Notes on some numerical schemes

Nasser M. Abbasi

May 26, 2022 Compiled on January 29, 2024 at 3:03am

Contents

1 Introduction

These are notes to describe some numerical schemes.

1.1 Centered difference for 1D wave PDE

Here we want to solve $u_{tt} = c^2 u_{xx}$ in 1D finite domain $0 < x < L$ and $t > 0$, with boundary conditions $u(0,t) = f(t)\,, u(L,t) = g(t)$ and initial position $u(x,0) = \alpha(x)$ and initial velocity $\frac{\partial u(x,0)}{\partial t} = \beta(x)$.

Centered difference is used.

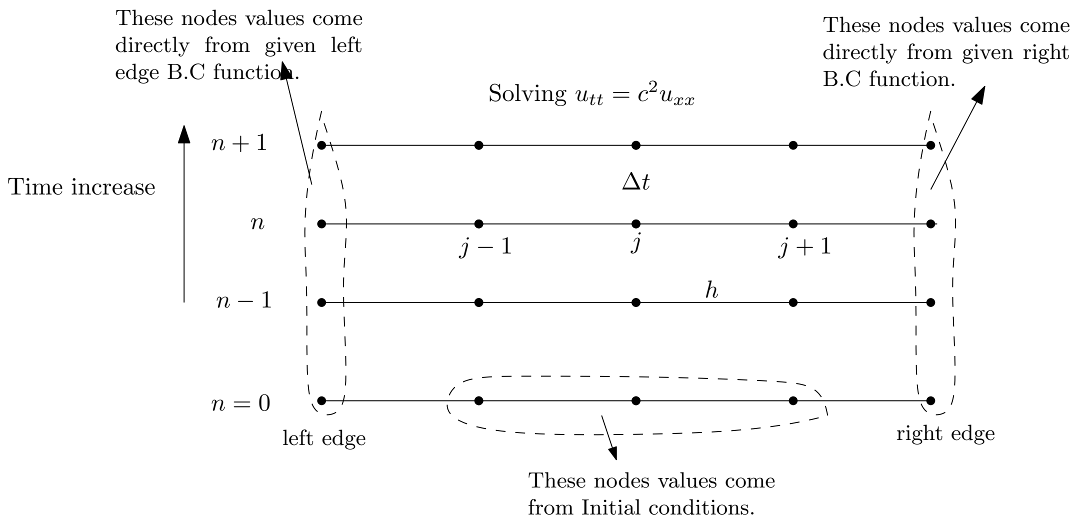

These nodes values come directly from given left edge B.C function.
These nodes values come directly from given right B.C function.
Solving $u_{tt} = c^2 u_{xx}$
Time increase
$n+1$
n
$n-1$
$n=0$
Δt
$j-1$
j
$j+1$
h
left edge
right edge
These nodes values come from Initial conditions.

Figure 1: Centered difference scheme used in time and space

At each internal node j we have the following finite difference represenation of $u_{tt} = c^2 u_{xx}$

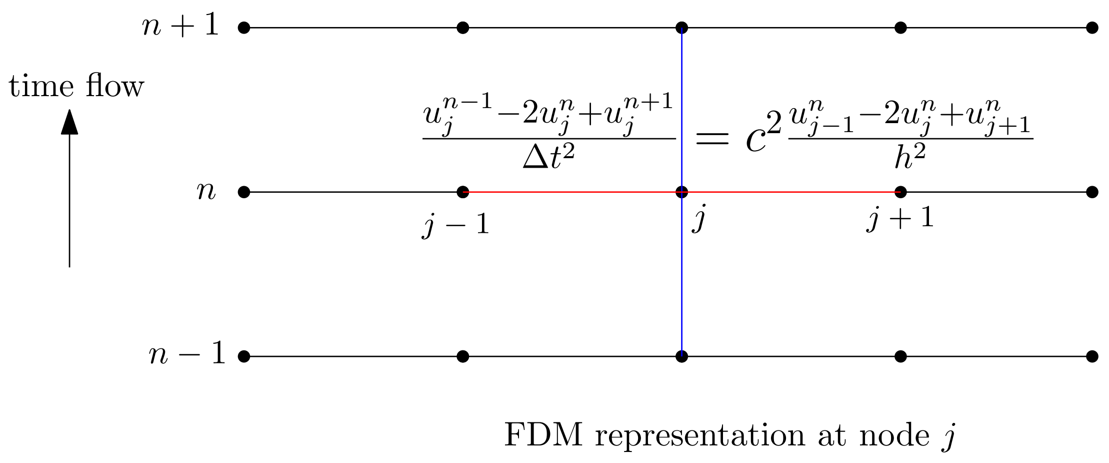

$n+1$
time flow
$\frac{u_j^{n-1} - 2u_j^n + u_j^{n+1}}{\Delta t^2} = c^2 \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{h^2}$
n
$j-1$
j
$j+1$
$n-1$
FDM representation at node j

Figure 2: FDM at each node

To handle initial conditions, initial velocity is used to solve for u_j^{-1}

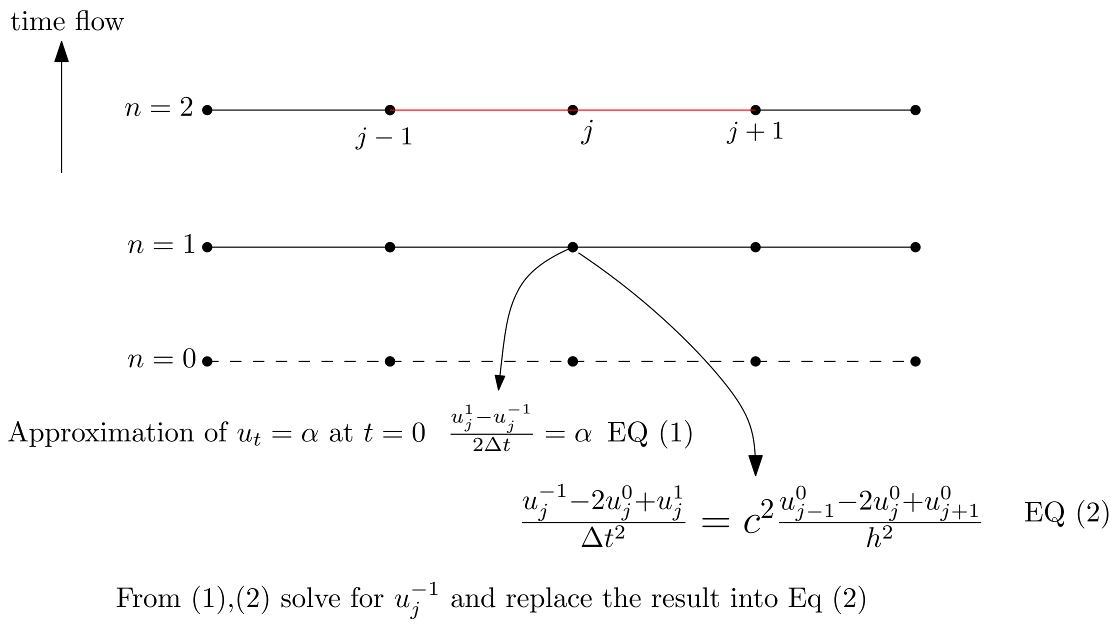

time flow
$n=2$
$j-1$
j
$j+1$
$n=1$
$n=0$
Approximation of $u_t=\alpha$ at $t=0$ $\frac{u_j^1-u_j^{-1}}{2\Delta t}=\alpha$ EQ (1)
$\frac{u_j^{-1}-2u_j^0+u_j^1}{\Delta t^2}=c^2\frac{u_{j-1}^0-2u_j^0+u_{j+1}^0}{h^2}$ EQ (2)
From (1),(2) solve for u_j^{-1} and replace the result into Eq (2)

Figure 3: Solving for u_j^{-1}

This gives all the information needed to find the matrices to use. Let $k=\Delta t$. From Eq(1)

$$\begin{aligned}\frac{u_j^1-u_j^{-1}}{2k}&=\alpha\\ u_j^{-1}&=u_j^1-2k\alpha\end{aligned}$$

Substituting this in Eq(2) gives

$$\begin{aligned}\frac{\left(u_j^1-2k\alpha\right)-2u_j^0+u_j^1}{k^2}&=c^2\frac{u_{j-1}^0-2u_j^0+u_{j+1}^0}{h^2}\\ 2u_j^1&=\frac{k^2c^2}{h^2}\left(u_{j-1}^0-2u_j^0+u_{j+1}^0\right)+2u_j^0+2k\alpha\\ u_j^1&=\frac{1}{2}\frac{k^2c^2}{h^2}\left(u_{j-1}^0-2u_j^0+u_{j+1}^0\right)+u_j^0+k\alpha\end{aligned}$$

Therefore for $n=1$ only and for $j=1\cdots N$ where N is number of nodes

$$\begin{pmatrix}u_1^1\\u_2^1\\u_3^1\\u_4^1\\u_5^1\end{pmatrix}=\frac{1}{2}\frac{k^2c^2}{h^2}\begin{pmatrix}1&0&0&0&0\\1&-2&1&0&0\\0&1&-2&1&0\\0&0&1&-2&1\\0&0&0&0&1\end{pmatrix}\begin{pmatrix}u_1^0\\u_2^0\\u_3^0\\u_4^0\\u_5^0\end{pmatrix}+\begin{pmatrix}u_1^0\\u_2^0\\u_3^0\\u_4^0\\u_5^0\end{pmatrix}+k\alpha$$

Where $\begin{pmatrix} u_1^0 \\ u_2^0 \\ u_3^0 \\ u_4^0 \\ u_5^0 \end{pmatrix}$ is known and comes from boundary and initial conditions. u_1^0 is left B.C. and u_N^0 comes from right B.C. and $u_2^0 \cdots u_{N-1}^0$ comes from initial conditions $u(x,0)$. Now, for $n = 2$ or higher times

$$\begin{aligned}
\frac{u_j^{n-1} - 2u_j^n + u_j^{n+1}}{k^2} &= c^2 \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{h^2} \\
u_j^{n-1} - 2u_j^n + u_j^{n+1} &= \frac{k^2c^2}{h^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) \\
u_j^{n+1} &= \frac{k^2c^2}{h^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) + 2u_j^n - u_j^{n-1}
\end{aligned}$$

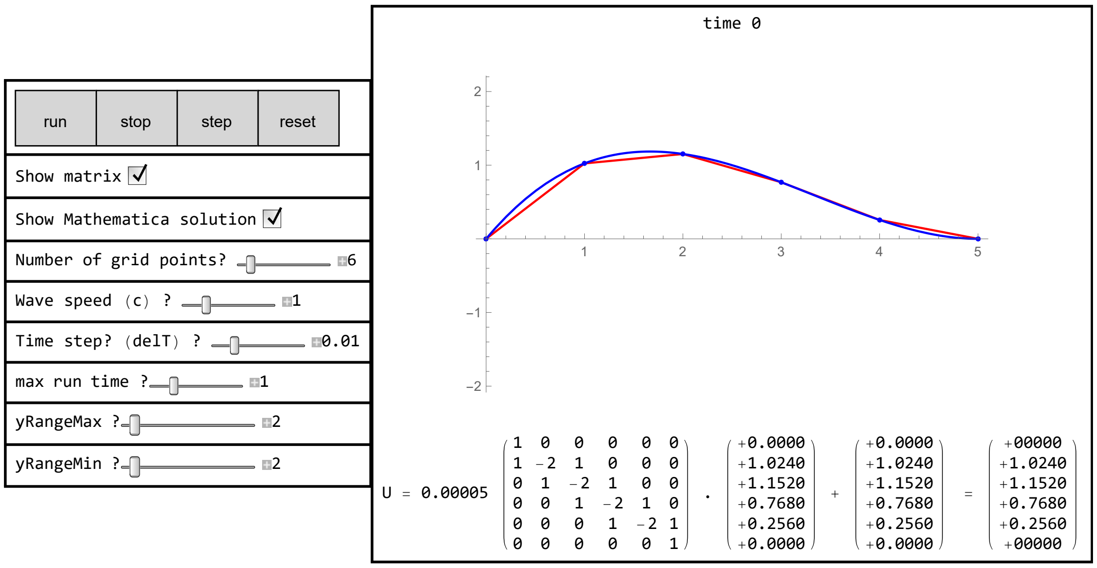
In Matrix form

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \\ u_5^{n+1} \end{pmatrix} = \frac{k^2c^2}{h^2} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ u_4^n \\ u_5^n \end{pmatrix} + 2 \begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \\ u_4^n \\ u_5^n \end{pmatrix} - \begin{pmatrix} u_1^{n-1} \\ u_2^{n-1} \\ u_3^{n-1} \\ u_4^{n-1} \\ u_5^{n-1} \end{pmatrix}$$

So to find u_j^{n+1} we need to know the last time step solution and also the solution for the step before that.

Small Mathematica was written to implement the above scheme. Here is screen show of the GUI for one example.

Out[•]=



Show matrix ✓

Show Mathematica solution ✓

Number of grid points? 6

Wave speed (c) ? 1

Time step? (delT) ? 0.01

max run time ? 1

yRangeMax ? 2

yRangeMin ? 2

```
              ⎛1  0  0  0  0  0⎞   ⎛+0.0000⎞   ⎛+0.0000⎞   ⎛+00000 ⎞
              ⎜1 -2  1  0  0  0⎟   ⎜+1.0240⎟   ⎜+1.0240⎟   ⎜+1.0240⎟
U = 0.00005   ⎜0  1 -2  1  0  0⎟ . ⎜+1.1520⎟ + ⎜+1.1520⎟ = ⎜+1.1520⎟
              ⎜0  0  1 -2  1  0⎟   ⎜+0.7680⎟   ⎜+0.7680⎟   ⎜+0.7680⎟
              ⎜0  0  0  1 -2  1⎟   ⎜+0.2560⎟   ⎜+0.2560⎟   ⎜+0.2560⎟
              ⎝0  0  0  0  0  1⎠   ⎝+0.0000⎠   ⎝+0.0000⎠   ⎝+00000 ⎠
```

Figure 4: 1D FDM for wave pde

The Mathematica notebook of the above is here

The following is an animation of $u_{tt} = 4u_{xx}$ with fixed ends, and string length $L = 5$ with initial position given by $8x\frac{(L-x)^2}{L^3}$ and zero initial velocity. This uses $\Delta t = 0.02$ seconds and 30 grid points over the length 5.

The following is listing of the source code

```
Finite difference for 1D wave PDE
By Nasser M. Abbasi. Version May 8, 2020



L = 5;
leftBC[x_, t_] := 0;(*t^2;*)
rightBC[x_, t_] := 0;(*t^2+25;*)
initialPosition[x_] := 8 x*(5 - x)^2/5^3; (*x^2;*)
initialVelocity = 0;



padIt1[v_, f_List] :=
  AccountingForm[v, f, NumberSigns -> {"-", " "},
   NumberPadding -> {"0", "0"}, SignPadding -> True];
(*these 2 functions thanks to xzczd*)
numberForm[a_List, n_] := numberForm[#, n] & /@ a;
numberForm[a_, n_] := padIt1[a, n];
```

```
makeA[n_] := Module[{A, i, j}, A = Table[0, {i, n}, {j, n}];
   Do[Do[
     A[[i, j]] =
      If[i == j, -2, If[i == j + 1 || i == j - 1, 1, 0]], {j, 1,
      n}], {i, 1, n}];
   A[[1, 1]] = 1;
   A[[1, 2]] = 0;
   A[[-1, -1]] = 1;
   A[[-1, -2]] = 0;
   A];

makeInitialU[nPoints_, L_, h_, leftBC_, rightBC_, initialPosition_] :=
   Module[{u, j, t = 0},
   u = Table[0, {j, nPoints}];
   Do[
    u[[j]] =
     If[j == 1, leftBC[0, 0],
      If[j == nPoints, rightBC[L, 0], initialPosition[(j - 1)*h]]],
    {j, 1, nPoints}
    ];
   u
   ];


makePlot[currentTime_, showMMA_, grid_, currentU_, u_, opt_, opt1_,
   yRangeMin_, yRangeMax_, solN_, showMatrix_, k_, c_, h_, A_,
   initialVelocity_] := Module[{},

   Grid[{
     {Row[{"time ", padIt1[currentTime, {4, 2}]}]},
     {Dynamic@If[showMMA,
        Show[
         ListLinePlot[Transpose[{grid, u}], Evaluate[opt]],
         Plot[solN[x, currentTime], {x, 0, 5}, Evaluate[opt1]],
         PlotRange -> {{0, 5}, {-yRangeMin, yRangeMax}}
         ],
        ListLinePlot[Transpose[{grid, u}],
         Evaluate@
          Join[opt, {PlotRange -> {{0, 5}, {-yRangeMin, yRangeMax}}}]
         ]
        ]
      },
     {Dynamic@If[showMatrix,
        Row[{"U = ", NumberForm[k^2*c^2/2*h^2], " ", MatrixForm[A],
          " . ", MatrixForm[numberForm[u, {5, 4}]], " + ",
          MatrixForm[numberForm[u, {5, 4}]],
          If[initialVelocity != 0, Row[{" + ", k*initialVelocity}]],
          " = ", MatrixForm[numberForm[currentU, {5, 4}]]}]
        ,
        "No matrix display"
        ]}
     }, Spacings -> {1, 1}, Frame -> True]
   ];


DynamicModule[{solN, lastU, currentU, currentTime = 0, A, h,
  showMatrix = False,
  showMMA = False, k = 0.02, nPoints = 30, maxTime = 4,
  yRangeMax = 1.2, yRangeMin = 1.2,
  opt, opt1, pde, ic, bc, grid, g = 0, u, x, t, nextU, c = 4,
  state = "STOP", tick = False},

 opt = {PlotStyle -> Red, AxesOrigin -> {0, 0}, Mesh -> All,
   MeshStyle -> {Blue, PointSize[0.01]},
   ImageSize -> 400, ImagePadding -> 10, ImageMargins -> 10};
 opt1 = {PlotStyle -> Blue, AxesOrigin -> {0, 0}, ImageSize -> 400,
   ImagePadding -> 10, ImageMargins -> 10};

 Dynamic[
  tick;
  If[currentTime == 0,
   A = makeA[nPoints];
   h = L/(nPoints - 1);
   lastU =
    N@makeInitialU[nPoints, L, h, leftBC, rightBC, initialPosition];
   currentU =
    0.5 (c^2*k^2)/h^2*(A . lastU) + lastU + (k*initialVelocity);
   currentU[[1]] = leftBC[0, k];
   currentU[[-1]] = rightBC[L, k];
   pde = D[u[x, t], {t, 2}] == c ^2 D[u[x, t], {x, 2}];
   ic = {u[x, 0] == initialPosition[x],
     Derivative[0, 1][u][x, 0] == initialVelocity};
   bc = {u[0, t] == leftBC[0, t], u[L, t] == rightBC[L, 0]};
   solN =
    Quiet@NDSolveValue[{pde, ic, bc}, u, {x, 0, 5}, {t, 0, maxTime}];
   grid = Range[0, L, h];
   g = makePlot[currentTime, showMMA, grid, currentU, lastU, opt,
     opt1, yRangeMin, yRangeMax, solN, showMatrix, k, c, h, A,
     initialVelocity];
   If[state == "RUN" || state == "STEP",
    If[(currentTime + k) <= maxTime,
     currentTime = currentTime + k
     ,
     state == "STOP"
     ]
    ]
   ,
   If[state != "STOP",
    nextU = (c^2*k^2)/h^2*A . currentU + 2 currentU - lastU;
    nextU[[1]] = leftBC[0, currentTime];
    nextU[[-1]] = rightBC[L, currentTime];

    g = makePlot[currentTime, showMMA, grid, currentU, nextU, opt,
      opt1, yRangeMin, yRangeMax, solN, showMatrix, k, c, h, A,
      initialVelocity];

    If[state == "RUN" || state == "STEP",
     If[(currentTime + k) <= maxTime,
      currentTime = currentTime + k
      ]
     ];
    If[state == "STEP", state = "STOP"];
    lastU = currentU;
    currentU = nextU
    ]
   ];

  Row[{Grid[{
      {Row[{Button[
          Text@Style["run", 12], {currentTime = 0; state = "RUN"},
          ImageSize -> {60, 40}],
       Button[Text@Style["stop", 12], {state = "STOP"},
        ImageSize -> {60, 40}],

       Button[Text@Style["step", 12], {state = "STEP"},
        ImageSize -> {60, 40}],

       Button[Text@Style["reset", 12], {currentTime = 0;
          state = "STOP"}, ImageSize -> {60, 40}]}]
      },
    {Row[{"Show matrix", Spacer[3],
       Checkbox[
        Dynamic[showMatrix, {showMatrix = #;
           tick = Not[tick]} &]]}]},
    {Row[{"Show Mathematica solution", Spacer[3],
       Checkbox[
        Dynamic[showMMA, {showMMA = #; tick = Not[tick]} &]]}]},
    {Row[{"Number of grid points? ",
       Manipulator[
        Dynamic[nPoints, {nPoints = #; currentTime = 0;
           state = "STOP"} &], {3, 50, 1}, ImageSize -> Tiny],
       Dynamic[nPoints]}]},
    {Row[{"Wave speed (c) ? ",
       Manipulator[
        Dynamic[c, {c = #; currentTime = 0;
           state = "STOP"} &], {0.01, 5, 0.01}, ImageSize -> Tiny],
       Dynamic[c]}]},
    {Row[{"Time step? (delT) ? ",
       Manipulator[
        Dynamic[k, {k = #; currentTime = 0;
           state = "STOP"} &], {0.001, 0.05, 0.01},
        ImageSize -> Tiny], Dynamic[k]}]},
    {Row[{"max run time ?",
       Manipulator[
        Dynamic[maxTime, {maxTime = #; currentTime = 0;
           state = "STOP"} &], {0, 5, 0.01}, ImageSize -> Tiny],
       Dynamic[maxTime]}]},
    {Row[{"yRangeMax ?",
       Manipulator[
        Dynamic[yRangeMax, {yRangeMax = #; tick = Not[tick]} &], {1,
          30, 0.01}, ImageSize -> Small], Dynamic[yRangeMax]}]},
    {Row[{"yRangeMin ?",
          Manipulator[
           Dynamic[yRangeMin, {yRangeMin = #; tick = Not[tick]} &], {1,
             30, 0.01}, ImageSize -> Small], Dynamic[yRangeMin]}]}
       }, Alignment -> Left, Spacings -> {1, 1}, Frame -> All
      ], g}
    ]
   ,
   ContinuousAction -> False,
   TrackedSymbols :> {currentTime, state, tick}
   ]

  ]
```