Approximation of polynomial functions by wavelet scaling functions (vanishing moments)

In[91]:=
```
fp = FourierParameters → {0, -2 Pi};
```

Approximating with D4

the D4 low-pass filter

In[92]:=
```
wv2 = WaveletFilterCoefficients[DaubechiesWavelet[2],
   "PrimalLowpass",
   WorkingPrecision → 5];
wv2 // MatrixForm
```

Out[92]//MatrixForm=

$$\begin{pmatrix} 0 & 0.34151 \\ 1 & 0.59151 \\ 2 & 0.15849 \\ 3 & -0.091506 \end{pmatrix}$$

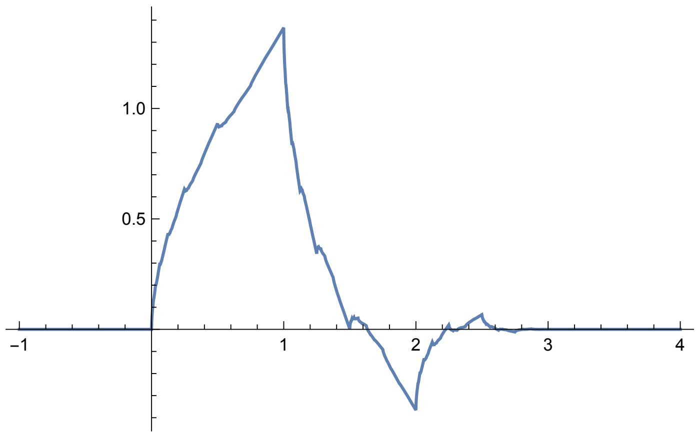
the scaling function from the *Mathematica* system

In[106]:=
```
dwv2 = WaveletPhi[DaubechiesWavelet[2], MaxRecursion → 15]
```

Out[106]=
```
{ InterpolatingFunction[ Domain: {{0., 3.}} Output: scalar ][#1]   0 ≤ #1 ≤ 3  &
{ 0                                                               True
```

In[107]:=
```
Plot[dwv2[t], {t, -1, 4}]
```

Out[107]=



the Fourier series of the filter

In[93]:=
```
m02[s_] :=
 Sum[wv2[[j + 1, 2]] Exp[-2 Pi I s j], {j, 0, 3}]
```

In[94]:=
```
m02[s]
```

Out[94]= $0.34151 + 0.59151\, e^{-2 i \pi s} + 0.15849\, e^{-4 i \pi s} - 0.091506\, e^{-6 i \pi s}$

checking orthogonality

In[95]:=
```
Assuming[s ∈ Reals, FullSimplify[
  Expand[m02[s] * Conjugate[m02[s]] +
    + Expand[m02[s + 1 / 2] * Conjugate[m02[s + 1 / 2]]]]]]
```

Out[95]= $1.0000 + 0.\times 10^{-5}\,\mathrm{Cos}[2\pi s] + 0.\times 10^{-6}\,\mathrm{Cos}[4\pi s] + 0.\times 10^{-6}\,\mathrm{Cos}[6\pi s] + 0.\times 10^{-5}\, i\,\mathrm{Sin}[2\pi s] + 0.\times 10^{-6}\, i\,\mathrm{Sin}[4\pi s] + 0.\times 10^{-6}\, i\,\mathrm{Sin}[6\pi s]$

vanishing moments of D4

In[96]:=
```
m02[s] /. s → 1 / 2
```

Out[96]= $0.\times 10^{-5}$

In[97]:=
```
D[m02[s], s] /. s → 1 / 2
```

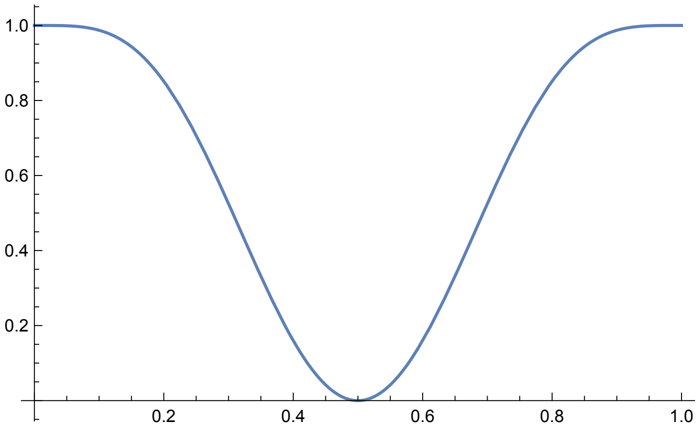
Out[97]= $0.\times 10^{-5}\, i$

In[98]:=
```
D[m02[s], {s, 2}] /. s → 1 / 2
```

Out[98]= -34.189

the filter characteristics of D4

In[99]:=
```
Plot[Abs[m02[s]], {s, 0, 1}]
```

Out[99]=


the finite products of the Fourier series

In[284]:=
```
m2[L_, s_] := Expand[Product[m02[s / (2 ^ j)], {j, 1, L}]]
```

In[285]:=
```
m2[2, s]
```

Out[285]= $0.11663 + 0.20200\, e^{-\frac{1}{2} i \pi s} + 0.25613\, e^{-i \pi s} + 0.31863\, e^{-\frac{3}{2} i \pi s} + 0.14788\, e^{-2 i \pi s} + 0.03962\, e^{-\frac{5}{2} i \pi s} - 0.00613\, e^{-3 i \pi s} - 0.06863\, e^{-\frac{7}{2} i \pi s} - 0.014503\, e^{-4 i \pi s} + 0.008373\, e^{-\frac{9}{2} i \pi s}$

band-limiting of the finite products

In[496]:=
```
mm2[L_, s_] := m2[L, s] * UnitBox[s / 2 ^ (L - 1)]
```

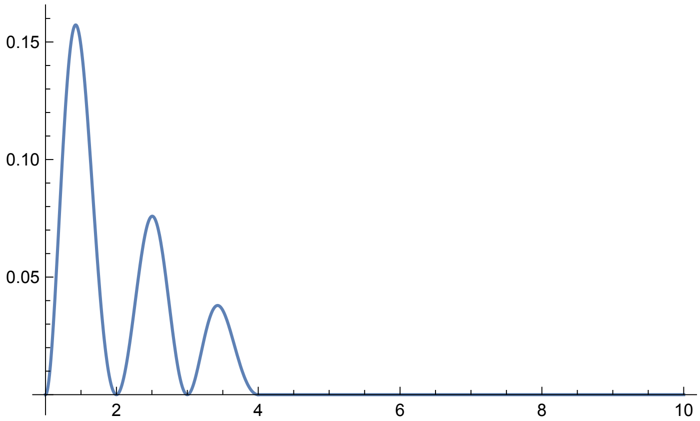
In[505]:=
```
ExpToTrig[mm2[2, s]]
```

Out[505]=

$$\left(0.11663 + 0.20200 \left(\text{Cos}\left[\frac{\pi s}{2}\right] - i\,\text{Sin}\left[\frac{\pi s}{2}\right]\right) + 0.25613 \left(\text{Cos}[\pi s] - i\,\text{Sin}[\pi s]\right) + 0.31863 \left(\text{Cos}\left[\frac{3\pi s}{2}\right] - i\,\text{Sin}\left[\frac{3\pi s}{2}\right]\right) + 0.14788 \left(\text{Cos}[2\pi s] - i\,\text{Sin}[2\pi s]\right) + 0.03962 \left(\text{Cos}\left[\frac{5\pi s}{2}\right] - i\,\text{Sin}\left[\frac{5\pi s}{2}\right]\right) - 0.00613 \left(\text{Cos}[3\pi s] - i\,\text{Sin}[3\pi s]\right) - 0.06863 \left(\text{Cos}\left[\frac{7\pi s}{2}\right] - i\,\text{Sin}\left[\frac{7\pi s}{2}\right]\right) - 0.014503 \left(\text{Cos}[4\pi s] - i\,\text{Sin}[4\pi s]\right) + 0.008373 \left(\text{Cos}\left[\frac{9\pi s}{2}\right] - i\,\text{Sin}\left[\frac{9\pi s}{2}\right]\right)\right) \text{UnitBox}\left[\frac{s}{2}\right]$$

In[512]:=
```
Plot[{Abs[mm2[4, s]]}, {s, 1, 10}, PlotRange → All]
```

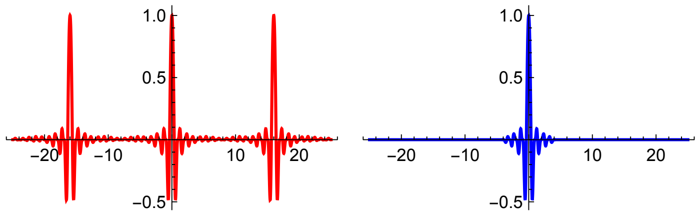
Out[512]=



comparing finite products and their band-limited versions

In[527]:=
```
GraphicsRow[
 {Plot[{Re[m2[4, s]]}, {s, -25, 25}, PlotRange → All, PlotStyle → {Red}],
  Plot[{Re[mm2[4, s]]}, {s, -25, 25}, PlotRange → All, PlotStyle → {Blue}]}]
```

Out[527]=



approximating the D4 scaling function via finite band-limited products

In[460]:=
```
IFT[t_, n_] := InverseFourierTransform[
   Expand[mm2[n, s]], s, t, fp];
```

In[461]:=
```
IFT[t, 3]
```

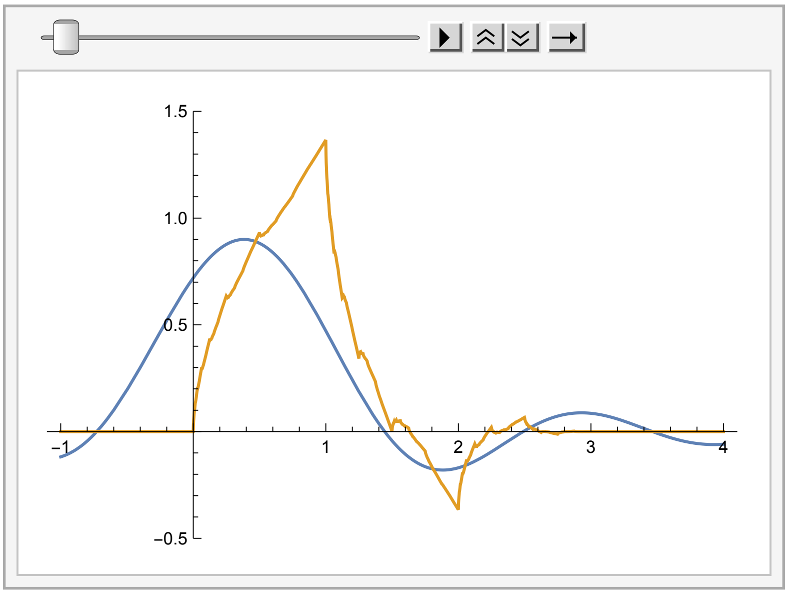
Out[461]=
```
0.276 Sinc[1.571 - 12.57 t] + 0.350 Sinc[3.14 - 12.57 t] +
 0.435 Sinc[4.71 - 12.57 t] + 0.478 Sinc[6.28 - 12.57 t] + 0.532 Sinc[7.85 - 12.57 t] +
 0.598 Sinc[9.42 - 12.57 t] + 0.660 Sinc[11.00 - 12.57 t] +
 0.404 Sinc[12.57 - 12.57 t] + 0.233 Sinc[14.14 - 12.57 t] +
 0.1479 Sinc[15.71 - 12.57 t] + 0.0396 Sinc[17.3 - 12.57 t] +
 0.0167 Sinc[18.8 - 12.57 t] - 0.0290 Sinc[20.4 - 12.57 t] -
 0.0976 Sinc[22.0 - 12.57 t] - 0.1601 Sinc[23.6 - 12.57 t] -
 0.0633 Sinc[25.1 - 12.57 t] - 0.00919 Sinc[26.7 - 12.57 t] +
 0.00224 Sinc[28.3 - 12.57 t] + 0.0251 Sinc[29.8 - 12.57 t] +
 0.00531 Sinc[31.4 - 12.57 t] - 0.00306 Sinc[33.0 - 12.57 t] + 0.1593 Sinc[12.57 t]
```

In[463]:=
```
cv[t_, n_] :=
 Map[Integrate[#, {x, -∞, ∞}] &, Expand[IFT[x, n] * 2^n * UnitBox[2^n (t - x)] ]]
```

In[464]:=
```
Do[cvn = cv[t, n], {n, 1, 5}];
```

In[468]:=
```
ListAnimate[
 Table[Plot[{cvn, dwv2[t]}, {t, -1, 4}, PlotRange → {-0.5, 1.5}], {n, 1, 5}]]
```

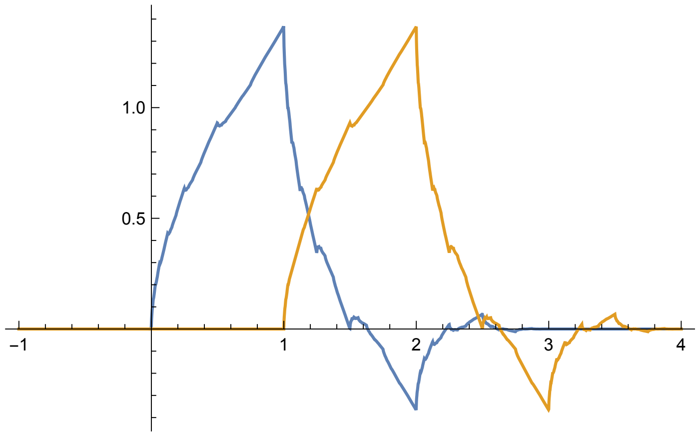
Out[468]=



In[470]:=
```
ift10 = IFT[t, 10];
```

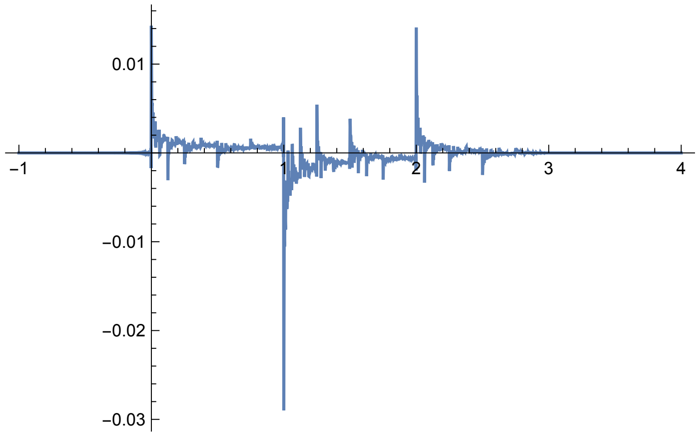
In[471]:=
```
Plot[{ift10, dwv2[t - 1]}, {t, -1, 4}, PlotRange → All]
```

Out[471]=



In[472]:=
```
Plot[{ift10 - dwv2[t]}, {t, -1, 4}, PlotRange → All]
```

Out[472]=



computing inner products numerically

In[108]:=
```
ρ2[poly_, l_] :=
 NIntegrate[(poly /. t → t - l) * dwv2[t], {t, -1, 4}, WorkingPrecision → 4]
```

In[109]:=
```
Table[{l, ρ2[-t + 1, l]}, {l, -3, 3}] // MatrixForm
```

Out[109]//MatrixForm=

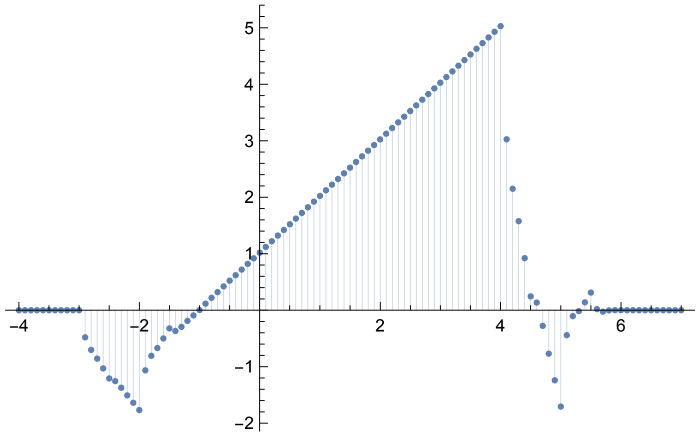
-3	-2.658
-2	-1.656
-1	-0.6537
0	0.3450
1	1.295
2	2.182
3	3.069

the approximation of a polynomial function if degree < 2 (discrete display)

In[489]:=
```
sd2[poly_, low_, high_, step_] :=
 Module[{r},
  r = Table[ρ2[poly, l], {l, -3, 3}];
  Table[{t, Sum[r[[l + 4]] * dwv2[t + l], {l, -3, 3}]}, {t, low, high, step}]
 ]
```

In[490]:=
```
ListPlot[sd2[t + 1, -4, 7, 0.1], Filling → Axis, PlotRange → All]
```

Out[490]=

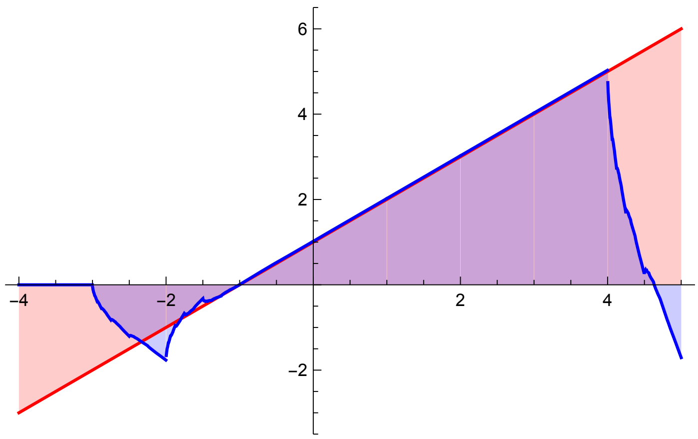


the approximation of a polynomial function if degree < 2 (continuous display)

In[153]:=
```
sc2[poly_, low_, high_] := Module[{r, fn},
  r = Table[ρ2[poly, l], {l, -3, 3}];
  fn[t_] := Sum[r[[l + 4]] * dwv2[t + l], {l, -3, 3}];
  Plot[{poly, fn[t]}, {t, low, high}, Filling → Axis,
   PlotStyle → {Red, Blue}]
 ]
```

In[154]:=
```
sc2[t + 1, -4, 5]
```

Out[154]=



Approximating with D8

the D8 filter coefficients

In[114]:=
```
wv4 = WaveletFilterCoefficients[DaubechiesWavelet[4],
   "PrimalLowpass" ,
   WorkingPrecision → 5];
wv4 // MatrixForm
```

Out[115]//MatrixForm=

0	0.16290
1	0.50547
2	0.44610
3	-0.019788
4	-0.13225
5	0.021808
6	0.023252
7	-0.0074935

the Fourier series of the D8 filter

In[116]:=
```
m04[s_] :=
 Sum[wv4[[j + 1, 2]] Exp[-2 Pi I s j], {j, 0, 7}]
```

vanishing moments of D8

In[117]:=
```
m04[s] /. s → 1/2
```

Out[117]= $0. \times 10^{-5}$

In[118]:=
```
D[m04[s], s] /. s → 1/2
```

Out[118]= $0. \times 10^{-4}\, i$

In[119]:=
```
D[m04[s], {s, 2}] /. s → 1/2
```

Out[119]= $0. \times 10^{-3}$

In[120]:=
```
D[m04[s], {s, 3}] /. s → 1/2
```

Out[120]= $0. \times 10^{-2}\, i$

In[121]:=
```
D[m04[s], {s, 4}] /. s → 1/2
```

Out[121]= 1.383×10^{4}

checking orthogonality

In[122]:=
```
Assuming[s ∈ Reals, FullSimplify[
  Expand[m04[s] * Conjugate[m04[s]] +
    + Expand[m04[s + 1 / 2] * Conjugate[m04[s + 1 / 2]]]]]]
```

Out[122]=
```
1.000 + 0. × 10^-5 Cos[2 π s] + 0. × 10^-5 Cos[4 π s] + 0. × 10^-6 Cos[6 π s] + 0. × 10^-6 Cos[8 π s] +
 0. × 10^-6 Cos[10 π s] + 0. × 10^-7 Cos[12 π s] + 0. × 10^-8 Cos[14 π s] + 0. × 10^-5 ⅈ Sin[2 π s] +
 0. × 10^-5 ⅈ Sin[4 π s] + 0. × 10^-6 ⅈ Sin[6 π s] + 0. × 10^-6 ⅈ Sin[8 π s] +
 0. × 10^-6 ⅈ Sin[10 π s] + 0. × 10^-7 ⅈ Sin[12 π s] + 0. × 10^-8 ⅈ Sin[14 π s]
```

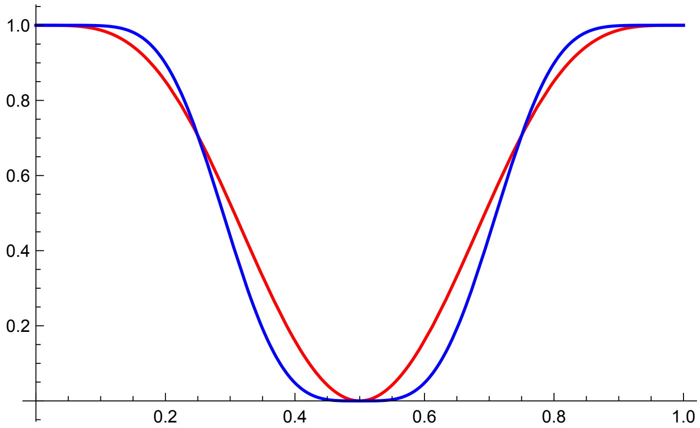
In[123]:=
```
Chop[%, 10^(-4)]
```

Out[123]=
```
1.000
```

displaying the filter characteristics of D4 and D8

In[124]:=
```
Plot[{Abs[m02[s]], Abs[m04[s]]}, {s, 0, 1}, PlotStyle → {Red, Blue}]
```

Out[124]=



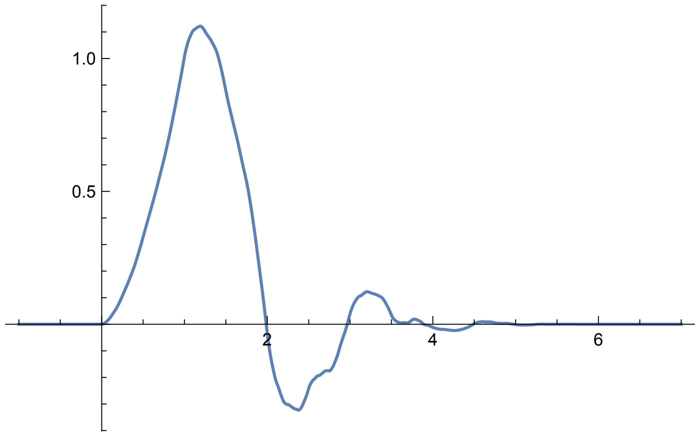
the D8 scaling function from the *Mathematica* system

In[492]:=
```
dwv4 = WaveletPhi[DaubechiesWavelet[4], MaxRecursion → 15]
```

Out[492]=
```
{ InterpolatingFunction[ Domain: {{0., 7.}} Output: scalar ][#1]   0 ≤ #1 ≤ 7
  0                                                                True        &
```

In[495]:=
```
Plot[dwv4[t], {t, -1, 7}, PlotRange → All]
```

Out[495]=



computing inner products numerically

In[126]:=
```
ρ4[poly_, l_] :=
 NIntegrate[(poly /. t → t - l) * dwv4[t], {t, 0, 10}, WorkingPrecision → 5]
```

In[127]:=
```
Table[{l, ρ4[t^2 + 1, l]}, {l, -3, 3}] // MatrixForm
```

Out[127]//MatrixForm=

-3	17.042
-2	10.032
-1	5.0243
0	2.0114
1	1.0004
2	1.9892
3	4.9701

the approximation of a polynomial function if degree < 4 (discrete display)

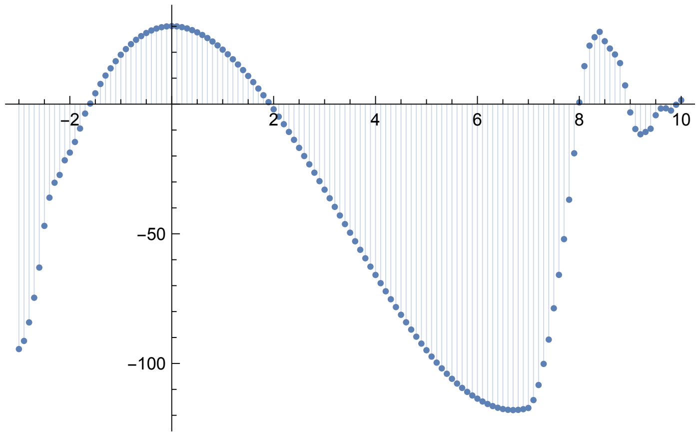
In[128]:=
```
sd4[poly_, low_, high_, step_] :=
 Module[{r},
  r = Table[ρ4[poly, l], {l, -6, 6}];
  Table[{t, Sum[r[[l + 7]] * dwv4[t + l], {l, -6, 6}]}, {t, low, high, step}]
 ]
```

In[129]:=
```
sd4[t^3 - 10 t^2 + 10, 1, 2, 0.1]
```

Out[129]=
```
{{1., 0.998907}, {1.1, -0.771625}, {1.2, -2.67529},
 {1.3, -4.70644}, {1.4, -6.86034}, {1.5, -9.12988}, {1.6, -11.5097},
 {1.7, -13.9937}, {1.8, -16.5759}, {1.9, -19.2499}, {2., -22.0099}}
```

In[130]:=
```
ListPlot[sd4[t^3 - 10 t^2 + 30, -3, 10, 0.1], Filling → Axis]
```

Out[130]=

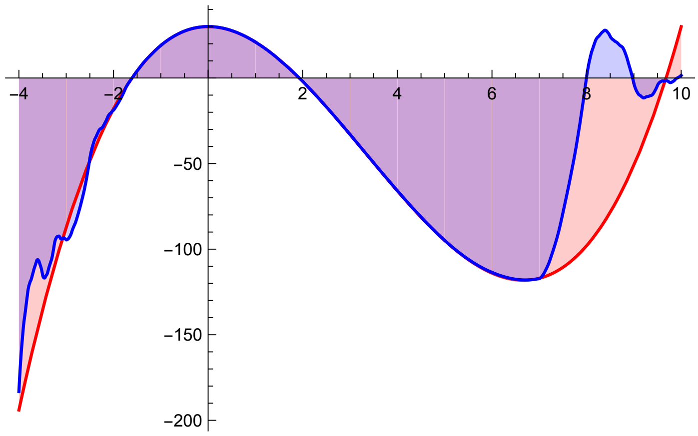


approximation of a polynomial function if degree < 4 (continuous display)

In[151]:=
```
sc4[poly_, low_, high_] :=
 Module[{r, fn},
  r = Table[ρ4[poly, l], {l, -6, 6}];
  fn[t_] := Sum[r[[l + 7]] * dwv4[t + l], {l, -6, 6}];
  Plot[{poly, fn[t]}, {t, low, high},
   PlotRange → All,
   Filling → Axis,
   PlotStyle → {Red, Blue}]
 ]
```

In[152]:=
```
sc4[t^3 - 10 t^2 + 30, -4, 10]
```

Out[152]=



Approximating with D10

In[133]:=
```
dwv5 = WaveletPhi[DaubechiesWavelet[5], MaxRecursion → 15]
```

Out[133]=
```
{ InterpolatingFunction[ Domain: {{0., 9.}} Output: scalar ][#1]   0 ≤ #1 ≤ 9   &
  0                                                                True
```

computing the inner products

In[134]:=
```
ρ[poly_, k_, wvphi_, N_, prec_] :=
 NIntegrate[(poly /. t → t - k) * wvphi[t], {t, 0, 2 N - 1}, AccuracyGoal → prec]
```

In[135]:=
```
Table[{k, ρ[t^3 - t^2, k, dwv5, 5, 4]}, {k, 0, 8}] // MatrixForm
```

Out[135]//MatrixForm=

0	0.154842
1	-0.151889
2	-1.29513
3	-9.27494
4	-30.0913
5	-69.7442
6	-134.234
7	-229.56
8	-361.722

approximation of a polynomial function if degree < 4 (continuous display)

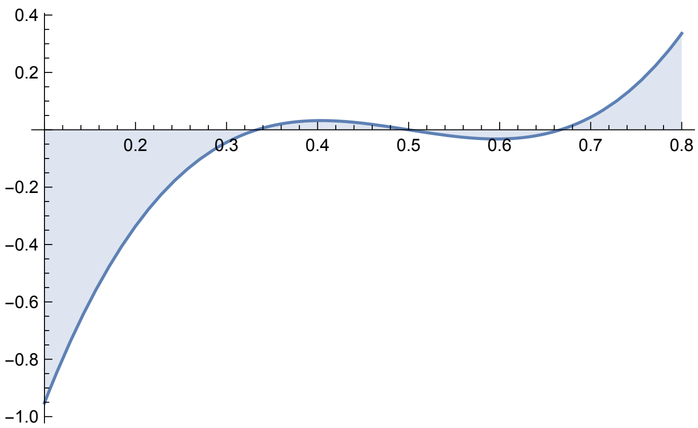
In[142]:=
```
sc5[poly_, wvphi_, N_, prec_, left_, right_] := Module[{r},
  r = Table[ρ[poly, k, wvphi, N, prec], {k, 0, 2 N - 2}];
  fn[t_] := Sum[r[[k + 1]] * wvphi[t + k], {k, 0, 2 N - 2}];
  Plot[{poly, fn[t]}, {t, left, right},
   PlotRange → All,
   Filling → Axis,
   PlotStyle → {Red, Blue}]
 ]
```

In[137]:=
```
pol = Expand[18 * (t - 1 / 3) * (t - 1 / 2) * (t - 2 / 3)]
```

Out[137]= $-2 + 13\,t - 27\,t^2 + 18\,t^3$

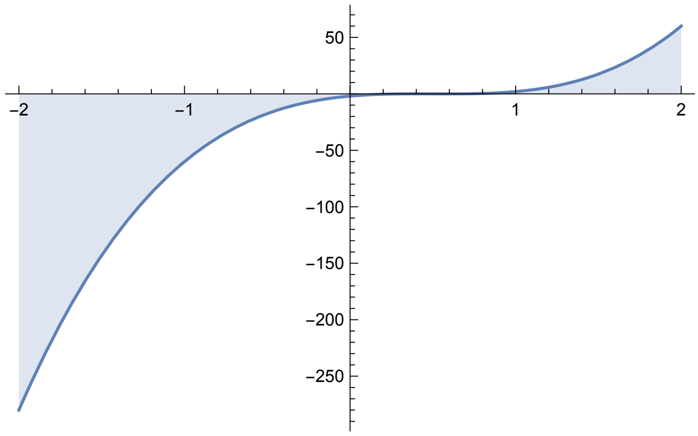
In[146]:=
```
Plot[pol, {t, 0.1, 0.8},
 PlotRange → All,
 Filling → Axis]
```

Out[146]=



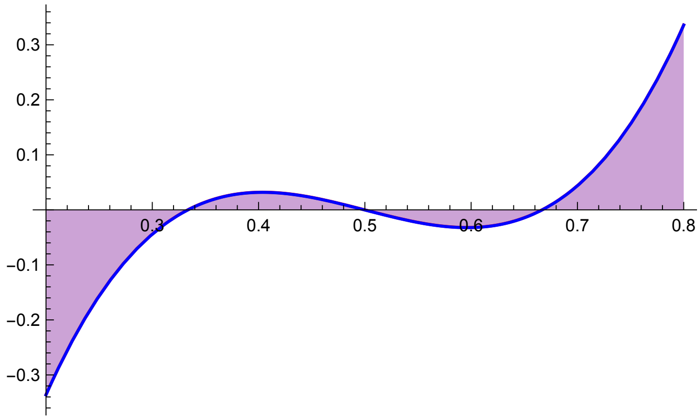
In[529]:=
```
Plot[pol, {t, -2, 2},
 PlotRange → All,
 Filling → Axis]
```

Out[529]=



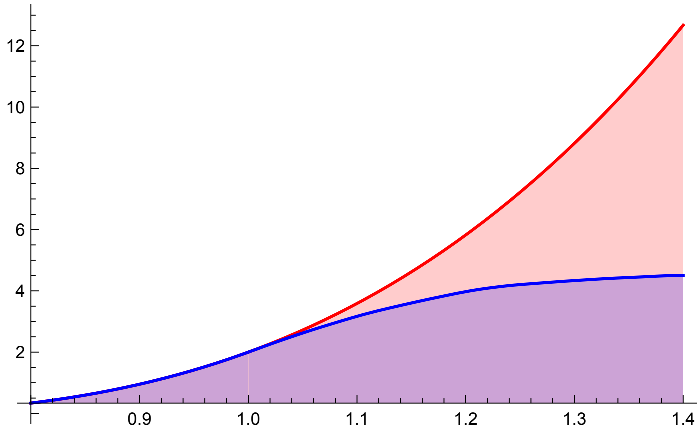
In[143]:=
```
sc5[pol, dwv5, 5, 3, 0.2, 0.8]
```

Out[143]=



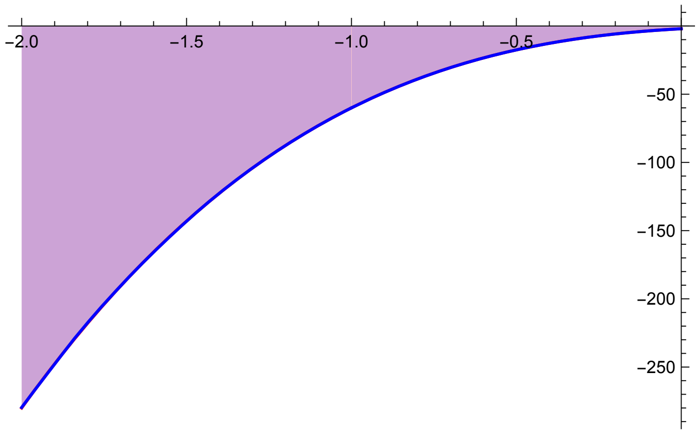
In[144]:=
```
sc5[pol, dwv5, 5, 3, 0.8, 1.4]
```

Out[144]=



In[528]:= sc5[pol, dwv5, 5, 3, -2, 0]

Out[528]=



In[145]:= sc5[pol, dwv5, 5, 3, -8, -2]

Out[145]=