```
(**************************************************************)
(*                                                            *)
(* 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 ConvLaSi                                           *)
(*                                                            *)
(**************************************************************)

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

ConvLaSi[InitBoundData_, Nodes_, Type_] := Module[{},

  (*** INITIAL CONDITION ***)

  φ[x_] := InitBoundData[[2]];

  hh = 1/(Nodes - 1);

  Which[Type === Sinc, x1_i_ := (i - 1) * hh,
   Type === Lagrange, x1_i_ := -1/2. (Cos[(i - 1) * π/(Nodes - 1)] - 1)];

  If[Type === Lagrange, Lagr[j_, x_] := ∏_{p=1}^{Nodes} (If[p ≠ j, (x - x1_p)/(x1_j - x1_p), 1]);

   InitValue[x_] := ∑_{k=1}^{Nodes} (φ[x] /. x → x1_k) * Lagr[k, x]];

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

   InitValue[x_] := ∑_{k=1}^{Nodes} (φ[x] /. x -> x1_k) * Sinc[k, x]];

  Print["Initial condition interpolation:"];
  Plot[Evaluate[{φ[x], InitValue[x]}], {x, 0, 1},
   PlotStyle → {GrayLevel[0], Dashing[{0.15, 0.05}]}, AxesLabel → {x, u[x, 0]}];

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

  (*** Derivative matrices ***)

  If[Type === Lagrange,

   FDM[j_, i_] := Which[i == j, ∑_{k=1}^{Nodes} If[k == i, 0, 1/(x1_i - x1_k)], i != j,
     (∏_{p=1}^{Nodes} (If[p == i, 1, x1_i - x1_p])) / ((x1_i - x1_j) ∏_{p=1}^{Nodes} (If[p == j, 1, x1_j - x1_p]))];

   FirstDer = Table[FDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
   SecondDer =
```

```
    Table[If[i ≠ j, 2 (FirstDer[[j, i]] * FirstDer[[i, i]] - FirstDer[[j, i]] / (x1_i - x1_j)), 0],
      {j, 1, Nodes}, {i, 1, Nodes}];
   Do[SecondDer[[i, i]] = - Sum_{k=1}^{Nodes} SecondDer[[k, i]], {i, 1, Nodes}]];

  If[Type === Sinc,
   FDM[j_, i_] := Which[i != j, (-1)^(i-j) / (hh * (i - j)), i == j, 0];
   SDM[j_, i_] := Which[i != j, 2 * (-1)^(i-j+1) / (hh^2 * (i - j)^2), i == j, - π^2 / (3 * hh^2)];
   FirstDer = Table[FDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
   SecondDer = Table[SDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}]];

  (*** Nodal Equations ***)

  NdEq[i_] := C_i '[t] == - (Sum_{k=1}^{Nodes} FirstDer[[k, i]] * C_k[t]) +
      ((Sum_{k=1}^{Nodes} SecondDer[[k, i]] * C_k[t]) + Exp[(1 - x)^2 - t] * (-5 + 2 x - 4 (1 - x)^2)) /. x -> x1_i;

  EqsSys = Flatten[Table[{NdEq[i], C_i[0] == (φ[x] /. x → x1_i)}, {i, 2, Nodes - 1}]];
  NdEqsSys = Join[EqsSys, {C_1 '[t] == D[InitBoundData[[1]], t], C_1[0] == (φ[x] /. x → x1_1),
     C_Nodes '[t] == D[InitBoundData[[3]], t], C_Nodes[0] == (φ[x] /. x → x1_Nodes)}];
  UnKnown = Join[Table[C_i, {i, 1, Nodes}]];
  NumSol = NDSolve[NdEqsSys, UnKnown, {t, 0, 1}, MaxSteps → 10000000] // Flatten;

  (*** Solution ***)
  Which[Type === Sinc, TotSol[x_, t_] := Sum_{k=1}^{Nodes} NumSol[[k, 2]][t] * Sinc[k, x],

   Type === Lagrange, TotSol[x_, t_] := Sum_{k=1}^{Nodes} NumSol[[k, 2]][t] * Lagr[k, x]];

  Print["Solution:"];

  Plot3D[Evaluate[TotSol[x, t]], {x, 0, 1}, {t, 0, 1}, PlotRange →
    {{0, 1}, {0, 1}, {-.1, 1}}, AxesLabel → TraditionalForm /@ {x, t, "u(x,t)"},
   Ticks → {{0, 1}, {0, 1}, {0, 1}}, PlotPoints → 101];
  ];

Nodes = 21;
InitCond = Exp[(1 - x)^2 - 1];
Bound0Cond = Exp[- t];
Bound1Cond = Exp[- t - 1];
IBData = {Bound0Cond, InitCond, Bound1Cond};

ConvLaSi[IBData, Nodes, Lagrange];

ConvLaSi[IBData, Nodes, Sinc];
```