Points, Vectors, and Maztrices

```
> with(LinearAlgebra); with(plots);
```

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

[animate, animate3d, animatecurve, arrow, changecoords, complexplot, complexplot3d, conformal, conformal3d, contourplot, contourplot3d, coordplot, coordplot3d, densityplot, display, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d, implicitplot, implicitplot3d, inequal, interactive, interactiveparams, intersectplot, listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported, polyhedraplot, rootlocus, semilogplot, setcolors, setoptions, setoptions3d, spacecurve, sparsematrixplot, surfdata, textplot, textplot3d, tubeplot] **(1)**

Let us inspect (column) vectors.

```
> p := ⟨0, 1⟩; r := ⟨1, 2⟩;
```

$$p := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$r := \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \textbf{(2)}$$

> $p[1];$

$$0 \tag{3}$$

> $r[2];$

$$2 \tag{4}$$

> $l := p + \lambda \cdot r;$

$$l := \begin{bmatrix} \lambda \\ 1 + 2\lambda \end{bmatrix} \tag{5}$$

Now, we want to compute the shortest distance from point q := <2,1> to the line.

> $q := \langle 2, 1 \rangle;$

$$q := \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{6}$$

> $lineplot := plot([l[1], l[2], \lambda = -2..2]);$

$$lineplot := PLOT(\ldots) \tag{7}$$

> $f := \lambda \rightarrow p + \lambda \cdot r;$

$$f := \lambda \rightarrow p + \lambda\, r \tag{8}$$

> $s := seq([l[1], l[2]], \lambda = -2..2);$

$$s := [-2, -3], [-1, -1], [0, 1], [1, 3], [2, 5] \tag{9}$$

> $t := seq\left(\left[f\left(\frac{x}{10}\right)[1], f\left(\frac{x}{10}\right)[2]\right], x = -20..20\right):$

> $pointline := pointplot([s]);$

$$pointline := PLOT(\ldots) \tag{10}$$

> $Qplot := pointplot(q);$

$$Qplot := PLOT(\ldots) \tag{11}$$

> $a1 := arrow([0, 0], p, width = [0.075, relative = false], head\_length = [0.4, relative = false], color = green);$

$$a1 := PLOT(\ldots) \tag{12}$$

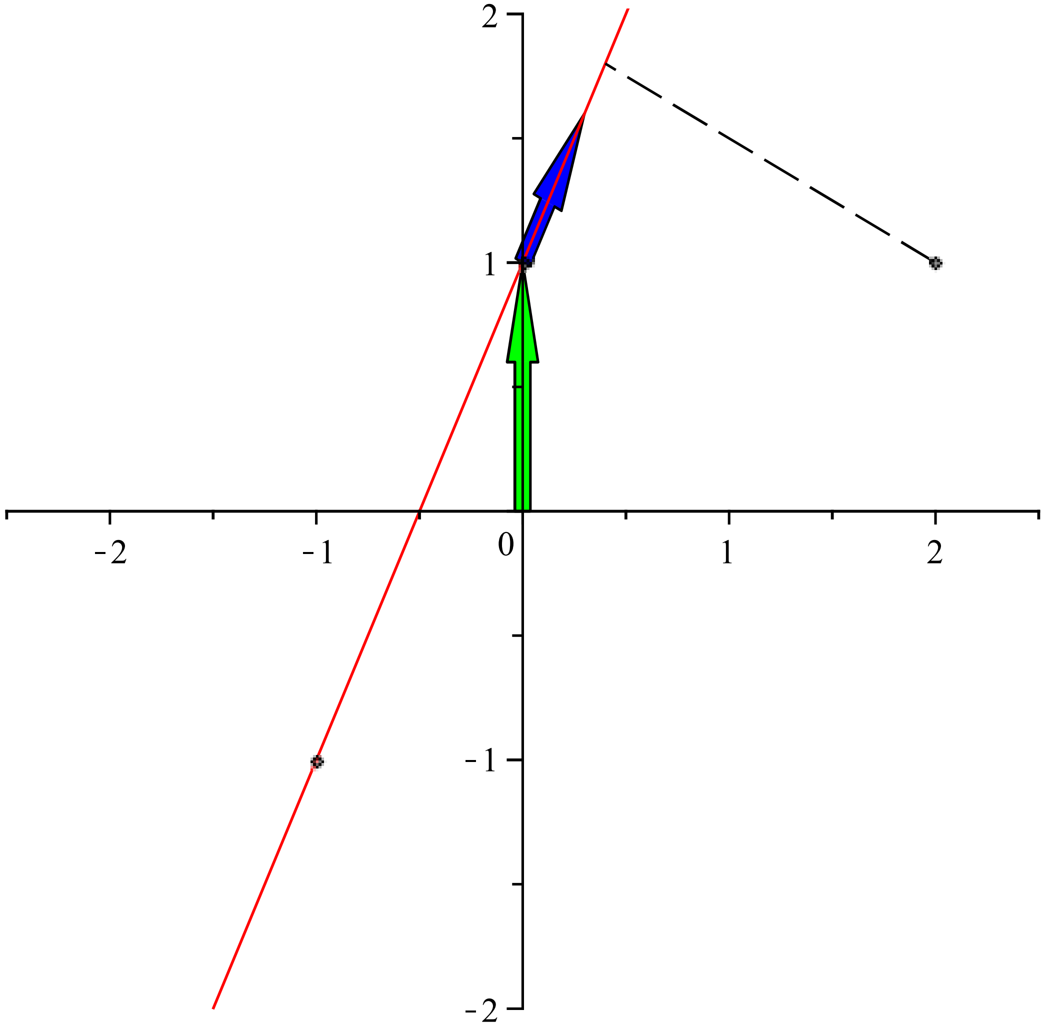
> $a2 := arrow(p, 0.3 \cdot r, width = [0.075, relative = false], head\_length = [0.4, relative = false], color = blue);$

$$a2 := PLOT(\ldots) \tag{13}$$

> $HitPoint := subs\left(\lambda = \frac{DotProduct(q - p, r)}{DotProduct(r, r)}, l\right);$

$$HitPoint := \begin{bmatrix} \frac{2}{5} \\ \frac{9}{5} \end{bmatrix} \tag{14}$$

> $display([a1, a2, lineplot, pointline, Qplot, pointplot([q, HitPoint], connect = true, thickness = 1, linestyle = dash)], view = [-2.5..2.5, -2..2]);$




```
> 
> myDotProduct := proc(u, v, k)
    local j, res, lu, lv;
   res := 0;
   for j from 1 to k do
     res := res + u[j]·v[j];
   end do;
   #return res;
   lu := convert(u,'list');
   lv := convert(v,'list');
   return sum(lu[i]·lv[i], i = 1..k);
 end proc:
```

$$> \frac{DotProduct(q-p, r)}{DotProduct(r, r)};$$

$$\frac{2}{5} \quad \textbf{(15)}$$

$$> \frac{myDotProduct(q-p, r, 2)}{myDotProduct(r, r, 2)};$$

$$\frac{2}{5} \tag{16}$$

>