```
(*********************************************************************)
(*                                                                   *)
(* 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 HeatTwoDim                                                *)
(*                                                                   *)
(*********************************************************************)

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

HeatTwoDim[InitBoundDataX_, InitBoundDataY_,
  NodesX_, NodesY_, TypeX_, TypeY_] := Module[{},

  (*** INITIAL CONDITION ***)

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

Print["Initial condition interpolation:"];

Plot3D[Evaluate[InitValue[x, y]], {x, 0, 1}, {y, 0, 1}, PlotRange → All];

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, (∏_{p=1}^{NodesX} (If[p == i, 1, x1_i - x1_p])) / ((x1_i - x1_j) ∏_{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, (∏_{p=1}^{NodesY} (If[p == i, 1, y1_i - y1_p])) / ((y1_i - y1_j) ∏_{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_{k=1}^{NodesY} SecondDerY[[k, i]], {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 ***)

NdEq[i_, j_] := C_{i,j}'[t] ==
   1/π^2 (( Sum_{k=1}^{NodesX} SecondDerX[[k, i]] * C_{k,j}[t] ) + ( Sum_{k=1}^{NodesY} SecondDerY[[k, j]] * C_{i,k}[t] )) /.
  {x -> x1_i, y -> y1_j};

Eq0X = Flatten[Table[C_{1,j}'[t] == D[InitBoundDataX[[1]], t] /. y → y1_j, {j, 1, NodesY}]];
EqMX =
 Flatten[Table[C_{NodesX,j}'[t] == D[InitBoundDataX[[3]], t] /. y → y1_j, {j, 1, NodesY}]];

Eq0Y =
 Flatten[Table[C_{i,1}'[t] == D[InitBoundDataY[[1]], t] /. x → x1_i, {i, 2, NodesX - 1}]];
EqMY = Flatten[Table[C_{i,NodesY}'[t] == D[InitBoundDataY[[3]], t] /. x → x1_i,
   {i, 2, NodesX - 1}]];

cEq0X = Flatten[Table[C_{1,j}[0] == φ[x, y] /. {x → x1_1, y → y1_j}, {j, 1, NodesY}]];
cEqMX = Flatten[Table[C_{NodesX,j}[0] == φ[x, y] /. {x → x1_NodesX, y → y1_j}, {j, 1, NodesY}]];

cEq0Y = Flatten[Table[C_{i,1}[0] == φ[x, y] /. {x → x1_i, y → y1_1}, {i, 2, NodesX - 1}]];
cEqMY =
 Flatten[Table[C_{i,NodesY}[0] == φ[x, y] /. {x → x1_i, y → y1_NodesY}, {i, 2, NodesX - 1}]];


EqsSys = Flatten[Table[{NdEq[i, j], C_{i,j}[0] == (φ[x, y] /. {x → x1_i, y → y1_j})},
   {i, 2, NodesX - 1}, {j, 2, NodesY - 1}]];
NdEqsSys = Join[EqsSys, Eq0X, EqMX, Eq0Y, EqMY, cEq0X, cEqMX, cEq0Y, cEqMY];
UnKnown = Flatten[Join[Table[C_{i,j}, {i, 1, NodesX}, {j, 1, NodesY}]]];
NumSol = NDSolve[NdEqsSys, UnKnown, {t, 0, 1}, MaxSteps → 10000000] // Flatten;


(*** Solution ***)
aa = Partition[NumSol, NodesY];
If[TypeX === Lagrange && TypeY === Lagrange,
 TotSol[x_, y_, t_] := Sum_{m=1}^{NodesX} Sum_{n=1}^{NodesY} (aa[[m, n, 2]][t] * LagrX[m, x] * LagrY[n, y])];
If[TypeX === Lagrange && TypeY === Sinc,
 TotSol[x_, y_, t_] := Sum_{m=1}^{NodesX} Sum_{n=1}^{NodesY} (aa[[m, n, 2]][t] * LagrX[m, x] * SincY[n, y])];
If[TypeX === Sinc && TypeY === Lagrange,
```

```
    TotSol[x_, y_, t_] := Sum[Sum[(aa[[m, n, 2]][t] * SincX[m, x] * LagrY[n, y]), {n, 1, NodesY}], {m, 1, NodesX}];
  If[TypeX === Sinc && TypeY === Sinc, TotSol[x_, y_, t_] :=
    Sum[Sum[(aa[[m, n, 2]][t] * SincX[m, x] * SincY[n, y]), {n, 1, NodesY}], {m, 1, NodesX}];
  Print["Solution:"];

  Plot3D[Evaluate[TotSol[x, y, 1]], {x, 0, 1},
   {y, 0, 1}, PlotRange → {Automatic, Automatic, {-1, 1}},
   AxesLabel → TraditionalForm /@ {x, y, "u(x,y)"},
   Ticks → {{0, 1}, {0, 1}, {-1, 0, 1}}, PlotPoints → 51];
 ];


{NodesX, NodesY} = {11, 11};
Anal[x_, y_, t_] := Exp[-t/2] Cos[π/2 (x + y)] + Exp[-2 t] Sin[π (x - y)];
conciniz = Anal[x, y, 0];
concBoundX0 = Anal[0, y, t];
concBoundX1 = Anal[1, y, t];
concBoundY0 = Anal[x, 0, t];
concBoundY1 = Anal[x, 1, t];
IBDataX = {concBoundX0, conciniz, concBoundX1};
IBDataY = {concBoundY0, conciniz, concBoundY1};

HeatTwoDim[IBDataX, IBDataY, NodesX, NodesY, Lagrange, Lagrange]
```