```
> with(LinearAlgebra) :
> phi[1] := (xi, h) →piecewise(xi ≥ 0 and xi ≤ h, 1 − xi/h) :
> phi[2] := (xi, h) →piecewise(xi ≥ 0 and xi ≤ h, xi/h) :
>
> RigidezLocal := proc(h); local i, j, M;
   description "obtém a matriz de rigidez local";
   M := Matrix(2, 2);
   for i from 1 to 2 do
   for j from 1 to 2 do
> M[i, j] := int(diff(phi[i](xi, h), xi)·diff(phi[j](xi, h), xi), xi = 0 ..h);
> end do;
> end do;
> return M;
> end proc:
>
> Conectividade :=proc(E); local e, M;
> description "obtém a matriz de conectividade da malha";
> M := Matrix(E, 2);
> for e from 1 to E do M[e, 1] := e; M[e, 2] := e + 1; end do;
> return M;
> end proc:
>
> RigidezGlobal :=proc(E, L); local e, i, j, B, K, M;
> description "obtém a matriz de rigidez global";
> B := Conectividade(E);
> M := Matrix(E + 1, E + 1);
> K := Matrix(2, 2);
> for e from 1 to E do
> K := RigidezLocal(L/E);
> for i from 1 to 2 do
> for j from 1 to 2 do
> M[B[e, i], B[e, j]] := M[B[e, i], B[e, j]] + K[i, j];
> end do
> end do
> end do
> return M;
> end proc:
>
> CargaLocal := proc(h); local M;
   description "obtém o vetor de carga local";
   M := Matrix(2, 1);
> M[1, 1] := int(2·phi[1](xi, h), xi = 0 ..h); M[2, 1] := int(2·phi[2](xi, h), xi = 0 ..h);
> return M;
> end proc:
>
> CargaGlobal := proc(E, L); local e, B, F, M;
   description "obtém o vetor de carga global";
```

```
 M := Matrix(E + 1, 1);
> F := Matrix(2, 1);
> B := Conectividade(E);
> for e from 1 to E do
> F := CargaLocal(L/E);
> M[B[e, 1], 1] := M[B[e, 1], 1] + F[1, 1];
> M[B[e, 2], 1] := M[B[e, 2], 1] + F[2, 1];
> end do
> return M;
> end proc:
>
> RearranjaRigidez := proc(E, K); local i, j, M;
>  description "rearranja a matriz de rigidez global";
> M := Matrix(E − 1, E − 1);
> for i from 1 to E − 1 do
> for j from 1 to E − 1 do
> M[i, j] := K[i + 1, j + 1];
> end do
> end do
> return M;
> end proc:
>
> RearranjaCarga := proc(E, F); local i, M;
>  description "rearranja a matriz de carga global";
> M := Matrix(E − 1, 1);
> for i from 1 to E − 1 do
> M[i, 1] := F[i + 1, 1];
> end do
> return M;
> end proc:
>
> Solucao := proc(E, L); local  i, K, M, F, N, U, P;
> description "resolve";
> N := CargaGlobal(E, L);
> F := RearranjaCarga(E, N);
> M := RigidezGlobal(E, L);
> K := RearranjaRigidez(E, M);
> P := MatrixMatrixMultiply(MatrixInverse(K), F);
> U := Matrix(E + 1, 1);
> for i from 2 to E do U[i, 1] := P[i − 1, 1]; end do;
> return U;
> end proc:
>
> Erro0 := proc(E, L); local e, h, U, Norma ;
> description "erro da solução";
> U := Solucao(E, L);
> h := E/L; Norma := 0;
> for e from 1 to E do
```

```
> Norma := Norma + int( (U[e, 1]·phi[1](x − (e − 1)·h, h) + U[e + 1, 1]·phi[2](x − (e
        − 1)·h, h) -x·(1 − x) )^2, x = 0 ..L);
> end do;
> return sqrt(Norma);
> end proc:
>
> evalf(Erro0(1, 1));
```

$$0.1825741858 \quad (1)$$

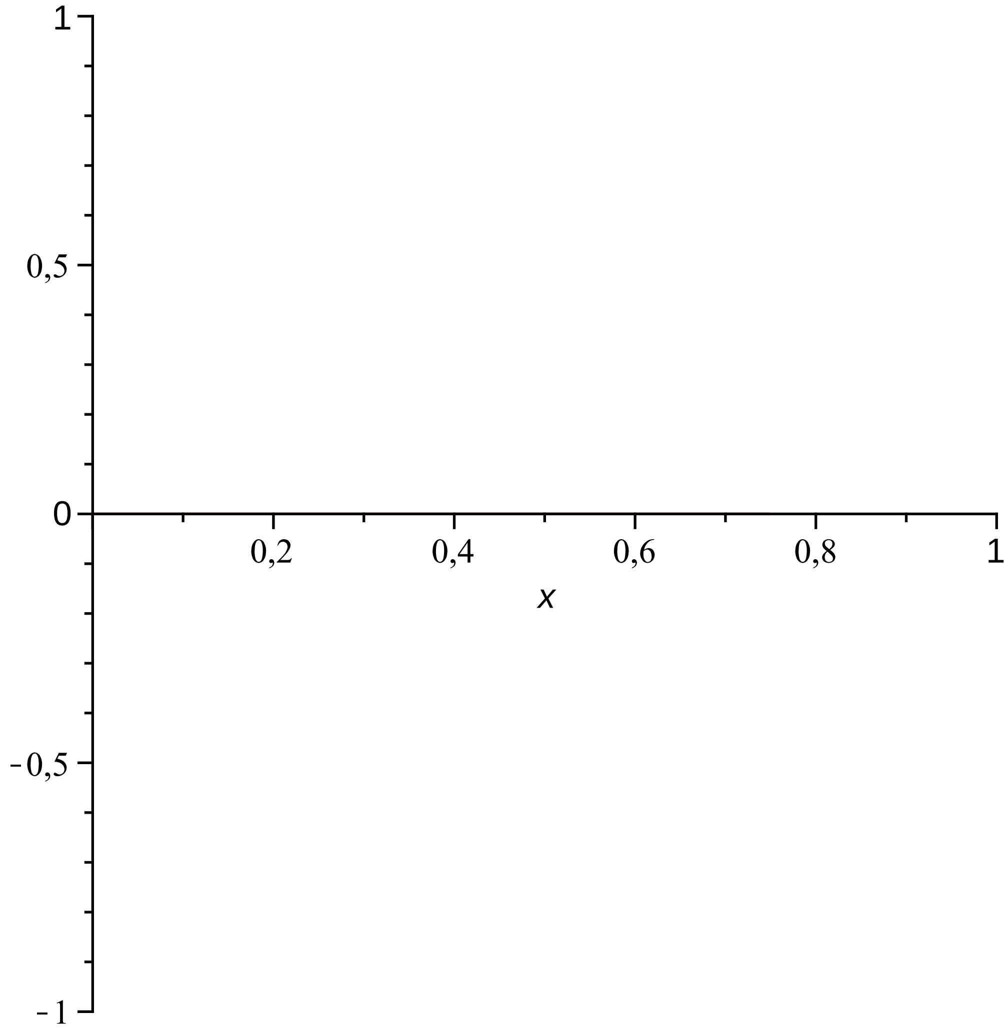
```
>
> int( (U[e, 1]·phi[1](xi, h) + U[e + 1, 1]·phi[2](xi, h) - ((xi + (e − 1)·h)·(1 − (xi + (e
        − 1)·h))) )^2, xi = 0 ..h) :
> evalf(sqrt(int( (x·(1 − x))^2, x = 0 ..1)))
```

$$0.1825741858 \quad (2)$$

```
> u := U[4, 1]·phi[1](x − 3/4, 1/4) + U[5, 1]·phi[2](x − 3/4, 1/4);
```

$$u := U_{4,1}\left(\begin{cases} 4-4x & 0 \le x-\frac{3}{4} \textbf{ and } x \le 1 \\ 0 & otherwise \end{cases}\right) + U_{5,1}\left(\begin{cases} 4x-3 & 0 \le x-\frac{3}{4} \textbf{ and } x \le 1 \\ 0 & otherwise \end{cases}\right) \quad (3)$$

```
> plot([u], x = 0 ..1)
```


1
0,5
0
-0,5
-1
0,2
0,4
0,6
0,8
1
x