michiel Vandecappelle, Nico Vervliet, *KU Leuven*

20 maart 2018

1 Opgave 1

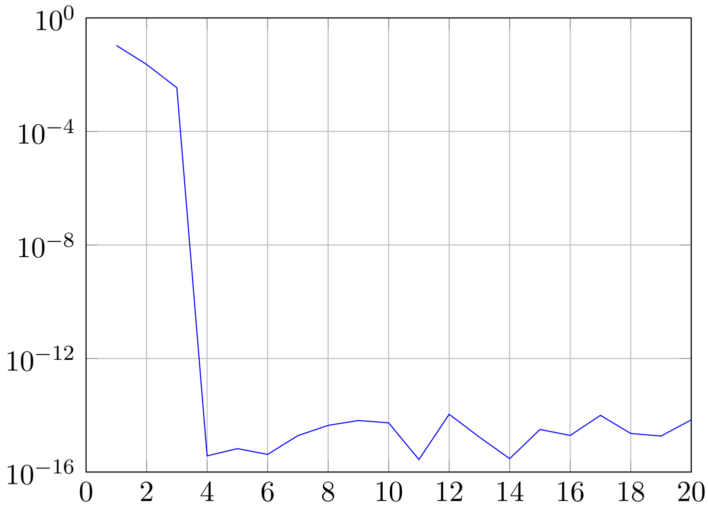
Deel a

```
%% Plot f
x = linspace(0.0001,2,100);
f = expint(1) - expint(x);
close all;
figure();
plot(x,f);
```



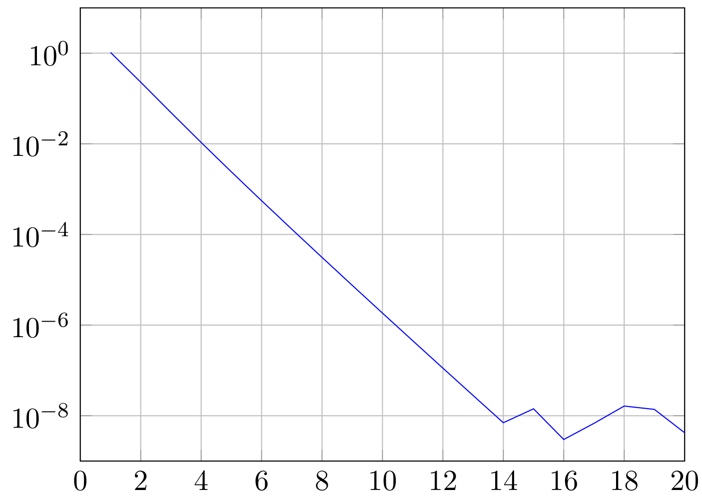
Deel b

```
%% Define x-values, function values and weights
N = 5; % Choose number of approximation points
x = linspace(1/2,3/2,N)'; % x-values
f = expint(1) - expint(x); % Function values
w = ones(size(x)); % Weights

%% Compute least-squares polynomial approximation for different degrees
nmax = 20; % Set max degree
res = zeros(1,nmax);
for k = 1:nmax
c = kkb1(x,f,w,k);
r = polyval(c(end:-1:1),x); % Evaluate polynomial approximation
res(k) = norm(f-r); % Compute residue
end

%% Plot residues for different degrees
close all;
figure;
semilogy(1:nmax,res);
grid on;
```



Graad groter dan $N-1$ met N het aantal punten, is zinloos, aangezien je vanaf graad $N-1$ interpoleert.

```
N = 1000;
```



Deel c

Gegeven:

$$f(x) \approx y_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n$$

Gevraagd:

$$\int f(x) \approx \int y_n(x) = a_0 x + \frac{a_1}{2} x^2 + \cdots \frac{a_n}{n+1} x^{n+1} = Y_n(x)$$

Dus de oplossing is $Y_n(\frac{3}{2}) - Y_n(\frac{1}{2})$.

```
function I = integraal(x,f,n)
%INTEGRAAL computes the integral of the function with function values f
%evaluated in the points x by computing the integral of the discrete
% least-squares polynomial approximation of degree n.
% INPUTS
% x     Vector holding the points in which the function values are given
% f     Function values in the points x
% n     Degree of the polynomial approximation
% OUTPUTS
% I     Value of the integral

% Compute coefficients of the integral approximation
w = ones(size(x)); % Set weights
c = kkb1(x,f,w,n); % Coefficients of function approximation
cint = [0;c./(1:n+1)']; % Coefficients of integral approximation

% Compute integral by subtracting the values in the upper and lower limits
Ilimits = polyval(cint(end:-1:1),[x(end),x(1)]); % Upper and lower limit
I = Ilimits(1) - Ilimits(2); % Subtract
```

```
end
```

```
%% Compute integral for different degrees

% Set up x and f values
format long % Show enough digits precision
N = 1000; % Number of evaluation points
x = linspace(1/2,3/2,N)';
f = expint(1) - expint(x); % Function values
nmax = 15;

% Compute integral
for n = 1:nmax
        disp(n);
        disp(integraal(x, f, n));
end
```

```
I

I =

  -0.034235583684441
  -0.034163937235641
  -0.034163937235640
  -0.034159422383244
  -0.034159422383242
  -0.034159158653888
  -0.034159158653889
  -0.034159142484693
  -0.034159142484680
  -0.034159141461244
  -0.034159141461195
  -0.034159141395035
  -0.034159141395065
  -0.034159141391664
  -0.034159141390239
```

Als cijfers niet meer veranderen, dan veronderstellen we dat deze juist zijn. Wel opletten bij te hoge graad: fout kan terug gaan stijgen door afrondingsfouten.

Deel d

Dezelfde werkwijze als in bovenstaande oefening kan gehanteerd worden:

$$y_n'(x) = a_1 + 2a_2(x) + \cdots + na_nx^{n-1}$$

```
function res = afgeleide(x,f,n)
%AFGELEIDE computes the residue of the approximation of the differential of
% the function with function values f evaluated in the points x when
% approximated by the differential of the discrete least-squares polynomial
% approximation of degree n.
% INPUTS
% x     Vector holding the points in which the function values are given
% f     Function values in the points x
% n     Degree of the polynomial approximation
% OUTPUTS
% res   Max of the residue of the approximation over the x-values

% Compute coefficients of the integral approximation
w = ones(size(x)); % Set weights
ftrue = exp(-x)./x; % Evalue analytic differential

c = kkb1(x,f,w,n); % Coefficients of function approximation
cdiff = c.*(0:n)'; % Coefficients of differential approximation

% Evaluate differential and compute maximum of residue
approxdiff = polyval(cdiff(end:-1:2),x); % Evaluate differential
res = max((ftrue-approxdiff)./ftrue); % Norm of residue

end
```

```
%% Compute residue of differential approximation for different degrees

% Set up x and f values
format long % Show enough digits precision
N = 1000; % Number of evaluation points
x = linspace(1/2,3/2,N)';
f = expint(1) - expint(x); % Function values
nmax = 15;

% Compute differential
for n = 1:nmax
        disp(n);
        disp(afgeleide(x, f, n));
```

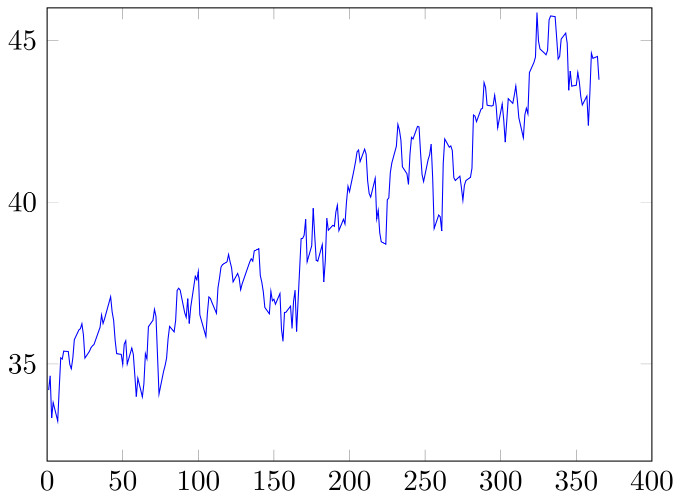
```
end
```

```
D

D =

   8.964265362049082
   3.235817705398087
   1.007276738254360
   0.297226946590354
   0.085903881853611
   0.024562456606252
   0.006971736607421
   0.001967575684390
   0.000552719470774
   0.000154667796682
   0.000043139754368
   0.000011999149800
   0.000003330175545
   0.000000919212936
   0.000001842803457
```

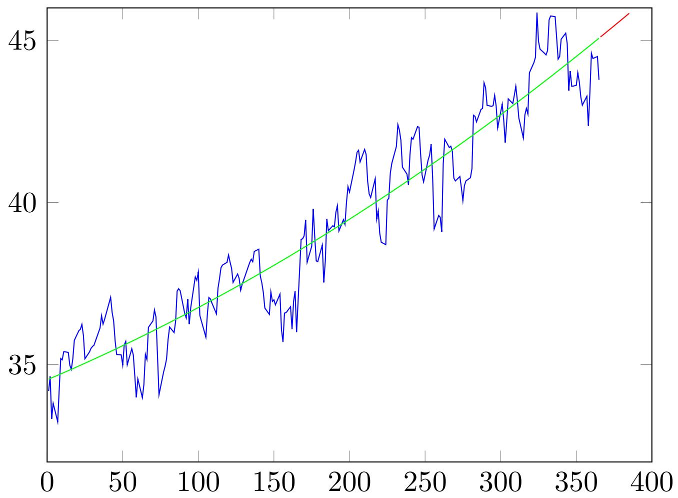
Afleiden is minder nauwkeurig dan integreren.

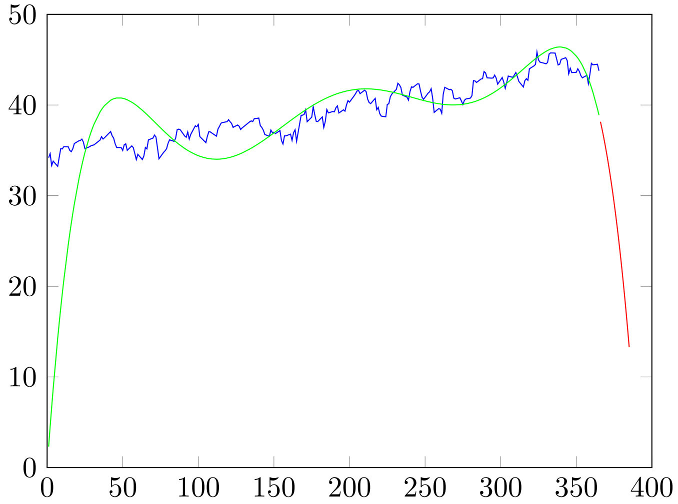
2 Opgave 2

```
%% Plot Inbev data
close all;
figure();
inbev = load('abinbev');
data = inbev.data;
plot(data(:,1), data(:,2));
```




```matlab
%% Approximate with polynomial
% Set up x-values, function values, weights and degree
x = data(:,1); % x-values
f = data(:,2); % Function values
w = ones(size(x)); % Weights
n = 4;

% Compute approximation
c = kkb1(x,f,w,n); % Compute coefficients
finter = polyval(c(end:-1:1), x); % Evaluate approximation

%% Evaluate in next days by extrapolation
next = 10; % Set how many days to extrapolate
xextra = x(end)+(1:next)'; % Set x-values for extrapolation
fextra = polyval(c(end:-1:1), xextra); % Evaluate approximation

%% Plot approximation and extrapolation together
hold on;
plot(x, finter, 'g')
plot(xextra, fextra, 'r')
```





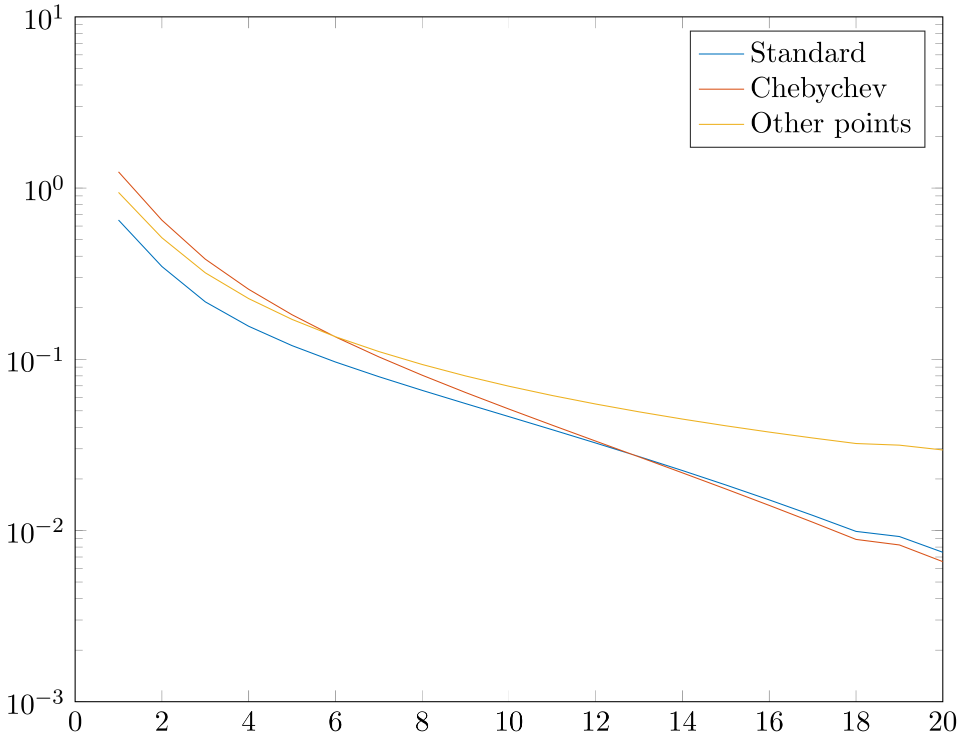
Extrapolatie is niet echt betrouwbaar. Als de tijd naar oneindig gaat, gaat de koers ook naar $\pm\infty$.

3 Opgave 3

```
%% Initialize stuff
nmax = 20; % Maximal approximation degree
res = zeros(nmax, 4); % Matrix holding all norms of residuals

%% Plot arcsine
x = linspace(0,1,100)';
f = asin(x);
```

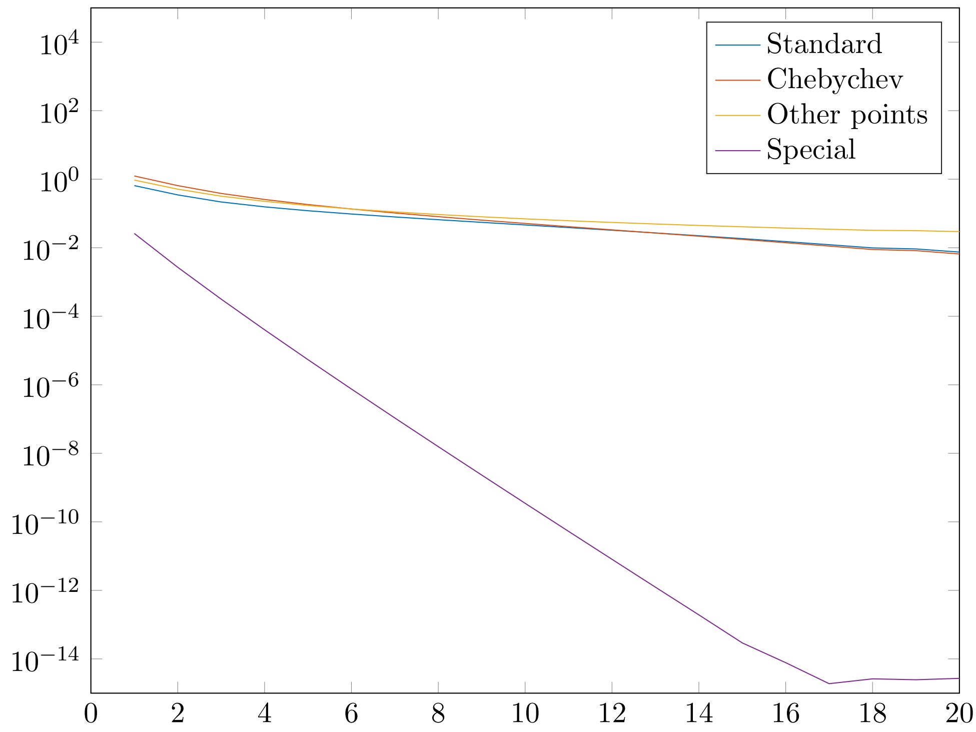
```matlab
w = ones(size(x));
close all;
figure();
plot(x, f);

%% First approximation: equidistant points, w = 1
ex = 1;
x = linspace(0,1,100)';
f = asin(x);
w = ones(size(x));
approx = zeros(length(x),nmax);

% Compute approximation for different degrees
for n = 1:nmax
c = kkb1(x,f,w,n);
approx(:,n) = polyval(c(end:-1:1),x);
res(n,ex) = norm(f-approx(:,n));
end
% Plot degree five approximation
hold on;
plot(x,approx(:,5));

%% Second approximation: equidistant points, Chebychev weight function
ex = 2;
x = linspace(0,0.9999,100)';
f = asin(x);
w = 1./sqrt(1-x.^2);
approx = zeros(length(x),nmax);

% Compute approximation for different degrees
for n = 1:nmax
c = kkb1(x,f,w,n);
approx(:,n) = polyval(c(end:-1:1),x);
res(n,ex) = norm(f-approx(:,n));
end
% Plot degree five approximation
plot(x,approx(:,5));

%% Third approxmation: cosine points, w = 1
ex = 3;
x = cos(linspace(-pi/2,0,100))';
f = asin(x);
w = ones(size(x));
approx = zeros(length(x),nmax);
```

```
% Compute approximation for different degrees
for n = 1:nmax
c = kkb1(x,f,w,n);
approx(:,n) = polyval(c(end:-1:1),x);
res(n,ex) = norm(f-approx(:,n));
end
% Plot degree five approximation
plot(x,approx(:,5));
legend('arcsin','Standard', 'Chebychev', 'Other points');

%% Plot residuals for different degrees
figure();
semilogy(1:nmax,res(:,1));
hold on;
semilogy(1:nmax,res(:,2));
semilogy(1:nmax,res(:,3));
legend('Standard', 'Chebychev', 'Other points');
```



De functie is niet veeltermachtig, vanwege de asymptoot in $x = 1$. Wat je ook probeert, niets gaat echt helpen. De tweede benadering geeft wel zeer goede resultaten.

```matlab
%% Compute approximation of special form
ex = 4;
x = linspace(0,0.9999,100)';
w = ones(size(x)); % Weights
f = asin(x);

% Approximate transformation of f with polynomial
ftrans = (pi/2 - asin(x))./sqrt(1-x); % Isolate polynomial part of approximation


for n = 1:nmax
c = kkb1(x,ftrans,w,n); % Compute polynomial approximation of transformed f
polyapprox = polyval(c(end:-1:1),x); % Evaluate polynomial part of approximation
approx(:,n) = -sqrt(1-x).*polyapprox+pi/2; % Transform again to approximation of f
res(n,ex) = norm(f-approx(:,n));
end

%% Plot approximation of special form
% Plot degree five approximation
figure(1);
hold on;
plot(x, approx(:,5));
legend('arcsin','Standard', 'Chebychev', 'Other points','Special');

% Plot residual
figure(2);
hold on;
semilogy(1:nmax, res(:,4));
legend('Standard', 'Chebychev', 'Other points','Special');
```


Standard
Chebychev
Other points
Special
10^{4}
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
2
4
6
8
10
12
14
16
18
20