```
(*********************************************************************)
(*                                                                   *)
(* 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 Traffic2                                                  *)
(*                                                                   *)
(*********************************************************************)
$TextStyle = {FontFamily -> "Times", FontSize → 12};
Off[General::"spell", General::"spell1"]

Traffic2[InitBoundDataU_, InitBoundDataQ_, Nodes_, η_, Type_] := Module[{},
  (*** INITIAL CONDITION ***)
  φU[x_] := InitBoundDataU[[2]];
  φQ[x_] := InitBoundDataQ[[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]);
   InitValueU[x_] := ∑_{k=1}^{Nodes} (φU[x] /. x → x1_k) * Lagr[k, x];
   InitValueQ[x_] := ∑_{k=1}^{Nodes} (φQ[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];
   InitValueU[x_] := ∑_{k=1}^{Nodes} (φU[x] /. x -> x1_k) * Sinc[k, x];
   InitValueQ[x_] := ∑_{k=1}^{Nodes} (φQ[x] /. x -> x1_k) * Sinc[k, x]];

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

  Plot[Evaluate[{φQ[x], InitValueQ[x]}],
   {x, 0, 1}, PlotStyle → {GrayLevel[0], Dashing[{0.15, 0.05}]},
   AxesLabel → {x, q[x, 0]}, PlotRange → {Automatic, {0, 1}}];

  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[SecondDer[[k, i]], {k, 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 ***)
NdEqU[i_] := U_i'[t] == -(Sum[FirstDer[[k, i]] * Q_k[t], {k, 1, Nodes}]) /. x -> x1_i;
NdEqQ[i_] :=
 Q_i'[t] == (((1 - 2 U_i[t]) + η (U_i[t]) (2 - 3 U_i[t]) * (Sum[FirstDer[[k, i]] * Q_k[t], {k, 1, Nodes}]))
     (Sum[FirstDer[[k, i]] * Q_k[t], {k, 1, Nodes}]) +
    η (U_i[t]) (1 - U_i[t]) (Sum[SecondDer[[k, i]] * Q_k[t], {k, 1, Nodes}])) /. x -> x1_i;
Q1U[t_] := D[InitBoundDataU[[1]], t];
QMU[t_] := D[InitBoundDataU[[3]], t];

Q1Q[t_] := D[InitBoundDataQ[[1]], t];
QMQ[t_] := D[InitBoundDataQ[[3]], t];

Eq0Q = Q_1'[t] == Q1Q[t];
EqMQ = Q_Nodes'[t] == QMQ[t];

EqsSysU = Flatten[Table[{NdEqU[i], U_i[0] == (φU[x] /. x → x1_i)}, {i, 1, Nodes}]];
EqsSysQ = Flatten[Table[{NdEqQ[i], Q_i[0] == (φQ[x] /. x → x1_i)}, {i, 2, Nodes - 1}]];
NdEqsSys = Join[EqsSysU, EqsSysQ,
  {Eq0Q, Q_1[0] == (φQ[x] /. x → x1_1), EqMQ, Q_Nodes[0] == (φQ[x] /. x → x1_Nodes)}];
UnKnown = Join[Table[U_i, {i, 1, Nodes}], Table[Q_i, {i, 1, Nodes}]];
NumSol = NDSolve[NdEqsSys, UnKnown, {t, 0, 1}, MaxSteps → 10000000] // Flatten;

(*** Solution ***)
Which[Type === Sinc, TotSolU[x_, t_] := Sum[NumSol[[k, 2]][t] * Sinc[k, x], {k, 1, Nodes}],
 Type === Lagrange, TotSolU[x_, t_] := Sum[NumSol[[k, 2]][t] * Lagr[k, x], {k, 1, Nodes}]];

Which[Type === Sinc, TotSolQ[x_, t_] := Sum[NumSol[[k + Nodes, 2]][t] * Sinc[k, x], {k, 1, Nodes}],
 Type === Lagrange, TotSolQ[x_, t_] := Sum[NumSol[[k + Nodes, 2]][t] * Lagr[k, x], {k, 1, Nodes}]];
```

```
  Print["Solution:"];

  Plot3D[Evaluate[TotSolU[x, t]], {x, 0, 1}, {t, 0, 1},
   PlotRange → All, AxesLabel → TraditionalForm /@ {x, t, "u(x,t)"},
   Ticks → {{0, 1}, {0, 1}, {0.2, 0.4}}, PlotPoints → 101];
  Plot3D[Evaluate[TotSolQ[x, t]], {x, 0, 1}, {t, 0, 1}, PlotRange →
    {{0, 1}, {0, 1}, {0, 0.3}}, AxesLabel → TraditionalForm /@ {x, t, "q(x,t)"},
   Ticks → {{0, 1}, {0, 1}, {0.1, 0.3}}, PlotPoints → 101];
 ];


{Nodes, η} = {11, .1};
{InitCondU, InitCondQ} = {0.2, 0.16};
{Bound0CondU, Bound0CondQ} = {0, 0.16};
{Bound1CondU, Bound1CondQ} = {0, 0.16 + 0.1 Sin[5 t]};

IBDataU = {Bound0CondU, InitCondU, Bound1CondU};
IBDataQ = {Bound0CondQ, InitCondQ, Bound1CondQ};

Traffic2[IBDataU, IBDataQ, Nodes, η, Lagrange];
```