```
> restart, with(LinearAlgebra);
restart, [`&x`, Add, Adjoint, BackwardSubstitute, BandMatrix,
Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CARE,
CharacteristicMatrix, CharacteristicPolynomial, Column,
ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix,
CompressedSparseForm, ConditionNumber, ConstantMatrix,
ConstantVector, Copy, CreatePermutation, CrossProduct, DARE,
DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix,
Dimension, Dimensions, DotProduct, EigenConditionNumbers,
Eigenvalues, Eigenvectors, Equal, ForwardSubstitute,
FrobeniusForm, FromCompressedSparseForm, FromSplitForm,
GaussianElimination, GenerateEquations, GenerateMatrix,
Generic, GetResultDataType, GetResultShape,
GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm,
HermitianTranspose, HessenbergForm, HilbertMatrix,
HouseholderMatrix, IdentityMatrix, IntersectionBasis,
IsDefinite, IsOrthogonal, IsSimilar, IsUnitary,
JordanBlockMatrix, JordanForm, KroneckerProduct, LA_Main,
LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map,
Map2, MatrixAdd, MatrixExponential, MatrixFunction,
MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower,
MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial,
Minor, Modular, Multiply, NoUserValue, Norm, Normalize,
NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm,
ProjectionMatrix, QRDecomposition, RandomMatrix, RandomVector,
Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row,
RowDimension, RowOperation, RowSpace, ScalarMatrix,
ScalarMultiply, ScalarVector, SchurForm, SingularValues,
SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix,
SubVector, SumBasis, SylvesterMatrix, SylvesterSolve,
ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector,
VandermondeMatrix, VectorAdd, VectorAngle,
VectorMatrixMultiply, VectorNorm, VectorScalarMultiply,
ZeroMatrix, ZeroVector, Zip]
> h_l:=2;
  h_r:=1;
  u_l:=0;
  u_r:=0;
  g :=10;

  u_m_1 := u_l+ 2*(sqrt(g*h_l)-sqrt(g*h_m));
  u_m_2 := u_r + (h_m -h_r) * sqrt(g/2*(1/h_m+1/h_r));


  solutions := evalf(eval(solve(u_m_1 = u_m_2, h_m)));
h_l := 2
h_r := 1
```

```
u_l := 0
u_r := 0
g := 10
u_m_1 := 4*5^(1/2)-2*10^(1/2)*h_m^(1/2)
u_m_2 := (h_m-1)*(5/h_m+5)^(1/2)
solutions := 1.45384089578493
```

```
> h_m_sol := solutions;
h_m_sol := 1.45384089578493
```

```
> u_m_sol := evalf(eval(u_m_1,h_m=h_m_sol));
u_m_sol := 1.31841878685976
```

```
> lambda_1:= u - sqrt(g*h);
  lambda_2:= u + sqrt(g*h);
lambda_1 := u-10^(1/2)*h^(1/2)
lambda_2 := u+10^(1/2)*h^(1/2)
```

```
> lamaba_1_l := evalf(subs([h=h_l,u=u_l],lambda_1));
  lamaba_1_r := evalf(subs([h=h_r,u=u_r],lambda_1));
  lamaba_1_m := evalf(subs([h=h_m_sol,u=u_m_sol],lambda_1));
  lamaba_2_l := evalf(subs([h=h_l,u=u_l],lambda_2));
  lamaba_2_r := evalf(subs([h=h_r,u=u_r],lambda_2));
  lamaba_2_m := evalf(subs([h=h_m_sol,u=u_m_sol],lambda_2));
lamaba_1_l := -4.472135954
lamaba_1_r := -3.162277660
lamaba_1_m := -2.49450777371037
lamaba_2_l := 4.472135954
lamaba_2_r := 3.162277660
lamaba_2_m := 5.13134534742988
```

```
> hu_m := h_m_sol * u_m_sol;
  hu_r := h_r * u_r;
  hu_l := h_l * u_l;
  s := (hu_m - hu_r)/(h_m_sol-h_r);
hu_m := 1.91677115010787
hu_r := 0
hu_l := 0
s := 4.22344299050607
```

```
> A:= u_l+2*sqrt(g*h_l);
  h_tilde := 1/(9*g)*(A-xi)^2;
A := 4*5^(1/2)
h_tilde := (1/90)*(4*5^(1/2)-xi)^2
```

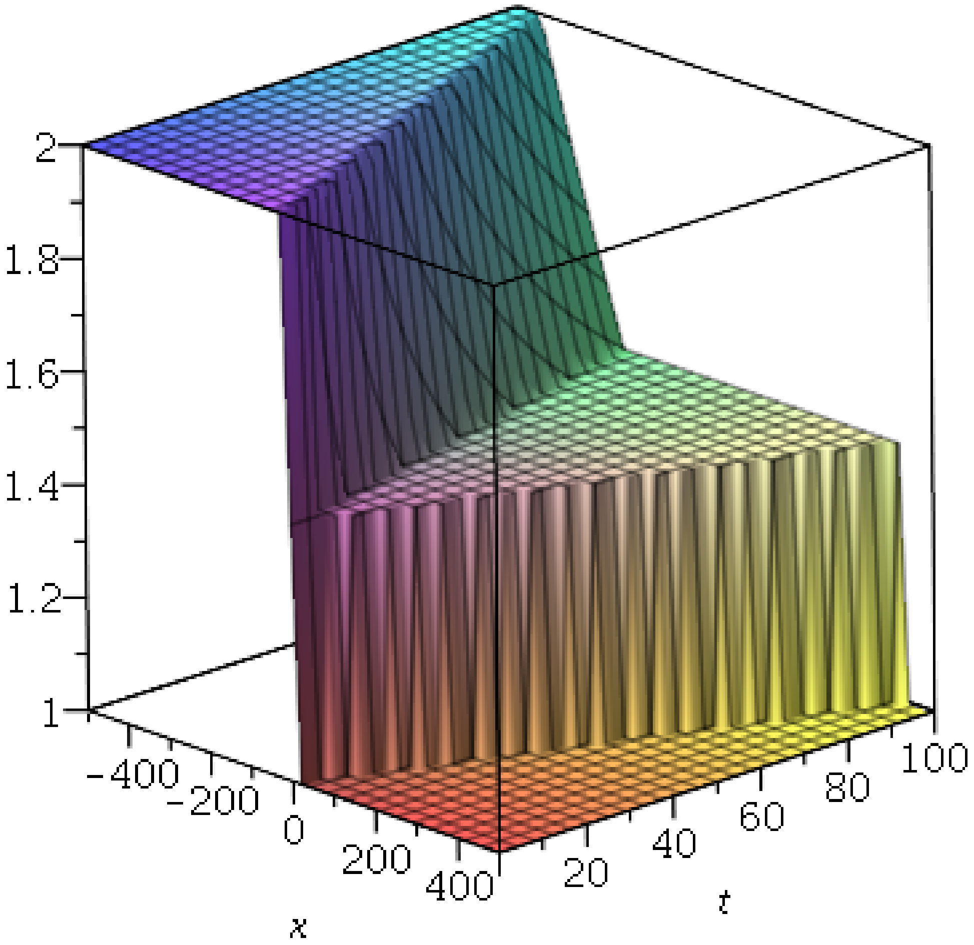
```
> solution_h_proc := proc(x,t)
   if x/t <= lamaba_1_l then
   return h_l;
   elif x/t >= s then
    return h_r;
   elif x/t > lamaba_1_l and x/t < lamaba_1_m then
    return eval(h_tilde,xi=x/t);
   else
```

```
    return h_m_sol;
   end if;
 end proc;

 solution_h := unapply(solution_h_proc,x,t);

 plot3d('solution_h_proc(x,t)',x=-500..500,t=0.0001..100);
```

```
solution_h_proc := proc (x, t) if x/t <= lamaba_1_l then return
h_l elif s <= x/t then return h_r elif lamaba_1_l < x/t and x/t
< lamaba_1_m then return eval(h_tilde, xi = x/t) else return
h_m_sol end if end proc
solution_h := proc (x, t) options operator, arrow;
solution_h_proc end proc
```



2
1.8
1.6
1.4
1.2
1
-400
-200
0
200
400
x
20
40
60
80
100
t