# Anhang B

# Mathematica-Programme

## B.1 MATHEMATICA-Programm zur Maximumsuche mit algebraischer Bisektion

```
(* bise.m  Algebraic Bisection for n variables. Version: 06.09.89, 17:59 *)
(* Features:
   + own CoefficientList
   + Divide into 4 parts
   x Careful search strategie (to be implemented)
   + arbitrary dimensions
   - estimation: moredim: neglecting negative coefficients
   + estimation: onedim: summing up in a smart way
   + no Breaks[ ] in (More)CleanMaxList
*)
Unprotect[Power]
0^0   := 1
0. ^0 := 1
Protect[Power]

Sub[f_,x_,y_] := If[x=={},f,Sub[f,Rest[x],Rest[y]]/.x[[1]]->y[[1]]]

(* summ[i_,j_] := Block[{ee},
                      ee=e[[i,j]];
                      If[ee>0,e[[i,j]]=0;ee,0] +
                      If[i<m,summ[i+1,j],0] +
                      If[j<n,summ[i,j+1],0]
                    ] *)

Major[f_,x_] := Block[{fe,co,cl,i,su=0},
                      If[CheckMaxDim==1,
                         cl=CoefficientList[f,x[[1]]];
                         For[i=Length[cl], i>=1, i--,
                             su=su+cl[[i]]; If[su<0,su=0]];
                         su,
                         fe = Expand[f];
                         If[TrueQ[Head[fe]==Plus],
                            Sum[co=fe[[i]]; If[Head[co]==Times,co=fe[[i,1]]];
                                If[co<0,0,co,1], {i,1,Length[fe]}],
                            co=If[Head[fe]==Times,fe[[1]],fe,fe];
                            If[co<0,0,co,1]
```

```
                          ]
                       ]
                    ]
Major::usage = " Major[f,{x1,x2,...}] determins the value
                 of a majorant of f at the point (x1=1, x2=1, ...)"

Surepos[g_,x_,a_,b_] := Block[{g1,Sya},
                               Sya=Array[Sy,Dimensions[x]];
                               g1 = Sub[g,x,a];
                               Major[g1-Sub[g,x,(Sya(b-a)+a)],Sya] < g1
                              ]

Bounds[g_,x_,a_,b_] := Block[{g1,Bya},
   (* lower and upper bounds of g in interval  *)
                             g1 = Sub[g,x,a];
                             Bya=Array[By,Dimensions[x]];
                             {g1 - Major[g1-Sub[g,x,(Bya(b-a)+a)], Bya],
                              g1 + Major[Sub[g,x,(Bya(b-a)+a)]-g1, Bya]}
                            ]


CleanMaxList[T_] := Block[{i,j,flag,hl={}},
                      For[i=1, i<=Length[T], i++,
                          flag=True;
                          For[j=1, j<=Length[T] && TrueQ[flag], j++,
                              flag = T[[j,1,1]]<T[[i,1,2]]
                             ];
                          If[flag==True,
                             AppendTo[hl,T[[i]]],
                             Print["CleanMaxList has hit one !"],
                             Print["flag in CML :",FullForm[flag]];
                             AppendTo[schrott,T[[i]]]
                            ]
                         ];
                      hl
                     ]

MoreCleanMaxList[T_] :=
     Block[{i,j,flag,hl={}},
            For[ i=1, i<=Length[T], i++,
                 flag=True;
                 For[ j=1, j<=Length[Checkx] && TrueQ[flag], j++,
                      If[T[[i,3,j]]!=Checkb[[j]],
                         flag = !Surepos[CheckD[[j]],Checkx,T[[i,2]],T[[i,3]]]
                        ]
                    ];
                 For[ j=1, j<=Length[Checkx] && TrueQ[flag], j++,
                      If[T[[i,2,j]]!=Checka[[j]],
                         flag = !Surepos[-CheckD[[j]],Checkx,T[[i,2]],T[[i,3]]]
                        ]
                    ];
                 If[flag==True,
                    AppendTo[hl,T[[i]]],
                    Print["MoreCleanMaxList has hit one !"],
                    Print["flag in MCML:",FullForm[flag]];
                    AppendTo[schrott,T[[i]]]
                   ]
               ];
            hl
          ]

DevideInt[a_,b_,i_] :=
  (* Devide 1 Interval in respect to variable [i] *)
            Block[{m1,m2,m3,t1b,t2a,t2b,t3a,t3b,t4a},
                  (* Print["DevideInt[",a,",",b,",i=",i]; *)
```

```
                  m2=(a[[i]]+b[[i]])/2;
                  m1=(a[[i]]+m2)/2;
                  m3=(m2+b[[i]])/2;
                  t1b=b; t1b[[i]]=m1;
                  t2a=a; t2a[[i]]=m1;
                  t2b=b; t2b[[i]]=m2;
                  t3a=a; t3a[[i]]=m2;
                  t3b=b; t3b[[i]]=m3;
                  t4a=a; t4a[[i]]=m3;
                  {{Bounds[Checkg,Checkx,a,t1b],a,t1b},
                   {Bounds[Checkg,Checkx,t2a,t2b],t2a,t2b},
                   {Bounds[Checkg,Checkx,t3a,t3b],t3a,t3b},
                   {Bounds[Checkg,Checkx,t4a,b],t4a,b}}
                 ]

CheckMaxList[T_,n_] :=
           Block[ {i,d,hl,nl,p},
                  If[n==0,
                     T,
                     hl=T;
                     For[ d=1, d<=CheckMaxDim, d++,
                          Print["Dividing variable ",d];
                          nl={};
                          For[ i=1, i<=Length[hl], i++,
                               nl=Join[nl,DevideInt[hl[[i,2]],hl[[i,3]],d]]
                             ];
                          Print[Length[nl]," Intervals. Starting Clean-Up."];
                          hl=MoreCleanMaxList[CleanMaxList[nl]]
                        ];
                     CheckMaxList[hl,n-1]
                    ]
                ]

AbsMax::usage =
"AbsMax[g,{x1,x2,...},{a1,a2,...},{b1,b2,...},n,sl]
 gives a list of intervals in one of that the absolute
 maximum of f in a1<x1<b1, a2<x2<b2, ... lies.
 n is the desired number of recursive subdivisions
 The list sl can be inputted as a starting list."

AbsMax[g_,x_,a_,b_,n_:1,sl_:{}] :=
            Block[ {i},
                    schrott = {};
                    Checka = a;
                    Checkb = b;
                    Checkg = g;
                    Checkx = x;
                    CheckMaxDim=Length[x];
                    CheckD = Table[D[g,x[[i]]],{i,Length[x]}];
                    If[sl=={},
                       CheckMaxList[{{{low,high},a,b}},n],
                       CheckMaxList[sl,n]]
                 ]

(* prec and mach are useful routines for demonstration *)

prec[T_] := N[T[[1,3]]-T[[1,2]]]

mach[f_,var_,a_,b_,n_:5,width_:132] :=
              Block[{i,j},
                    ResetMedium[PageWidth->132];
                    intlist={};
                    For[ i=1; tl={}, i<=n, i++,
                        Print[" "];
                        AppendTo[tl,Reverse[
```

```
                         Timing[intlist=AbsMax[f,var,a,b,1,intlist];i]]];
                      tl >> bise.tl;
                      Print[tl[[i]]," Precision=",prec[intlist]];
                      Print[Length[intlist],
                            " Intervals remaining:
 {{f1,f2},{a1,a2,...},{b1,b2,....}"];
                      For[j=1, j<=Length[intlist], j++,
 Print[N[intlist[[j]],16]]]
                     ];
                  If[$Display=={},$Display="bise.ps"];
                  ListPlot[tl/.Second->1,
                           PlotJoined->True,
                           AxesLabel->{"Iterations","CPUs per Iteration"}
                          ]
                 ]

(* Example Cube *)
cube = 100(x2-x1^3)^2 + (1-x1)^2
Plot3D[-cube,{x1,-1,2},{x2,-1,2},PlotLabel->"Cube"]
machcube[n_:5] := mach[-cube,{x1,x2},{-1,-1},{2,2},n,80]

(* Example Beale *)
beale = (3/2 - x1(1-x2))^2 + (9/4 - x1(1-x2^2))^2 + (21/8 - x1(1-x2^3))^2
Plot3D[-beale,{x1,-1,2},{x2,-1,2},PlotLabel->"Beale"]
machbeale[n_:5] := mach[-beale,{x1,x2},{-1,-1},{2,2},n,80]

(* Example Wood *)
wood = 100(x2-x1^2)^2 + (1-x1)^2 + 90(x4-x3^2)^2 + (1-x3)^2 +
       (101/10)((x2-1)^2+(x4-1)^2) + (198/10)(x2-1)(x4-1)
machwood[n_:5] := mach[-wood,{x1,x2,x3,x4},{-1,-1,-1,-1},{2,2,2,2},n,80]

(* Example n dim. *)
machn[n_] := Block[{xx,aa},
                 xx = Array[x,{n}];
                 aa = Table[-1,{n}];
                 spin = Expand[Sum[-(x[i]-1/3)^2,{i,1,n}]];
                 Print[spin];
                 mach[spin,xx,aa,-aa,10,132]
                ]
```

## B.2 MATHEMATICA-Programm zur Maximumsuche durch Potenzierung und Fit

```
Unprotect[Times]
0. _ := 0
Protect[Times]

MaxFitWeightPower = 2

MaxFit1[fp_,{x_,a_,b_}] := Block[{A,B,dA,a4,a3,a2,a1,a0,fpn,alpha,beta,gamma},
                            fpn = fp^MaxFitWeightPower;
                            Array[A,{3,3}];
                            Array[B,{3}];
                            Print["Calculating a4"];
                            a4 = Integrate[Expand[fpn x^4],{x,a,b}];
                            Print["Calculating a3"];
                            a3 = Integrate[Expand[fpn x^3],{x,a,b}];
                            Print["Calculating a2"];
                            a2 = Integrate[Expand[fpn x^2],{x,a,b}];
                            Print["Calculating a1"];
                            a1 = Integrate[Expand[fpn x],{x,a,b}];
                            Print["Calculating a0"];
                            a0 = Integrate[Expand[fpn], {x,a,b}];
                            A = {{a4,a3,a2},{a3,a2,a1},{a2,a1,a0}};
                            dA = Det[A];
                            Print[StringForm["Determinant=``",dA]];
                            B = {Print["Calculating b2"];
                                 Integrate[Expand[fpn fp x^2],{x,a,b}],
                                 Print["Calculating b1"];
                                 Integrate[Expand[fpn fp x],{x,a,b}],
                                 Print["Calculating b0"];
                                 Integrate[Expand[fpn fp],{x,a,b}]};
                            alpha = Det[{B,{a3,a2,a1},{a2,a1,a0}}]/dA;
                            beta  = Det[{{a4,a3,a2},B,{a2,a1,a0}}]/dA;
                            gamma = Det[{{a4,a3,a2},{a3,a2,a1},B}]/dA;
                           (* Plot[{fp,alpha x^2+beta x+gamma},{x,a,b}]; *)
                            {alpha,beta,gamma}
                           ]

MaxFind1::usage = "MaxFind1[f,{x,a,b},mfp] finds the absoulute Maximum of a
 non-negative
                  function f[x] in the interval a<x<b.
                  mfp is the used power, can be omitted, default is 8."
MaxFind1[f_,{x_,a_,b_},mfp_:8] :=
                           Block[{k,d,w},
                            Print[StringForm["Interval ``,
 MaxFitPower=``",{a,b},mfp]];
                            MaxFitResult = MaxFit1[f^mfp,{x,a,b}];
                            ka=MaxFitResult[[1]];
                            kb=MaxFitResult[[2]];
                            kc=MaxFitResult[[3]];
                            k =kb/(-2 ka);
                            d =kb^2 - 4 ka kc;
                            fehlb = Sqrt[d]/(2 ka);
                            k
                           ]

MaxFit2[fp_,{x1_,a1_,b1_},{x2_,a2_,b2_}] :=
         Block[{A,B,a40,a22,a04,a30,a03,a21,a12,a20,a02,a11,a10,a01,a00,fpn,r,
                c20,c02,c10,c01,c00},
                fpn = fp^MaxFitWeightPower;
                Array[A,{5,5}];
```

```
                Array[B,{5}];
                a40 = Integrate[Expand[fpn x1^4],{x1,a1,b1},{x2,a2,b2}];
                a04 = Integrate[Expand[fpn x2^4],{x1,a1,b1},{x2,a2,b2}];
                a22 = Integrate[Expand[fpn x1^2 x2^2],{x1,a1,b1},{x2,a2,b2}];
                a30 = Integrate[Expand[fpn x1^3],{x1,a1,b1},{x2,a2,b2}];
                a03 = Integrate[Expand[fpn x2^3],{x1,a1,b1},{x2,a2,b2}];
                a21 = Integrate[Expand[fpn x1^2 x2],{x1,a1,b1},{x2,a2,b2}];
                a12 = Integrate[Expand[fpn x1 x2^2],{x1,a1,b1},{x2,a2,b2}];
                a20 = Integrate[Expand[fpn x1^2],{x1,a1,b1},{x2,a2,b2}];
                a02 = Integrate[Expand[fpn x2^2],{x1,a1,b1},{x2,a2,b2}];
                a11 = Integrate[Expand[fpn x1 x2],{x1,a1,b1},{x2,a2,b2}];
                a10 = Integrate[Expand[fpn x1],{x1,a1,b1},{x2,a2,b2}];
                a01 = Integrate[Expand[fpn x2],{x1,a1,b1},{x2,a2,b2}];
                a00 = Integrate[Expand[fpn],{x1,a1,b1},{x2,a2,b2}];

                A = {{a40,a22,a30,a21,a20},
                     {a22,a04,a12,a03,a02},
                     {a30,a12,a20,a11,a10},
                     {a21,a03,a11,a02,a01},
                     {a20,a02,a10,a01,a00}};
                Print[TableForm[A]];
                Print[StringForm["Determinant=``",Det[A]]];
                B = {Integrate[Expand[fpn fp x1^2],{x1,a1,b1},{x2,a2,b2}],
                     Integrate[Expand[fpn fp x2^2],{x1,a1,b1},{x2,a2,b2}],
                     Integrate[Expand[fpn fp x1],{x1,a1,b1},{x2,a2,b2}],
                     Integrate[Expand[fpn fp x2],{x1,a1,b1},{x2,a2,b2}],
                     Integrate[Expand[fpn fp],{x1,a1,b1},{x2,a2,b2}]};
                r = Solve[A.{c20,c02,c10,c01,c00}==B];
                Flatten[{c20,c02,c10,c01,c00}/.r]
               ]

MaxFind2[f_,{x1_,a1_,b1_},{x2_,a2_,b2_},mfp_:8] :=
          Block[{k},
                 MaxFitResult = MaxFit2[f^mfp,{x1,a1,b1},{x2,a2,b2}];
                 k20=MaxFitResult[[1]];
                 k02=MaxFitResult[[2]];
                 k10=MaxFitResult[[3]];
                 k01=MaxFitResult[[4]];
                 k00=MaxFitResult[[5]];
                 {-k10/(2 k20), -k01/(2 k02)}
               ]

machmfp = 10

test = 10 - (x1-0.2)^2 - (x2-0.4)^2

mach2 := MaxFind2[test,{x1,-1,1},{x2,-1,1},machmfp]

fit2 := k20 x1^2 + k02 x2^2 + k10 x1 + k01 x2 + k00

fit1 := ka x^2 + kb x + kc
```