```
(*****************************************************************)
(*                                                               *)
(* 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 KdVV                                                  *)
(*                                                               *)
(*****************************************************************)

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

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

  (*** INITIAL CONDITION ***)

  φ[x_] := InitBoundData[[2]];

  hh = 1/(Nodes - 1);

  Which[Type === Sinc, Subscript[x1, i_] := (i - 1) * hh,
   Type === Lagrange, Subscript[x1, i_] := -1/2. (Cos[(i - 1) * π/(Nodes - 1)] - 1)];

  If[Type === Lagrange, Lagr[j_, x_] := Product[(If[p ≠ j, (x - Subscript[x1, p])/(Subscript[x1, j] - Subscript[x1, p]), 1]), {p, 1, Nodes}];

   InitValue[x_] := Sum[(φ[x] /. x → Subscript[x1, k]) * Lagr[k, x], {k, 1, Nodes}]];

  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_] := Sum[(φ[x] /. x -> Subscript[x1, k]) * Sinc[k, x], {k, 1, Nodes}]];

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

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

  (*** DERIVATIVE MATRICES ***)
  If[Type === Lagrange,

   FDM[j_, i_] := Which[i == j, Sum[If[k == i, 0, 1/(Subscript[x1, i] - Subscript[x1, k])], {k, 1, Nodes}],
     i != j, Product[(If[p == i, 1, Subscript[x1, i] - Subscript[x1, p]]), {p, 1, Nodes}] /
       ((Subscript[x1, i] - Subscript[x1, j]) Product[(If[p == j, 1, Subscript[x1, j] - Subscript[x1, p]]), {p, 1, Nodes}])];

   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[SecondDer[[k, i]], {k, 1, Nodes}], {i, 1, Nodes}];
 ThirdDer =
  Table[If[i ≠ j, 3 (FirstDer[[j, i]] * SecondDer[[i, i]] - SecondDer[[j, i]] / (x1_i - x1_j)), 0],
   {j, 1, Nodes}, {i, 1, Nodes}];
 Do[ThirdDer[[i, i]] = - Sum[ThirdDer[[k, i]], {k, 1, Nodes}], {i, 1, Nodes}];
 FourthDer =
  Table[If[i ≠ j, 3 (FirstDer[[j, i]] * ThirdDer[[i, i]] - ThirdDer[[j, i]] / (x1_i - x1_j)), 0],
   {j, 1, Nodes}, {i, 1, Nodes}];
 Do[FourthDer[[i, i]] = - Sum[FourthDer[[k, i]], {k, 1, Nodes}], {i, 1, Nodes}];
 FifthDer =
  Table[If[i ≠ j, 3 (FirstDer[[j, i]] * FourthDer[[i, i]] - FourthDer[[j, i]] / (x1_i - x1_j)), 0],
   {j, 1, Nodes}, {i, 1, Nodes}];
 Do[FifthDer[[i, i]] = - Sum[FifthDer[[k, i]], {k, 1, Nodes}], {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)];
 TDM[j_, i_] := Which[i != j, (-1)^(i-j) / hh^3 (6 / (i - j)^3 - π^2 / (i - j)), i == j, 0];
 QDM[j_, i_] := Which[i != j, 4 * (-1)^(i+j) / hh^4 (- 6 / (i - j)^4 + π^2 / (i - j)^2), i == j, π^4 / (5 * hh^4)];
 CDM[j_, i_] := Which[i != j, (-1)^(i+j) / hh^5 (120 / (i - j)^5 - 20 * π^2 / (i - j)^3 + π^4 / (i - j)), i == j, 0];

 FirstDer = Table[FDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
 SecondDer = Table[SDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
 ThirdDer = Table[TDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
 FourthDer = Table[QDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}];
 FifthDer = Table[CDM[j, i], {j, 1, Nodes}, {i, 1, Nodes}]];

(*** Nodal Equations ***)
NdEq[i_] := C_i '[t] == (- (T1 - T0) / (X1 - X0)) * (Sum[FirstDer[[k, i]] * C_k[t], {k, 1, Nodes}]) * C_i[t] +
   (- (T1 - T0) / (X1 - X0)^3) * (Sum[ThirdDer[[k, i]] * C_k[t], {k, 1, Nodes}]) +
```

```
      ((T1 - T0)/(X1 - X0)^5) * (Sum_{k=1}^{Nodes} FifthDer[[k, i]] * C_k[t]) /. 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},
   PlotPoints → 81, Boxed → True, AxesLabel → {x, t, u}, PlotRange → All];
  ];


(* One Soliton Solution *)

{T0, T1, X0, X1} = {0, 100, -30, 60};
{Nodes, x0} = {61, 0};
InitCond = 105/169 Sech[1/(2 Sqrt[13]) (((X1 - X0) x + X0) - x0 - 36/169 ((T1 - T0) t + T0))]^4 //. t → 0;
Bound0Cond = 0;
Bound1Cond = 0;

IBData = {Bound0Cond, InitCond, Bound1Cond};

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