Analyses with Renewed Time Constant

```
In[ ]:= Quit
        終了

In[ ]:= 8 Pi * 0.0007 * 0.01 / 1.6 / 1.6 / 10 ^ (-8)
        円周率

Out[ ]= 6872.23

In[ ]:= Pi (0.01 ^ 2 - 0.007 ^ 2)
        円周率

Out[ ]= 0.000160221
```

One - way Valve Analysis

```
In[ ]:= initialCanalPressure = 0;
        direction = 1;(* direction of the diod 0: normal, 1: reverse *)
        dx = 20; (* efficiency of the diode *)
        diod[v_, r_, x_] := Which[
                            最初の真
            direction == 0, If[v > 0, v / r, v / (x * r)],
                            If文
            direction == 1, If[v < 0, v / r, v / (x * r)]];
                            If文

        n = 100;
        cycle = 1;(* cycle per sec of CSF wave *)
        convmmhg = 133.3223874 ;(* conversion constant from mmHg to Pascal*)
        convcmH2O = 98.0665 ;(* 1 cmH2O = 98.0665 Pascal *)
        subaraP = 10 convcmH2O ;
        pulseAmp = subaraP / 5;
        en[t_] = pulseAmp * Sin[2 Pi * cycle * t] + subaraP ;
                            正弦   円周率
        rc = 1.78 × 10 ^ 11 * 100 / n;
        rs = 6872 * 100 / n;
        Rflux = 10 rc;
        R0 = rs;
        Rout = rc;
        dsub = 10 ^ (-10);
        dtheca = 10 ^ (-3);
```

```
dcanal = 10^(-13);
Dcist = dsub;
STFactor = 50;
SFFactor = .005;
CFFactor = 150000;(* canal flow gain for display *)
rpattern = 1;
r = Which[
    最初の真
    rpattern == 1, Table[rc, n],
                   リストを作成
    rpattern == 2, Join[Table[rc, n/2 - 1], {100 rc}, Table[rc, n/2]],
                   繋ぐ  リストを作成                    リストを作成
    rpattern == 3, Join[Table[rc, n - 1], {1000 rc}]
                   繋ぐ  リストを作成
  ];
Rpattern = 1;
R = Which[
    最初の真
    Rpattern == 1, Table[rs, n],
                   リストを作成
    Rpattern == 2, Join[Table[rs, n/2 - 1], {100 rs}, Table[rs, n/2]]
                   繋ぐ  リストを作成                    リストを作成
  ];

c = Table[dcanal, n - 1];(* canal capacitance *)
    リストを作成
d = Join[Table[dsub, n - 1], {dtheca}];(* subarachnoid capacitance *)
    繋ぐ  リストを作成

k = 25;(* location of one-way valve of the canal *)
Vob[t_] := (Rout r[[1]] en[t] + Rflux r[[1]] Vcist[t] + Rflux Rout (v[1][t] + u[1][t]))/
    (Rout r[[1]] + Rflux r[[1]] + Rflux Rout);
eqn = Join[{v[1]'[t] == ((Vob[t] - v[1][t] - u[1][t])/r[[1]] + ((v[2][t] + u[2][t] - v[1][t] - u[1][t])/r[[2]]))/c[[1]],
      繋ぐ
        u[1]'[t] == ((Vcist[t] - u[1][t])/R[[1]] + ((u[2][t] - u[1][t])/R[[2]]) + c[[1]] v[1]'[t])/d[[1]],
        Vcist'[t] == ((en[t] - Vcist[t])/R0 + (Vob[t] - Vcist[t])/Rout + (u[1][t] - Vcist[t])/R[[1]])/Dcist},
    Table[
    リストを作成
      v[i]'[t] == ((v[i - 1][t] + u[i - 1][t] - v[i][t] - u[i][t])/r[[i]] + ((v[i + 1][t] + u[i + 1][t] - v[i][t] - u[i][t])/r[[i + 1]]))/
```

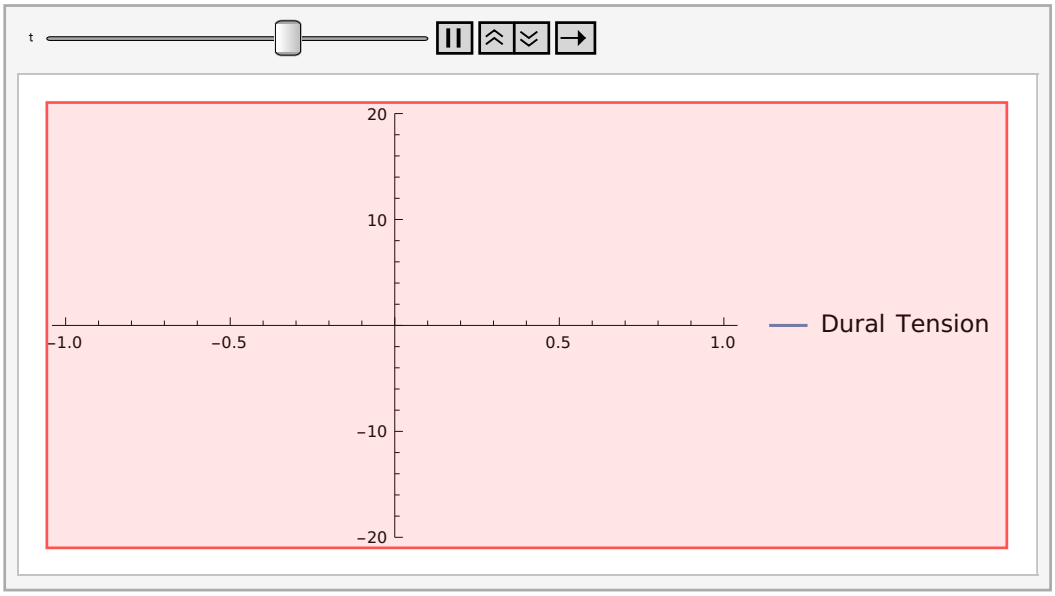
```
    c[[i]], {i, 2, n - 2}],
  Table[u[i]'[t] == ((u[i - 1][t] - u[i][t])/R[[i]] + (u[i + 1][t] - u[i][t])/R[[i + 1]] + c[[i]] v[i]'[t])/d[[i]],
   {i, 2, k - 2}],
  {u[k - 1]'[t] == (c[[k - 1]] v[k - 1]'[t] - (u[k - 1][t] - u[k - 2][t])/R[[k - 1]] -
      diod[u[k - 1][t] - u[k][t], R[[k]], dx])/d[[k - 1]],
   u[k]'[t] == (c[[k]] v[k]'[t] + diod[u[k - 1][t] - u[k][t], R[[k]], dx] - (u[k][t] - u[k + 1][t])/R[[k + 1]])/
    d[[k]]},
  Table[u[i]'[t] == ((u[i - 1][t] - u[i][t])/R[[i]] + (u[i + 1][t] - u[i][t])/R[[i + 1]] + c[[i]] v[i]'[t])/d[[i]],
   {i, k + 1, n - 1}],
    {v[n - 1]'[t] == ((v[n - 2][t] + u[n - 2][t] - v[n - 1][t] - u[n - 1][t])/r[[n - 1]] +
     (u[n][t] - v[n - 1][t] - u[n - 1][t])/r[[n]])/c[[n - 1]],
      u[n]'[t] == ((v[n - 1][t] + u[n - 1][t] - u[n][t])/r[[n]] + (u[n - 1][t] - u[n][t])/R[[n]])/d[[n]]},
    {Vcist[0] == 100},
    Table[v[i][0] == initialCanalPressure , {i, 1, n - 1}],
    Table[u[i][0] == subaraP , {i, 1, n}]
      ];
funcs = Join[{Vcist}, Table[v[i], {i, 1, n - 1}], Table[u[i], {i, 1, n}]];
sol = NDSolve[eqn, funcs , {t, 0, 20}];
DuralTension [i_][t_] := First[u[i][t] /. sol];
SyrinxTension [i_][t_] := First[v[i][t] /. sol];
(* SubarachnoidFlow [i_][t_]:=First[(u[i][t]-u[i+1][t])/R[[i]]/.sol]; *)
SubarachnoidFlow [i_][t_] := Which[
  i == 1, First[(Vcist[t] - u[1][t])/R[[1]] /. sol],
  i == k, First[diod[u[k - 1][t] - u[k][t], R[[k]], dx] /. sol],
  True , First[(u[i - 1][t] - u[i][t])/R[[i]] /. sol]
```

```
  ];
SyrinxFlow[i_][t_] := First[(v[i][t] + u[i][t] - v[i + 1][t] - u[i + 1][t]) / r[[i]] /. sol];
CanalFlow[i_][t_] := Which[
    i == 1, First[ (Vob[t] - u[1][t] - v[1][t]) / r[[1]] /. sol],
    i == n, First[ (u[n - 1][t] + v[n - 1][t] - u[n][t]) / r[[n]] /. sol],
    True, First[ (u[i - 1][t] + v[i - 1][t] - u[i][t] - v[i][t]) / r[[i]] /. sol]
  ];
AbsCanalPressure[i_][t_] := First[(u[i][t] + v[i][t]) /. sol];
```

In[ ]:=

```
canalWave = Animate[ListLinePlot[
    {
     Table[{i, DuralTension[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
     Table[{i, STFactor * SyrinxTension[i][t] / convcmH2O}, {i, 1, n - 1}],
     (*syrinx tension*)
     Table[{i, SFFactor * SubarachnoidFlow[i][t] * 10^6}, {i, 1, n - 1}],
     (*subarachnoid flow*)
     Table[{i, CFFactor * CanalFlow[i][t] * 10^6}, {i, 1, n}] (* Central Canal Flow *)
    },
    range1 = 20;
    PlotRange → {-range1, range1}, PlotLegends → {"Dural Tension", "Channel Tension",
      "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20}, AnimationRate → 0.5]
```

Out[ ]=



In[ ]:=
```
(SubarachnoidFlow[50][5.4] /. sol) * 10 ^ 6
```

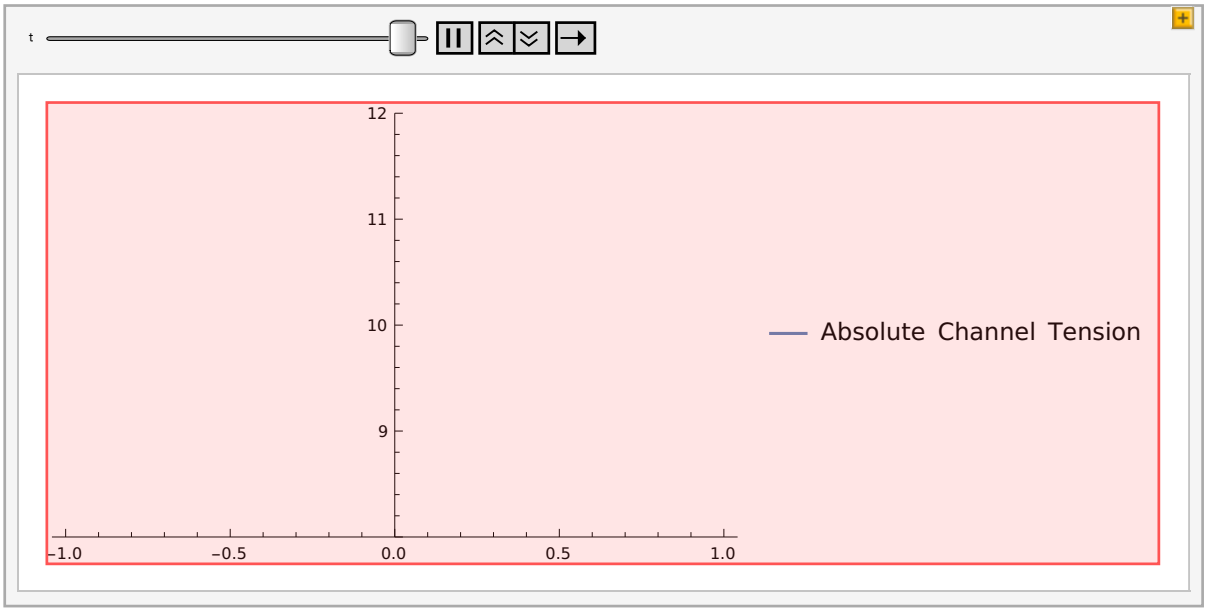
Out[ ]=
```
{1529.85}
```

In[ ]:=
```
10 * convmmhg / R[[5]]
```

Out[ ]=
```
0.194008
```

In[ ]:=
```
absoluteCanaTensionWaveOneway  = Animate[ListLinePlot[
```
アニ…　折れ線グラフ（点を繋いでプロット）
```
  {
   Table[{i, (DuralTension[i][t] + SyrinxTension[i][t]) / convcmH2O}, {i, 1, n - 1}]
```
リストを作成
```
   (* Absolute Canal Tension *)
  },
  range1 = 20;
  PlotRange → {8, 12}, PlotLegends → {"Absolute Channel Tension"}],
```
プロット範囲　プロットの凡例
```
 {t, 0, 20}, AnimationRate → 0.5]
```
アニメーション速度

Out[ ]=



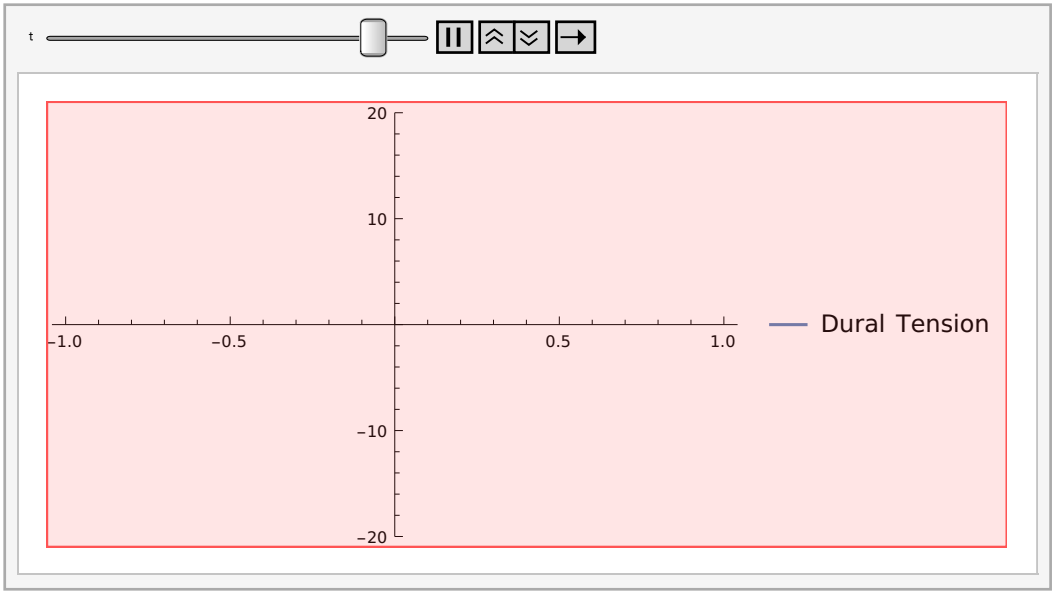
Normal Flow Analysis

In[ ]:=
```
eqnNorm = Join[{v[1]'[t] == (Vob[t] - v[1][t] - u[1][t])/r[[1]] + (v[2][t] + u[2][t] - v[1][t] - u[1][t])/r[[2]])/c[[1]],
    u[1]'[t] == (Vcist[t] - u[1][t])/R[[1]] + (u[2][t] - u[1][t])/R[[2]] + c[[1]] * v[1]'[t])/d[[1]],
    Vcist'[t] == (en[t] - Vcist[t])/R0 + (Vob[t] - Vcist[t])/Rout + (u[1][t] - Vcist[t])/R[[1]])/Dcist},
  Table[
   v[i]'[t] == ((v[i - 1][t] + u[i - 1][t] - v[i][t] - u[i][t])/r[[i]] + (v[i + 1][t] + u[i + 1][t] - v[i][t] - u[i][t])/r[[i + 1]])/
    c[[i]], {i, 2, n - 2}],
  Table[u[i]'[t] == ((u[i - 1][t] - u[i][t])/R[[i]] + (u[i + 1][t] - u[i][t])/R[[i + 1]] + c[[i]] * v[i]'[t])/d[[i]],
   {i, 2, n - 1}],
  {v[n - 1]'[t] == ((v[n - 2][t] + u[n - 2][t] - v[n - 1][t] - u[n - 1][t])/r[[n - 1]] +
     (u[n][t] - v[n - 1][t] - u[n - 1][t])/r[[n]])/c[[n - 1]],
    u[n]'[t] == ((v[n - 1][t] + u[n - 1][t] - u[n][t])/r[[n]] + (u[n - 1][t] - u[n][t])/R[[n]])/d[[n]]},
  {Vcist[0] == 100},
  Table[v[i][0] == initialCanalPressure , {i, 1, n - 1}],
```

```
        Table[u[i][0] == subaraP , {i, 1, n}]
        リストを作成
            ];
funcs = Join[{Vcist}, Table[v[i], {i, 1, n - 1}], Table[u[i], {i, 1, n}]];
        繋ぐ          リストを作成                  リストを作成
solNorm = NDSolve[eqnNorm , funcs , {t, 0, 20}];
          微分方程式の数値解
DuralTensionNorm [i_][t_] := First[u[i][t] /. solNorm];
                             最初
SyrinxTensionNorm [i_][t_] := First[v[i][t] /. solNorm];
                              最初
SubarachnoidFlowNorm [i_][t_] := First[(u[i][t] - u[i + 1][t]) / R[[i]] /. solNorm];
                                 最初
SyrinxFlowNorm [i_][t_] := First[(v[i][t] + u[i][t] - v[i + 1][t] - u[i + 1][t]) / r[[i]] /. solNorm];
                           最初
CanalFlowNorm [i_][t_] :=  Which[
                           最初の真
  i == 1, First[ (Vob[t] - u[1][t] - v[1][t]) /. solNorm],
          最初          r[[1]]

  i == n, First[ (u[n - 1][t] + v[n - 1][t] - u[n][t]) /. solNorm],
          最初          r[[n]]

  True , First[ (u[i - 1][t] + v[i - 1][t] - u[i][t] - v[i][t]) /. solNorm]
  真      最初          r[[i]]

 ]
```

In[ ]:=
```
canalWave = Animate[ListLinePlot[
                    アニ…   折れ線グラフ（点を繋いでプロット）
    {
     Table[{i, DuralTensionNorm[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
     リストを作成
     Table[{i, STFactor * SyrinxTensionNorm[i][t] / convcmH2O}, {i, 1, n - 1}],
     リストを作成
     (*syrinx tension*)
     Table[{i, SFFactor * SubarachnoidFlowNorm[i][t] * 10^6}, {i, 1, n - 1}],
     リストを作成
     (*subarachnoid flow*)
     Table[{i, CFFactor * CanalFlowNorm[i][t] * 10^6}, {i, 1, n}] (* Central Canal Flow *)
     リストを作成
    },
    range1 = 20;
    PlotRange → {-range1, range1}, PlotLegends → {"Dural Tension", "Channel Tension",
    プロット範囲                    プロットの凡例
      "Subarachnoid Flow", "Channel Flow"}, {t, 0, 20}, AnimationRate → 0.5]
                                                    アニメーション速度
```

Out[ ]=



Simple Block Analysis

In[ ]:=
```
Rsb = Join[Table[rs, k - 1], {20 rs}, Table[rs, n - k]];
      繋ぐ  リストを作成          リストを作成
```

$$\text{eqnsb} = \text{Join}\left[\left\{\text{v[1]}'[t] == \left(\frac{\text{Vob}[t] - \text{v[1]}[t] - \text{u[1]}[t]}{\text{r[[1]]}} + \frac{(\text{v[2]}[t] + \text{u[2]}[t] - \text{v[1]}[t] - \text{u[1]}[t])}{\text{r[[2]]}}\right) \Big/ \text{c[[1]]},\right.\right.$$

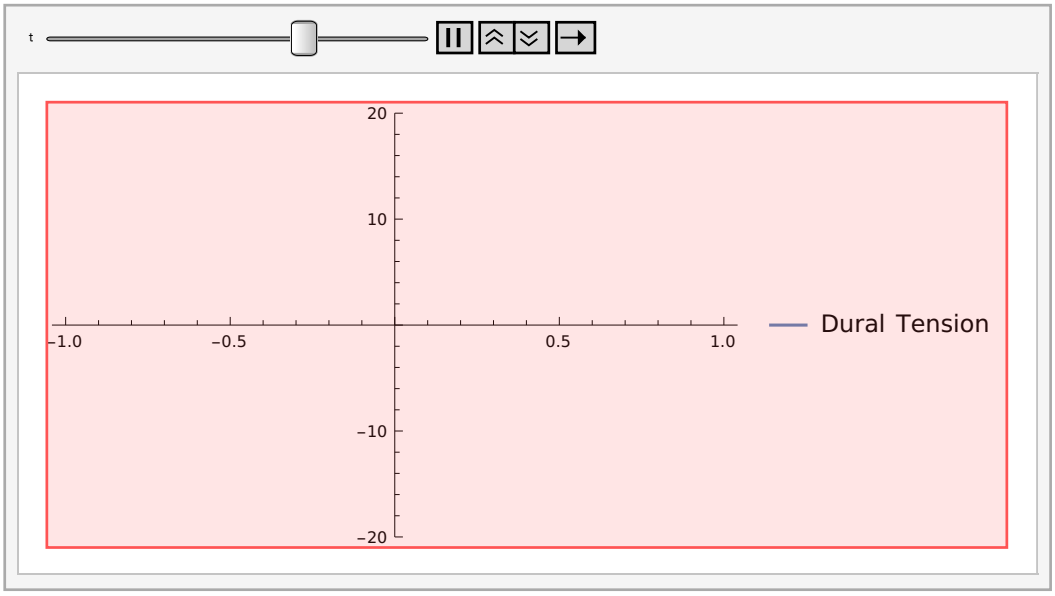
繋ぐ

```
    u[1]'[t] == ((Vcist[t] - u[1][t])/Rsb[[1]] + (u[2][t] - u[1][t])/Rsb[[2]] + c[[1]] * v[1]'[t])/d[[1]],
    Vcist'[t] == ((en[t] - Vcist[t])/R0 + (Vob[t] - Vcist[t])/Rout + (u[1][t] - Vcist[t])/Rsb[[1]])/Dcist},

  Table[
  リストを作成
    v[i]'[t] == ((v[i - 1][t] + u[i - 1][t] - v[i][t] - u[i][t])/r[[i]] + (v[i + 1][t] + u[i + 1][t] - v[i][t] - u[i][t])/r[[i + 1]])/
      c[[i]], {i, 2, n - 2}],
  Table[u[i]'[t] == ((u[i - 1][t] - u[i][t])/Rsb[[i]] + (u[i + 1][t] - u[i][t])/Rsb[[i + 1]] + c[[i]] * v[i]'[t])/d[[i]],
  リストを作成
    {i, 2, n - 1}],
      {v[n - 1]'[t] == ((v[n - 2][t] + u[n - 2][t] - v[n - 1][t] - u[n - 1][t])/r[[n - 1]] +
       (u[n][t] - v[n - 1][t] - u[n - 1][t])/r[[n]])/c[[n - 1]],
           u[n]'[t] == ((v[n - 1][t] + u[n - 1][t] - u[n][t])/r[[n]] + (u[n - 1][t] - u[n][t])/Rsb[[n]])/
      d[[n]]},
      {Vcist[0] == 100},
      Table[v[i][0] == initialCanalPressure , {i, 1, n - 1}],
      リストを作成
      Table[u[i][0] == subaraP , {i, 1, n}]
      リストを作成
          ];
funcs = Join[{Vcist}, Table[v[i], {i, 1, n - 1}], Table[u[i], {i, 1, n}]];
        繋ぐ          リストを作成                  リストを作成
solsb = NDSolve[eqnsb , funcs , {t, 0, 20}];
        微分方程式の数値解
DuralTensionSb [i_][t_] := First[u[i][t] /. solsb];
                           最初
SyrinxTensionSb [i_][t_] := First[v[i][t] /. solsb];
                            最初
SubarachnoidFlowSb [i_][t_] := First[(u[i][t] - u[i + 1][t])/Rsb[[i + 1]] /. solsb];
                               最初
SyrinxFlowSb [i_][t_] := First[(v[i][t] + u[i][t] - v[i + 1][t] - u[i + 1][t])/r[[i]] /. solsb];
                         最初
CanalFlowSb [i_][t_] :=  Which[
                         最初の真
  i == 1, First[(Vob[t] - u[1][t] - v[1][t])/r[[1]] /. solsb],
          最初
```

```
    i == n, First[(u[n - 1][t] + v[n - 1][t] - u[n][t]) / r[[n]] /. solsb],
                  最初

    True, First[(u[i - 1][t] + v[i - 1][t] - u[i][t] - v[i][t]) / r[[i]] /. solsb]
    真    最初

  ]
```

In[ ]:=
```
simpleBlockWave = Animate[ListLinePlot[
                  アニ…    折れ線グラフ（点を繋いでプロット）
  {
   Table[{i, DuralTensionSb[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
   リストを作成
   Table[{i, STFactor * SyrinxTensionSb[i][t] / convcmH2O}, {i, 1, n - 1}],
   リストを作成
   (*syrinx tension*)
   Table[{i, SFFactor * SubarachnoidFlowSb[i][t] * 10^6}, {i, 1, n - 1}],
   リストを作成
   (*subarachnoid flow*)
   Table[{i, CFFactor * CanalFlowSb[i][t] * 10^6}, {i, 1, n}] (* Central Canal Flow *)
   リストを作成
  },
  range1 = 20;
  PlotRange → {-range1, range1}, PlotLegends → {"Dural Tension", "Channel Tension",
  プロット範囲                   プロットの凡例
    "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20}, AnimationRate → 0.5]
                                                       アニメーション速度
```

Out[ ]=



Rendering of Movies

```
In[ ]:= renderRate = 0.02;
        range1 = 20;
```

One - way valve

```
In[ ]:= totalAnimationOneway = Table[ListLinePlot[
                                リ…  折れ線グラフ（点を繋いでプロット）
          {
           Table[{i, DuralTension[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
           リストを作成
           Table[{i, STFactor * SyrinxTension[i][t] / convcmH2O}, {i, 1, n - 1}],
           リストを作成
           (*syrinx tension*)
           Table[{i, SFFactor * SubarachnoidFlow[i][t] * 10^6}, {i, 1, n - 1}],
           リストを作成
           (*subarachnoid flow*)
           Table[{i, CFFactor * CanalFlow[i][t] * 10^6}, {i, 1, n}] (* Central Canal Flow *)
           リストを作成
          },
          range1 = 20;
          PlotRange → {-range1, range1},
          プロット範囲
          PlotLegends → {"Dural Tension", "Channel Tension",
          プロットの凡例
            "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20, renderRate}];
```

```
In[ ]:= Export[
        エキスポート
          "/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/totalAnimation
            .mp4", totalAnimationOneway ];
```

```
... General :
制限付きのFFmpegを使用しています . より完全なコーデックサポートのためにはFFmpegをインストールしてください .
```

```
In[ ]:= Export[
        エキスポート
          "/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/totalAnimation .gif",
          totalAnimationOneway ];
```

In[ ]:=
```
canalFlowOneway = Table[ListLinePlot[
                  リ…  折れ線グラフ（点を繋いでプロット）
    {
     Table[{i, CFFactor * CanalFlow[i][t] * 10^6}, {i, 1, n}]
     リストを作成
    },
    range1 = 20;
    PlotRange → {-range1, range1}, PlotLegends → {"Channel  Flow"}],
    プロット範囲                    プロットの凡例
   {t, 0, 20, renderRate}];
```

In[ ]:=
```
Export[
エキスポート
 "/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/canalFlowOneway
   .mp4", canalFlowOneway]
```

Out[ ]=
```
/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/canalFlowOneway .
  mp4
```

```
"/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/canalFlowOneway .
  mp4"
```

In[ ]:=
```
canalFlowComparison = Table[ListLinePlot[
                      リ…  折れ線グラフ（点を繋いでプロット）
    {
     Table[{i, CFFactor * CanalFlow[i][t] * 10^6}, {i, 1, n}],
     リストを作成
     Table[{i, CFFactor * CanalFlowSb[i][t] * 10^6}, {i, 1, n}]
     リストを作成
    },
    range1 = 20;
    PlotRange → {-range1, range1}, PlotLegends → {"Channel  Flow (One-way  Valve)",
    プロット範囲                    プロットの凡例
      "Channel  Flow (Simple  Block)"}], {t, 0, 20, renderRate}];
                                 ブロック
```

In[ ]:=
```
Export["/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
エキスポート
    canalFlowComparison .mp4", canalFlowComparison]
```

Out[ ]=
```
/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
  canalFlowComparison .mp4
```

```
In[ ]:= syrinxTensionOneway = Table[ListLinePlot[
                                リ…  折れ線グラフ（点を繋いでプロット）
        {
         Table[{i, STFactor * SyrinxTension[i][t] / convcmH2O}, {i, 1, n - 1}]
         リストを作成
        },
        PlotRange → {-10, 10}], {t, 0, 20, renderRate}];
        プロット範囲

In[ ]:= Export["/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        エキスポート
        syrinxTensionOneway_10.mp4", syrinxTensionOneway]

Out[ ]= /home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        syrinxTensionOneway_10.mp4
```

Normal

```
In[ ]:= totalAnimationNorm = Table[ListLinePlot[
                                リ…  折れ線グラフ（点を繋いでプロット）
        {
         Table[{i, DuralTensionNorm[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
         リストを作成
         Table[{i, STFactor * SyrinxTensionNorm[i][t] / convcmH2O}, {i, 1, n - 1}],
         リストを作成
         (*syrinx tension*)
         Table[{i, SFFactor * SubarachnoidFlowNorm[i][t] * 10^6}, {i, 1, n - 1}],
         リストを作成
         (*subarachnoid flow*)
         Table[{i, CFFactor * CanalFlowNorm[i][t] * 10^6}, {i, 1, n}]
         リストを作成
         (* Central Canal Flow *)
        },
        range1 = 20;
        PlotRange → {-range1, range1},
        プロット範囲
        PlotLegends → {"Dural Tension", "Channel Tension",
        プロットの凡例
          "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20, renderRate}];

In[ ]:= Export["/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        エキスポート
        totalAnimationNorm.mp4", totalAnimationNorm]

Out[ ]= /home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        totalAnimationNorm.mp4
```

```
In[ ]:= syrinxTensionNorm = Table[ListLinePlot[
                              リ…  折れ線グラフ（点を繋いでプロット）
        {
         Table[{i, STFactor * SyrinxTensionNorm[i][t] / convcmH2O}, {i, 1, n - 1}]
         リストを作成
         (*syrinx tension*)
        },
        range1 = 20;
        PlotRange → {-range1, range1},
        プロット範囲
        PlotLegends → {"Dural Tension", "Channel Tension",
        プロットの凡例
          "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20, renderRate}];

In[ ]:= Export["/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        エキスポート
    syrinxTensionNorm.mp4", syrinxTensionNorm]

    ••• General :
    制限付きのFFmpegを使用しています . より完全なコーデックサポートのためにはFFmpegをインストールしてください .

Out[ ]= /home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/syrinxTensionNorm
        .mp4
```

Simple Block

```
In[ ]:= totalAnimationSb = Table[ListLinePlot[
                              リ…  折れ線グラフ（点を繋いでプロット）
        {
         Table[{i, DuralTensionSb[i][t] / convcmH2O}, {i, 1, n - 1}],    (* Dural Tension *)
         リストを作成
         Table[{i, STFactor * SyrinxTensionSb[i][t] / convcmH2O}, {i, 1, n - 1}],
         リストを作成
         (*syrinx tension*)
         Table[{i, SFFactor * SubarachnoidFlowSb[i][t] * 10^6}, {i, 1, n - 1}],
         リストを作成
         (*subarachnoid flow*)
         Table[{i, CFFactor * CanalFlowSb[i][t] * 10^6}, {i, 1, n}] (* Central Canal Flow *)
         リストを作成
        },
        range1 = 20;
        PlotRange → {-range1, range1},
        プロット範囲
        PlotLegends → {"Dural Tension", "Channel Tension",
        プロットの凡例
          "Subarachnoid Flow", "Channel Flow"}], {t, 0, 20, renderRate}];
```

```
In[ ]:= Export["/home/chang/Dropbox/Projects/MRI_flow/mathematica/new_analysis/images/
        totalAnimationSb.mp4", totalAnimationSb];
```

Only Canal Flow

```
In[ ]:= canalFlowOneWay = Table[ListLinePlot[
          {
           Table[{i, CFFactor * CanalFlow[i][t]}, {i, 1, n - 1}]
          },
          range1 = 200;
          PlotRange → {-range1, range1},
          PlotLegends → {"Canal Flow"}
         ], {t, 0, 20, renderRate}];
```

Table : 反復演算 {t, 0, 20, renderRate }は適正な範囲を持ちません .

```
In[ ]:= Export[
          "/home/chang/Dropbox/Projects/MRI_flow/mathematica/concrete/canalFlow.mp4", canalFlow]
```

Export : canalFlow は映像フレームに変換することはできません .

```
Out[ ]= $Failed
```

```
In[ ]:= 10 / 4000 * 100 * 100 * 100
```

```
Out[ ]= 2500
```