Gruppe 1
Freitag 13:15-14:45, B-Woche
(Dr. B. Schulze)

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

Numerik

Serie 10

19. Wir berechnen den Wert des Interpolationspolynoms an der Stelle $t = -3$ mit Hilfe des Aitken/Neville Verfahrens:

$$\begin{aligned} P_{0,1} &= 25 - \frac{-3+2}{-1+3}(25-10) \\ &= 40 \end{aligned}$$

$$\begin{aligned} P_{1,1} &= P_{1,0} - \frac{t-t_1}{t_i-t_1}(P_{1,0} - P_{2,0}) \\ &= 10 - \frac{-3+1}{1}(10-3) \\ &= 24 \end{aligned}$$

$$\begin{aligned} P_{2,1} &= 3 - \frac{-3-0}{1-0}(3-4) \\ &= 0 \end{aligned}$$

$$\begin{aligned} P_{3,1} &= 4 - \frac{-3-1}{2-1}(4+11) \\ &= 64 \end{aligned}$$

$$\begin{aligned} P_{0,2} &= 40 - \frac{-3+2}{0+2}(40-24) \\ &= 48 \end{aligned}$$

$$\begin{aligned} P_{1,2} &= 24 - \frac{-3+1}{1+1}(24-0) \\ &= 48 \end{aligned}$$

$$\begin{aligned} P_{2,2} &= 0 - \frac{-3+0}{2}(0-64) \\ &= -96 \end{aligned}$$

$$\begin{aligned} P_{0,3} &= 48 - \frac{-3+2}{1+2}(48-48) \\ &= 48 \end{aligned}$$

$$\begin{aligned} P_{1,3} &= 48 - \frac{-3+1}{2+1}(48+96) \\ &= 144 \end{aligned}$$

$$\begin{aligned} P_{0,4} &= 48 - \frac{-3+2}{2+2}(48-144) \\ &= 24 \end{aligned}$$

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

20. a) Hier die Ergebnisse unseres Algorithmus:

x	$f(x)$	$p(x)$	Differenz
-5.00	3.846154e-02	3.846154e-02	0.000000e+00
-4.75	4.244032e-02	-1.960417e+01	1.964661e+01
-4.50	4.705882e-02	4.705882e-02	0.000000e+00
-4.25	5.245902e-02	1.805097e+00	-1.752638e+00
-4.00	5.882353e-02	5.882353e-02	0.000000e+00
-3.75	6.639004e-02	-2.113576e-01	2.777477e-01
-3.50	7.547170e-02	7.547170e-02	0.000000e+00
-3.25	8.648649e-02	1.524268e-01	-6.594036e-02
-3.00	1.000000e-01	1.000000e-01	0.000000e+00
-2.75	1.167883e-01	9.517489e-02	2.161343e-02
-2.50	1.379310e-01	1.379310e-01	-1.355253e-20
-2.25	1.649485e-01	1.742570e-01	-9.308525e-03
-2.00	2.000000e-01	2.000000e-01	1.355253e-20
-1.75	2.461538e-01	2.410968e-01	5.057043e-03
-1.50	3.076923e-01	3.076923e-01	0.000000e+00
-1.25	3.902439e-01	3.935025e-01	-3.258630e-03
-1.00	5.000000e-01	5.000000e-01	0.000000e+00
-0.75	6.400000e-01	6.379309e-01	2.069061e-03
-0.50	8.000000e-01	8.000000e-01	-1.084202e-19
-0.25	9.411765e-01	9.414267e-01	-2.502684e-04
0.00	1.000000e+00	1.000000e+00	0.000000e+00
0.25	9.411765e-01	9.436660e-01	-2.489512e-03
0.50	8.000000e-01	8.000000e-01	5.421011e-20
0.75	6.400000e-01	6.351648e-01	4.835186e-03
1.00	5.000000e-01	5.000000e-01	8.131516e-19
1.25	3.902439e-01	3.977439e-01	-7.500021e-03
1.50	3.076923e-01	3.076923e-01	5.285486e-18
1.75	2.461538e-01	2.329403e-01	1.321356e-02
2.00	2.000000e-01	2.000000e-01	1.627659e-17
2.25	1.649485e-01	1.942775e-01	-2.932908e-02
2.50	1.379310e-01	1.379310e-01	1.031754e-16
2.75	1.167883e-01	3.066423e-02	8.612409e-02
3.00	1.000000e-01	1.000000e-01	4.112921e-16
3.25	8.648649e-02	4.381169e-01	-3.516305e-01
3.50	7.547170e-02	7.547170e-02	1.553790e-15
3.75	6.639004e-02	-2.096912e+00	2.163302e+00
4.00	5.882353e-02	5.882353e-02	-1.064971e-15
4.25	5.245902e-02	2.380323e+01	-2.375078e+01

Wie man deutlich sieht wird die Newton-Interpolation umso schlechter, je näher man an der Intervallmitte und den Grenzen des Stützwertintervalls interpoliert.

b) Hier unser Quellcode:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_VEC 20
typedef struct {
  long double v[MAX_VEC];
} vector_t;



long double DividedDiff(vector_t* x,vector_t* y,int us,int os) {
  // Divifdierte Differenzen von unterer Schranke us bis oberer Schranke os
  // printf("us=%d,os=%d\n",us,os);
  if(us == os )
    return y->v[us];
  else
    return (DividedDiff(x,y,us,os-1) - DividedDiff(x,y,us+1,os)) /
      (x->v[us] - x->v[os]);
}

// t sind die 20 x-Werte und f die 20 y-Werte der Funktion f.
// x ist der Wert fuer den Interpoliert werden soll!
long double NewtonPol (vector_t* t,vector_t* f, long double x) {
  static vector_t c;
  static vector_t *my_t = NULL;
  static vector_t *my_f = NULL;
  long double pn;
  int i;

  if (my_t != t && my_f != f) {
    // Initialisieren von c
    // durch Berechnen der Dividierten Differenzen.
    my_t = t;
    my_f = f;
    for (i = 0; i < MAX_VEC; i++)
      c.v[i] = DividedDiff(t,f,0,i);

  }
  pn = c.v[MAX_VEC-1];
  for(i = MAX_VEC - 2; i >= 0; i-- )
    pn = pn*(x - my_t->v[i]) + c.v[i];

  return pn;

}
```

Claudia Heinich, 8966108
Sven Pohl, 8968075
Arne Brutschy, 8964813

```
long double BerechneF(long double x) {
  return 1/(1+ pow(x, 2.0));
}

int main() {
  vector_t t,f;
  int i;
  long double pol,exakt;
  long double j;

  // Initialisiere Werte:
  for(i = 0; i < MAX_VEC; i++) {
    t.v[i] = -5 + 10*(long double)i/MAX_VEC;
    f.v[i] = BerechneF(t.v[i]);
  }

  // Interpoliere die Werte mittels des brechneten Polynoms fuer
  // die stuetzwerte sowie alle viertel bis zum naechsten Stuezwert
  for (j = t.v[0];j < t.v[MAX_VEC-1]; j += 0.25 ) {
    pol = NewtonPol (&t,&f,j);
    exakt = BerechneF(j);
    printf("          %.2Lf & %Le & %Le & %Le \\\\\\n",j,exakt,pol,exakt-pol);
  }

  return EXIT_SUCCESS;
}
```