```
(*********************************************************************)
(*                                                                   *)
(* Generalized Collocation Methods:Solutions to Nonlinear Problems *)
(* Bellomo,N.,Lods,B.,Revelli,R.,Ridolfi,L.                          *)
(* A Birkhäuser book                                                 *)
(* ISBN:978-0-8176-4525-0                                            *)
(*                                                                   *)
(* Program EvolPatt                                                  *)
(*                                                                   *)
(*********************************************************************)

$TextStyle = {FontFamily -> "Times", FontSize → 12};
Off[General::"spell", General::"spell1"]

EvolPatt[BoundDataUX_, BoundDataVX_, BoundDataUY_,
  BoundDataVY_, IniData_, NodesX_, NodesY_, TypeX_, TypeY_] :=
 Module[{},

  (*** INITIAL CONDITION ***)

  φU[x_, y_] := IniData[[1]];
  φV[x_, y_] := IniData[[2]];

  hhX = 1/(NodesX - 1);
  hhY = 1/(NodesX - 1);

  Which[TypeX === Sinc, x1_i_ := (i - 1) * hhX,
   TypeX === Lagrange, x1_i_ := -1/2. (Cos[(i - 1) * π/(NodesX - 1)] - 1)];
  Which[TypeY === Sinc, y1_i_ := (i - 1) * hhY, TypeY === Lagrange,
   y1_i_ := -1/2. (Cos[(i - 1) * π/(NodesY - 1)] - 1)];

  If[TypeX === Lagrange, LagrX[j_, x_] := ∏_(p=1)^NodesX (If[p ≠ j, (x - x1_p)/(x1_j - x1_p), 1])];

  If[TypeY === Lagrange, LagrY[j_, y_] := ∏_(p=1)^NodesY (If[p ≠ j, (y - y1_p)/(y1_j - y1_p), 1])];

  If[TypeX === Sinc, SincX[j_, x_] := Which[0 ≤ x ≤ 1 && x ≠ (j - 1) * hhX,
     Sin[π * (x - (j - 1) * hhX)/hhX]/(π * (x - (j - 1) * hhX)/hhX), x == (j - 1) * hhX, 1]];

  If[TypeY === Sinc, SincY[j_, y_] := Which[0 ≤ y ≤ 1 && y ≠ (j - 1) * hhY,
     Sin[π * (y - (j - 1) * hhY)/hhY]/(π * (y - (j - 1) * hhY)/hhY), y == (j - 1) * hhY, 1]];

  If[TypeX === Lagrange && TypeY === Lagrange, InitValueU[x_, y_] :=
    ∑_(m=1)^NodesX ∑_(n=1)^NodesY ((φU[x, y] /. {x → x1_m, y → y1_n}) * LagrX[m, x] * LagrY[n, y]);

   InitValueV[x_, y_] := ∑_(m=1)^NodesX ∑_(n=1)^NodesY
     ((φV[x, y] /. {x → x1_m, y → y1_n}) * LagrX[m, x] * LagrY[n, y])];
```

```
If[TypeX === Lagrange && TypeY === Sinc, InitValue[x_, y_] :=
  Sum_{m=1}^{NodesX} Sum_{n=1}^{NodesY} ((φ[x, y] /. {x → x1_m, y → y1_n}) * LagrX[m, x] * SincY[n, y])];
If[TypeX === Sinc && TypeY === Lagrange, InitValue[x_, y_] :=
  Sum_{m=1}^{NodesX} Sum_{n=1}^{NodesY} ((φ[x, y] /. {x → x1_m, y → y1_n}) * SincX[m, x] * LagrY[n, y])];

If[TypeX === Sinc && TypeY === Sinc, InitValueU[x_, y_] := Sum_{m=1}^{NodesX}
   Sum_{n=1}^{NodesY} ((φU[x, y] /. {x → x1_m, y → y1_n}) * SincX[m, x] * SincY[n, y]); InitValueV[x_,
   y_] := Sum_{m=1}^{NodesX} Sum_{n=1}^{NodesY} ((φV[x, y] /. {x → x1_m, y → y1_n}) * SincX[m, x] * SincY[n, y])];

Print["Initial condition interpolation:"];
Plot3D[Evaluate[InitValueU[x, y]],
 {x, 0, 1}, {y, 0, 1}, PlotRange → All, AxesLabel → {x, y, u}];
Plot3D[Evaluate[InitValueV[x, y]], {x, 0, 1}, {y, 0, 1},
 PlotRange → {Automatic, Automatic, {0, 2}}, AxesLabel → {x, y, v}];

Print["Don't worry! I'm working"];

(*** Derivative Matrices ***)
If[TypeX === Lagrange,
 FDMX[j_, i_] := Which[i == j, Sum_{k=1}^{NodesX} If[k == i, 0, 1/(x1_i - x1_k)],
   i != j, (Prod_{p=1}^{NodesX} (If[p == i, 1, x1_i - x1_p])) / ((x1_i - x1_j) Prod_{p=1}^{NodesX} (If[p == j, 1, x1_j - x1_p]))];
 FirstDerX = Table[FDMX[j, i], {j, 1, NodesX}, {i, 1, NodesX}];
 SecondDerX =
  Table[If[i ≠ j, 2 (FirstDerX[[j, i]] * FirstDerX[[i, i]] - FirstDerX[[j, i]] / (x1_i - x1_j)), 0],
   {j, 1, NodesX}, {i, 1, NodesX}];
 Do[SecondDerX[[i, i]] = - Sum_{k=1}^{NodesX} SecondDerX[[k, i]], {i, 1, NodesX}]];

If[TypeY === Lagrange,
 FDMY[j_, i_] := Which[i == j, Sum_{k=1}^{NodesY} If[k == i, 0, 1/(y1_i - y1_k)],
   i != j, (Prod_{p=1}^{NodesY} (If[p == i, 1, y1_i - y1_p])) / ((y1_i - y1_j) Prod_{p=1}^{NodesY} (If[p == j, 1, y1_j - y1_p]))];
 FirstDerY = Table[FDMY[j, i], {j, 1, NodesY}, {i, 1, NodesY}];
 SecondDerY =
  Table[If[i ≠ j, 2 (FirstDerY[[j, i]] * FirstDerY[[i, i]] - FirstDerY[[j, i]] / (y1_i - y1_j)), 0],
   {j, 1, NodesY}, {i, 1, NodesY}];
```

```
Do[SecondDerY[[i, i]] = -Sum[SecondDerY[[k, i]], {k, 1, NodesY}], {i, 1, NodesY}];

If[TypeX === Sinc,
 FDMX[j_, i_] := Which[i != j, (-1)^(i-j)/(hhX * (i - j)), i == j, 0];
 SDMX[j_, i_] := Which[i != j, 2 * (-1)^(i-j+1)/(hhX^2 * (i - j)^2), i == j, -π^2/(3 * hhX^2)];
 FirstDerX = Table[FDMX[j, i], {j, 1, NodesX}, {i, 1, NodesX}];
 SecondDerX = Table[SDMX[j, i], {j, 1, NodesX}, {i, 1, NodesX}]];

If[TypeY === Sinc,
 FDMY[j_, i_] := Which[i != j, (-1)^(i-j)/(hhY * (i - j)), i == j, 0];
 SDMY[j_, i_] := Which[i != j, 2 * (-1)^(i-j+1)/(hhY^2 * (i - j)^2), i == j, -π^2/(3 * hhY^2)];
 FirstDerY = Table[FDMY[j, i], {j, 1, NodesY}, {i, 1, NodesY}];
 SecondDerY = Table[SDMY[j, i], {j, 1, NodesY}, {i, 1, NodesY}]];

(*** Nodal Equations ***)

NdEqU[i_, j_] := CU_{i,j}'[t] ==
   TT (D1 ((Sum_{k=1}^{NodesX} SecondDerX[[k, i]] * CU_{k,j}[t]) + (Sum_{k=1}^{NodesY} SecondDerY[[k, j]] * CU_{i,k}[t])) +
      ϰ (a - CU_{i,j}[t] + CU_{i,j}[t]^2 * CV_{i,j}[t])) /. {x -> x1_i, y -> y1_j};

NdEqV[i_, j_] := CV_{i,j}'[t] == TT (D2 ((Sum_{k=1}^{NodesX} SecondDerX[[k, i]] * CV_{k,j}[t]) +
         (Sum_{k=1}^{NodesY} SecondDerY[[k, j]] * CV_{i,k}[t])) +
      ϰ * (b - CU_{i,j}[t]^2 * CV_{i,j}[t])) /. {x -> x1_i, y -> y1_j};


(*** Neumann Condition Calculus ***)


Do[{aa, bb} = {FirstDerX[[1, 1]], FirstDerX[[NodesX, 1]]};
 {dd, ee} = {FirstDerX[[1, NodesX]], FirstDerX[[NodesX, NodesX]]};
 {cc, ff} =
  {(Sum_{k=2}^{NodesX-1} FirstDerX[[k, 1]] * CU_{k,j}[t]), (Sum_{k=2}^{NodesX-1} FirstDerX[[k, NodesX]] * CU_{k,j}[t])};
 {gg, hh} = {BoundDataUX[[1]], BoundDataUX[[2]]};

 u_{1,j} = Simplify[(cc * ee - bb * ff - ee * gg + bb * hh)/(bb * dd - aa * ee)];
 u_{NodesX,j} = Simplify[-(cc * dd - aa * ff - dd * gg + aa * hh)/(bb * dd - aa * ee)],
 {j, 1, NodesY}];
```

```
Do[{aa, bb} = {FirstDerY[[1, 1]], FirstDerY[[NodesY, 1]]};
 {dd, ee} = {FirstDerY[[1, NodesY]], FirstDerY[[NodesY, NodesY]]};
 {cc, ff} =
  {(Sum[FirstDerY[[k, 1]] * CU_{i,k}[t], {k, 2, NodesY-1}]),
   (Sum[FirstDerY[[k, NodesY]] * CU_{i,k}[t], {k, 2, NodesY-1}])};
 {gg, hh} = {BoundDataUY[[1]], BoundDataUY[[2]]};

 u_{i,1} = Simplify[(cc * ee - bb * ff - ee * gg + bb * hh)/(bb * dd - aa * ee)];
 u_{i,NodesY} = Simplify[-(cc * dd - aa * ff - dd * gg + aa * hh)/(bb * dd - aa * ee)],
 {i, 2, NodesX - 1}];

Do[{aa, bb} = {FirstDerX[[1, 1]], FirstDerX[[NodesX, 1]]};
 {dd, ee} = {FirstDerX[[1, NodesX]], FirstDerX[[NodesX, NodesX]]};
 {cc, ff} =
  {(Sum[FirstDerX[[k, 1]] * CV_{k,j}[t], {k, 2, NodesX-1}]),
   (Sum[FirstDerX[[k, NodesX]] * CV_{k,j}[t], {k, 2, NodesX-1}])};
 {gg, hh} = {BoundDataVX[[1]], BoundDataVX[[2]]};

 v_{1,j} = Simplify[(cc * ee - bb * ff - ee * gg + bb * hh)/(bb * dd - aa * ee)];
 v_{NodesX,j} = Simplify[-(cc * dd - aa * ff - dd * gg + aa * hh)/(bb * dd - aa * ee)],
 {j, 1, NodesY}];

Do[{aa, bb} = {FirstDerY[[1, 1]], FirstDerY[[NodesY, 1]]};
 {dd, ee} = {FirstDerY[[1, NodesY]], FirstDerY[[NodesY, NodesY]]};
 {cc, ff} =
  {(Sum[FirstDerY[[k, 1]] * CV_{i,k}[t], {k, 2, NodesY-1}]),
   (Sum[FirstDerY[[k, NodesY]] * CV_{i,k}[t], {k, 2, NodesY-1}])};
 {gg, hh} = {BoundDataVY[[1]], BoundDataVY[[2]]};

 v_{i,1} = Simplify[(cc * ee - bb * ff - ee * gg + bb * hh)/(bb * dd - aa * ee)];
 v_{i,NodesY} = Simplify[-(cc * dd - aa * ff - dd * gg + aa * hh)/(bb * dd - aa * ee)],
 {i, 2, NodesX - 1}];

cond1U = Table[{CU_{1,j}[t] → u_{1,j}}, {j, 1, NodesY}];
cond2U = Table[{CU_{NodesX,j}[t] → u_{NodesX,j}}, {j, 1, NodesY}];
cond3U = Table[{CU_{i,1}[t] → u_{i,1}}, {i, 2, NodesX - 1}];
cond4U = Table[{CU_{i,NodesY}[t] → u_{i,NodesY}}, {i, 2, NodesX - 1}];
condTabU = Flatten[Join[cond1U, cond2U, cond3U, cond4U]];

cond1V = Table[{CV_{1,j}[t] → v_{1,j}}, {j, 1, NodesY}];
cond2V = Table[{CV_{NodesX,j}[t] → v_{NodesX,j}}, {j, 1, NodesY}];
cond3V = Table[{CV_{i,1}[t] → v_{i,1}}, {i, 2, NodesX - 1}];
cond4V = Table[{CV_{i,NodesY}[t] → v_{i,NodesY}}, {i, 2, NodesX - 1}];
condTabV = Flatten[Join[cond1V, cond2V, cond3V, cond4V]];

EqsSysU =
 Flatten[Table[{NdEqU[i, j] /. condTabU, CU_{i,j}[0] == (φU[x, y] /. {x → x1_i, y → y1_j})},
   {i, 2, NodesX - 1}, {j, 2, NodesY - 1}]];
EqsSysV = Flatten[Table[{NdEqV[i, j] /. condTabV, CV_{i,j}[0] ==
     (φV[x, y] /. {x → x1_i, y → y1_j})}, {i, 2, NodesX - 1}, {j, 2, NodesY - 1}]];
```

```
NdEqsSys = Flatten[Join[EqsSysU, EqsSysV]];
UnKnown = Flatten[Join[Table[CU_{i,j}, {i, 2, NodesX - 1}, {j, 2, NodesY - 1}],
   Table[CV_{i,j}, {i, 2, NodesX - 1}, {j, 2, NodesY - 1}]]];
NumSol = NDSolve[NdEqsSys, UnKnown, {t, 0, 1}, MaxSteps → 10000000] // Flatten;

(*** Solution ***)

aaa = Partition[NumSol, NodesY - 2];
If[TypeX === Lagrange && TypeY === Lagrange, TotSolU[x_, y_, t_] :=
  Sum_{m=1}^{NodesX-2} Sum_{n=1}^{NodesY-2} (aaa[[m, n, 2]][t] * LagrX[m + 1, x] * LagrY[n + 1, y]) +
    (Sum_{j=1}^{NodesY} (u_{1,j} * LagrX[1, x] * LagrY[j, y]) + Sum_{j=1}^{NodesY} (u_{NodesX,j} * LagrX[NodesX, x] *
        LagrY[j, y]) + Sum_{i=2}^{NodesX-1} (u_{i,1} * LagrX[i, x] * LagrY[1, y]) +
      Sum_{i=2}^{NodesX-1} (u_{i,NodesY} * LagrX[i, x] * LagrY[NodesY, y])) /. condTabU /. NumSol];
If[TypeX === Lagrange && TypeY === Sinc, TotSolU[x_, y_, t_] :=
  Sum_{m=1}^{NodesX-2} Sum_{n=1}^{NodesY-2} (aaa[[m, n, 2]][t] * LagrX[m + 1, x] * SincY[n + 1, y]) +
    (Sum_{j=1}^{NodesY} (u_{1,j} * LagrX[1, x] * SincY[j, y]) + Sum_{j=1}^{NodesY} (u_{NodesX,j} * LagrX[NodesX, x] *
        SincY[j, y]) + Sum_{i=2}^{NodesX-1} (u_{i,1} * LagrX[i, x] * SincY[1, y]) +
      Sum_{i=2}^{NodesX-1} (u_{i,NodesY} * LagrX[i, x] * SincY[NodesY, y])) /. condTabU /. NumSol];
If[TypeX === Sinc && TypeY === Lagrange, TotSolU[x_, y_, t_] :=
  Sum_{m=1}^{NodesX-2} Sum_{n=1}^{NodesY-2} (aaa[[m, n, 2]][t] * SincX[m + 1, x] * LagrY[n + 1, y]) +
    (Sum_{j=1}^{NodesY} (u_{1,j} * SincX[1, x] * LagrY[j, y]) + Sum_{j=1}^{NodesY} (u_{NodesX,j} * SincX[NodesX, x] *
        LagrY[j, y]) + Sum_{i=2}^{NodesX-1} (u_{i,1} * SincX[i, x] * LagrY[1, y]) +
      Sum_{i=2}^{NodesX-1} (u_{i,NodesY} * SincX[i, x] * LagrY[NodesY, y])) /. condTabU /. NumSol];
If[TypeX === Sinc && TypeY === Sinc, TotSolV[x_, y_, t_] :=
  Sum_{m=1}^{NodesX-2} Sum_{n=1}^{NodesY-2} (aaa[[m, n, 2]][t] * SincX[m + 1, x] * SincY[n + 1, y]) +
    (Sum_{j=1}^{NodesY} (u_{1,j} * SincX[1, x] * SincY[j, y]) + Sum_{j=1}^{NodesY} (u_{NodesX,j} * SincX[NodesX, x] *
        SincY[j, y]) + Sum_{i=2}^{NodesX-1} (u_{i,1} * SincX[i, x] * SincY[1, y]) +
```

```
          Sum_{i=2}^{NodesX-1} (u_{i,NodesY} * SincX[i, x] * SincY[NodesY, y]) ] /. condTabU /. NumSol];
  Print["Solution:"];

  plot1 = Evaluate[TotSolU[x, y, t] /. t → 1];
  Plot3D[plot1, {x, 0, 1}, {y, 0, 1}, PlotRange → {Automatic, Automatic, {0, 3}},
   AxesLabel → TraditionalForm /@ {x, y, "u(x,y)"},
   Ticks → {{0, 1}, {0, 1}, {0, 3}}, PlotPoints → 51];
 ];


{NodesX, NodesY} = {51, 51};
{D1, D2, ϰ, a, b, TT} = {0.05, 1, 100, 0.1305, 0.7695, 2};
concinizV = b / (a + b)^2;
concinizU = a + b + 10^-3 Exp[-100 ((x - 1/3)^2 + (y - 1/2)^2)];
BDataUX = {0, 0};
BDataUY = {0, 0};
BDataVX = {0, 0};
BDataVY = {0, 0};
IData = {concinizU, concinizV};
{TypeX, TypeY} = {Lagrange, Lagrange};

EvolPatt[BDataUX, BDataVX, BDataUY, BDataVY, IData, NodesX, NodesY, TypeX, TypeY]
```