ДОДАТОК А

Код програми

Input

```
Clear["Global`*"]
λ = 1;
μ = 1;
BoundProbType = {1, 2};
k1 = 0;
k2 = 0;
cons1 = Switch[BoundProbType[[1]],
   1, u[0] = 0,
   2, u'[0] = 0,
   3, u'[0] - k1 u[0] == 0,
   _, Return[-1]
  ];
cons2 = Switch[BoundProbType[[2]],
   1, u[1] = 0,
   2, u'[1] = 0,
   3, u'[1] + k2 u[1] = 0,
   _, Return[-1]
  ];
flagNotSaveToFile = 0;
OutputFolder = "E:\\from_d\\ucheb\\dip\\Output";
CurrentProblemFolder = "1,2; h=0.05; e=10^-5";
OutputFolder = FileNameJoin[{OutputFolder, CurrentProblemFolder}];
eps = 10^-5;
h = 0.05;
MAXITER = 300;
hp = 0.1;
pbeg = 0.1; pend = 0.9;
hq = 0.1;
qbeg = 0.1; qend = 0.9;
```

Function definitions

AlphaBethaCalculations

```
SetDirectory[OutputFolder];
FuncMinMax11[p_] := {2^(1-p) Beta[p + 1, p + 2],
   2^-p (Beta[p + 1, p + 2] - 2 Beta[1/2, p + 1, p + 2]) + 2^(-3 p) Beta[1/2, p + 1]};
FuncMinMax12[p_] := {2^(p+1) Beta[1/2, p + 1, p + 1], 2^(p+3) Beta[1/2, p + 2, p + 1]};
FuncMinMax21[p_] := {2^(-p-1) Beta[1/2, p + 1], 2^-p (-1/(p + 1) + Beta[1/2, p + 1])};
Func13[x_, k1_, k2_, p_] :=
 1/((k2 + 2) - (k2 + 1) x) 2^(p-1) (k2 + 1)^p
  ((1 + k2 (1 - x))/x NIntegrate[s^(p+1) ((k2 + 2) - (k2 + 1) s)^p, {s, 0, x}] + NIntegrate[s^p ((k2 + 2) - (k2 + 1) s)^p (1 + k2 (1 - s)), {s, x, 1}]);
Func23[x_, k1_, k2_, p_] :=
 1/(2 + k2 (1 - x) (1 + x)) 2^(p-1) k2^p
  ((1 + k2 (1 - x)) NIntegrate[(2 + k2 (1 - s) (1 + s))^p, {s, 0, x}] + NIntegrate[(1 + k2 (1 - s)) (2 + k2 (1 - s) (1 + s))^p, {s, x, 1}]);
Func31[x_, k1_, k2_, p_] :=
 1/(1 + (k1 + 1) x) 2^(p-1) (k1 + 1)^p
  (NIntegrate[(k1 s + 1) (1 - s)^p (1 + (k1 + 1) s)^p, {s, 0, x}] + (k1 x + 1)/(1 - x) NIntegrate[(1 - s)^(p+1) (1 + (k1 + 1) s)^p, {s, x, 1}]);
Func32[x_, k1_, k2_, p_] :=
 1/(2 + k1 x (2 - x)) 2^(p-1) k1^p (NIntegrate[(k1 s + 1) (2 + k1 s (2 - s))^p, {s, 0, x}] + (k1 x + 1) NIntegrate[(2 + k1 s (2 - s))^p, {s, x, 1}]);
Func33[x_, k1_, k2_, p_] := (1/((2 + k2) + k1 (2 + k2) x - (k1 + k2 + k1 k2) x^2) 2^(p-1) (k1 + k2 + k1 k2)^p)
  ((1 + k2 (1 - x)) NIntegrate[(k1 s + 1) ((2 + k2) + k1 (2 + k2) s - (k1 + k2 + k1 k2) s^2)^p, {s, 0, x}] +
   (k1 x + 1) NIntegrate[(1 + k2 (1 - s)) ((2 + k2) + k1 (2 + k2) s - (k1 + k2 + k1 k2) s^2)^p, {s, x, 1}]);
```

```
(*Define AlBetCalc*)
AlBetCalc[{p_, q_, λ_, μ_}, OutputAddress_] := Module[{},
  FuncMinMax[pp_] = {-1, -1};
  Func[xx_, pp_] = -1;
  m1 = -1; M1 = -1; m2 = -1; M2 = -1;
  If[BoundProbType[[1]] ≠ 3 && BoundProbType[[2]] ≠ 3,
   FuncMinMax[pp_] = Switch[BoundProbType,
     {1, 1}, FuncMinMax11[pp],
     {1, 2}, FuncMinMax12[pp],
     {2, 1}, FuncMinMax21[pp],
     _, {-2, -2}
    ];
   {m1, M1} = FuncMinMax[p];
   {M2, m2} = FuncMinMax[-q];
   ,

  Func[xx_, pp_] = Switch[BoundProbType,
    {1, 3}, Func13[xx, k1, k2, pp] // Hold,
    {2, 3}, Func23[xx, k1, k2, pp] // Hold,
    {3, 1}, Func31[xx, k1, k2, pp] // Hold,
    {3, 2}, Func32[xx, k1, k2, pp] // Hold,
    {3, 3}, Func33[xx, k1, k2, pp] // Hold,
    _, {-2, -2}
   ];
  m1 = Minimize[{ReleaseHold@Func[x, p], 0 ≤ x ≤ 1}, x][[1]] // Quiet;
  M1 = Maximize[{ReleaseHold@Func[x, p], 0 ≤ x ≤ 1}, x][[1]] // Quiet;
  m2 = Minimize[{ReleaseHold@Func[x, -q], 0 ≤ x ≤ 1}, x][[1]] // Quiet;
  M2 = Maximize[{ReleaseHold@Func[x, -q], 0 ≤ x ≤ 1}, x][[1]] // Quiet;
 ];
 expr1 = (λ m1 α^p + μ m2 β^-q) - α;
 expr2 = (λ M1 β^p + μ M2 α^-q) - β;
 res = FindRoot[{expr1 == 0, expr2 == 0}, {{α, 1}, {β, 20}}, AccuracyGoal → Infinity, PrecisionGoal → 10, MaxIterations → 10^10] // Chop;
 alpha = res[[1, 2]];
 betha = res[[2, 2]];
 (*Output*)
  (*Output*)
  ABGraph = Show[
     RegionPlot[{expr1 ≥ 0 && expr2 ≤ 0}, {α, -0.1, 5}, {β, -0.1, 35}, AspectRatio → 1, PlotPoints → 50, FrameLabel → {"α", "β"}],
     ListPlot[{{α, β} /. res}], PlotStyle → Directive[PointSize[0.015], Black]]
    ] // Quiet;
  r1 = Row[{"m1 = ", m1, ", M1 = ", M1}];
  r2 = Row[{"m2 = ", m1, ", M2 = ", M2}];
  r3 = Row[{""}];
  r4 = Row[{"λm1 α^p+μm2 β^-q≥α"}];
  r5 = Row[{"λM1 β^p+μM2 β^-q≤β"}];
  r6 = Row[{""}];
  r7 = Row[{"α = ", alpha, ", β = ", betha}];
  ABOutputString = Column[{r1, r2, r3, r4, r5, r6, r7}];
  ABOutputArray = {alpha, betha};
  If[OutputAddress ≠ "",
   Save[OutputAddress, ABOutputArray];
   Save[OutputAddress, ABGraph];
   Save[OutputAddress, ABOutputString];
  ];
  ABOutputArray
 ](*end of module*)
```

ethod

```
G[x_, xi_] := GreenFunction[{-u''[xx], cons1, cons2}, u[xx], {xx, 0, 1}, xi] /. xx → x
xarr = Table[k, {k, 0, 1, h}];
Interp[arr_, x_] := Piecewise[
    Table[{arr[[i]] + (arr[[i+1]] - arr[[i]]) (x - xarr[[i]]) / h, xarr[[i]] ≤ x ≤ xarr[[i+1]]},
     {i, 1, Length[arr] - 1}]
   ] // Simplify;
NInt[G1_, G2_] := Module[{xi, xx},
  ukki = Table[
          If[i = 1 && ToString[cons1] = "u[0] == 0" || i = Length[xarr] && ToString[cons2] = "u[1] == 0", 0,
           NIntegrate[G1[xarr[[i]], xi], {xi, 0, xarr[[i]]}] +
            NIntegrate[G2[xarr[[i]], xi], {xi, xarr[[i]], 1}]
           ],
        {i, 1, Length[xarr]}];
  ukk = Function[{xx}, Interp[ukki, xx]]
 ]
```

```
T[vk_, wk_, x_] := Module[{xi, xx},
  F1[xx_, xi_] = G[xx, xi][[2]] f[vk, wk, xi];
  F2[xx_, xi_] = G[xx, xi][[1]] f[vk, wk, xi];
  NInt[F1, F2][x] // Chop]

G1[x_, xi_] = G[x, xi][[1]];
G2[x_, xi_] = G[x, xi][[2]];
u0[x_] = NInt[G1, G2][x];
v0[x_] := alp u0[x];
w0[x_] := bet u0[x];
nzero = eps / 2;
xmax = NMaximize[{u0[x], x ≥ nzero, x ≤ 1 - nzero}, x][[2, 1, 2]];
xmin = NMinimize[{u0[x], x ≥ nzero, x ≤ 1 - nzero}, x][[2, 1, 2]];

(*Define methods*)
MainMethod[{p_, q_, λ_, μ_}, {alpha_, betha_}, OutputAddress_] := Module[{},
  f[v_, w_, x_] := λ v[x]^p + μ w[x]^-q;
  T[vk_, wk_, x_] := Module[{xi, xx},
    F1[xx_, xi_] = G[xx, xi][[2]] f[vk, wk, xi];
    F2[xx_, xi_] = G[xx, xi][[1]] f[vk, wk, xi];
    NInt[F1, F2][x] // Chop];
  vk[x_] := v0[x] /. alp → alpha;
  wk[x_] := w0[x] /. bet → betha;
  k = 0;
  S = {};
  size = Medium;
  s0 = Plot[{wk[x], vk[x]}, {x, 0, 1}, PlotLegends → {"w_k", "v_k"}, ImageSize → size
    , PlotStyle → {{Black, Thickness[0.004]}, {Dashed, Black, Thickness[0.005]}}];
  AppendTo[S, s0];
  Error = {};
  pogr[x_] := Chop[1/2 (wk[x] - vk[x])];
  err = Maximize[{pogr[x], 0 ≤ x ≤ 1}, x][[1]];
  AppendTo[Error, err];
  (*Method main loop*)
  While[Last[Error] > eps && k ≤ MAXITER,
   k = k + 1;
   vk[x_] = T[vk, wk, x];
   wk[x_] = T[wk, vk, x];
   err = Maximize[{pogr[x], 0 ≤ x ≤ 1}, x][[1]];
   AppendTo[Error, err];
   sk = Plot[{wk[x], vk[x]}, {x, 0, 1}, ImageSize → size
     , PlotStyle → {{Black, Thickness[0.004]}, {Dashed, Black, Thickness[0.005]}}];
   AppendTo[S, sk];
  ];

  ut[x_] = 1/2 (vk[x] + wk[x]);
  st = Plot[Legended[ut[x], "u*"], {x, 0, 1}, PlotStyle → {DotDashed, Thickness → 0.005}, ImageSize → size];
  AppendTo[S, st];
  pogr[x_] = Chop[1/2 (wk[x] - vk[x])];
  err = FindMaxValue[pogr[x], {x, 0, 1}];
  normut = FindMaxValue[ut[x], {x, 0, 1}];
  relerr = err / normut;
  ErrRatios = Table[Error[[j + 1]] / Error[[j]], {j, 1, k}];
  convRateEst1 = Median[ErrRatios];
  convRateEst2 = Mean[ErrRatios];
  (*Output*)
  MethodGraph = Show[S, ImageSize → size, AxesLabel → {"x", "w^|k| (x), v^|k| (x)"}];
  label = Row[{Subscript["w", k], Subscript["-v", k]}];
  ErrorGraph = Plot[wk[x] - vk[x], {x, 0, 1}, PlotLegends → label, ImageSize → Small];
  tab1 = Table[{j, Error[[j + 1]], ErrRatios[[j]]}, {j, 1, k}];
  ConvRateEstGrid = Grid[Transpose[PrependTo[tab1, {"k", "err", "q est"}]]];
  tab = Table[{xo, ut[xo]}, {xo, 0, 1, 0.2}];
  UnGrid = Grid[Transpose[PrependTo[tab, {"x", "u^|*|"}]]];
  OutputString = Row[{" ||u_*|=", normut, "  Error=", err, "  Relative error=", relerr, "  k=", k, " , Convergence rate estimate=",
     convRateEst1, " or ", convRateEst2}];
  OutputArray = {normut, err, relerr, k, convRateEst1, convRateEst2};
  If[OutputAddress ≠ "",
   {Save[OutputAddress, MethodGraph];
    Save[OutputAddress, ErrorGraph];
    Save[OutputAddress, ConvRateEstGrid];
    Save[OutputAddress, UnGrid];
    Save[OutputAddress, OutputString];
    Save[OutputAddress, OutputArray];}
  ];
  OutputArray
] (*end of module*)
```

Main loop

```
params = {0, 0, 0, 0};
albet = {0, 0};
methodOutput = {0, 0, 0, 0, 0, 0, 0};
OutputFileName = "";
OutputAddress = "";
FullOutput = {{"p", "q", "λ", "μ", "α", "β", "‖uₙ‖", "абс. похибка", "відн. похибка", "ітерацій", "оцінка збіжності1", "оцінка збіжності2"}};
ShortOutput = {{"p", "q", "‖\!\(\*SubscriptBox[u, n]\)‖"}};
GeneralOutputFileName = "GeneralOutput.nb";
GeneralOutputAddress = FileNameJoin[{OutputFolder, GeneralOutputFileName}];
IterationOutputFileName = "IterationOutput.nb";
IterationOutputAddress = FileNameJoin[{OutputFolder, IterationOutputFileName}];
```

```
(*Loop*)
For[p = pbeg, p <= pend, p = p + hp,
 For[q = qbeg, q <= qend, q = q + hq,
  {params = {p, q, λ, μ};
   If[flagNotSaveToFile ≠ 1,
    OutputFileName = "Output p=" <> ToString[p] <> " q=" <> ToString[q] <> " lam=" <> ToString[λ] <> " mu=" <> ToString[μ] <> " .nb";
    OutputAddress = FileNameJoin[{OutputFolder, OutputFileName}];
   ];
   albet = AlBetCalc[params, OutputAddress];
   methodOutput = MainMethod[params, albet, OutputAddress];
   IteartionOutput = Flatten[{params, albet, methodOutput}];
   AppendTo[FullOutput, IteartionOutput];
   AppendTo[ShortOutput, {p, q, methodOutput[[1]]}];
   Save[IterationOutputAddress, IteartionOutput];
   Print["p=", p, " ,q=", q, " done"]}
 ]
]
```

```
(*Output*)
FullOutputGrid = Grid[FullOutput, Frame → All]
ShortOutputGrid = Grid[ShortOutput, Frame → All]
NormOutputGraph = ListPlot3D[Drop[ShortOutputGrid[[1]], 1]]
Save[GeneralOutputAddress, FullOutputGrid];
Save[GeneralOutputAddress, ShortOutputGrid];
Save[GeneralOutputAddress, NormOutputGraph];
```

ВІДОМІСТЬ АТЕСТАЦІЙНОЇ РОБОТИ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1	Пояснювальна записка	80 с.
2	Презентаційний матеріал	27 с.
	Інші документи	
3	Роздруківки програм	4 с.
4	Рецензія	2 с.
5	Відгук керівника	1 с.

					Застосування методу двобічних наближень до розв'язання мішаних крайових задач для напівлінійних звичайних диференціальних рівнянь			
Змін	**Арк.**	**Номер докум.**	**Підп.**	**Дата**				
Розроб.		**Вороненко М.Д**			**(Тема роботи) Відомість атестаційної роботи**		**Аркуш**	**Аркушів**
Перевір.		**Сидоров М.В.**						
Н. контр.		**Сидоров М.В.**				**ХНУРЕ Кафедра ПМ**		
Затв.		**Тевяшев А.Д.**						