▪ Introduction to *Mathematica.* Volume I - Mathematics

download this document from `www.hakenberg.de`

⚠ Please **deactivate NUM-Lock**.
⚠ Proceed by copying the expressions into *Mathematica*. Press [SHIFT][RET] together to **evaluate** the expressions. Vary the input.
⚠ When you recieve weird error messages upon evaluation, although your input seems correct, use the menu item **Kernel** → **Quit-Kernel** to reset the memory of *Mathematica* and start evaluation from the beginning.

First, we perform a **derivation**, define the **value** of a **variable**, a **list** and a **mathematical function** $f : \mathbb{R}^2 \to \mathbb{R}^3$.
💡 use [CTRL]2, [CTRL]-, [CTRL]^ to create expressions as $\sqrt{2}$, x_2, 4^{10} (alternatively use the icons on the palettes at the side)
💡 for greek letters π, α, etc. use sequences like [ESC]pi[ESC], [ESC]a[ESC] 💡 the imaginary unit is [ESC]ii[ESC]

```
12 + √6 - D[Sin[x] x^3, {x, 2}]
myVar = 47!
myList = {7, Sin[π / 3], -√23, 4^10, a, x1, 2 + 4 ⅈ, {2, 99}}
myF[x1, x2] = {(R + r Cos[x2]) Cos[x1], (R + r Cos[x2]) Sin[x1], r Sin[x2]}
myEqs = 3 + x - 4 x^3 == 0      (*[SHIFT][RET]*)
```

Upon **evaluation**, the definitions are **memorized** by *Mathematica*. We work with `myVar`, `myList`, etc. in the sequel. We **solve** for unknowns, convert to **numerical** values and **substitute** a coordinate into the function.
💡 `Re` gives the real part of a complex number 💡 the right arrow → is dumped upon the combination [ESC]->[ESC]
💡 the appearence of /. reminds us of the λ in λ-calculus, /. represents a substitution

```
N[myList]
myF[x1, x2] /. {x1 → 44}
mySol = Solve[myEqs, {x}]
Re[x] /. mySol
```

Explicit math: **Sums**, **integrals**, **derivatives**, **limits**, **Taylor**-expansion.
💡 for ∞ use the sequence [ESC]inf[ESC]

```
{∑_{k=1}^{n} k^2, ∑_{k=1}^{∞} k^-8, ∫Cosh[t] ⅆt, ∫_0^∞ Exp[-λ t] ⅆt}
∂_{x2} myF[x1, x2]
Limit[myVar Sin[x] / x, x → 0]
Series[Cosh[x], {x, 0, 8}]
```

Lets produce some **graphics**. ✍ Why do we substitute values for *R* and *r* in the third example?

```
Plot[{Sin[3 t], t^2, Cosh[t], 4 - t}, {t, 0, 2}]
Plot3D[Abs[Gamma[x + ⅈ y]], {x, -5, 3}, {y, -1, 1}, PlotPoints → 50]
ParametricPlot[{t Cos[2 t], t Sin[t]}, {t, 0, 2 π}]
ParametricPlot3D[Evaluate[myF[x1, x2] /. {R → 2, r → 1}],
  {x1, 0, 2 π - 1.5}, {x2, 0, 2 π - 1}]
ContourPlot[Cos[x y], {x, -5, 5}, {y, -5, 5}, {PlotPoints → 30, ContourLines → False}]
```

Implicit math: **Solve** systems of **non linear equations**, **differential equations**, and **simplification**.
✍ activate the help browser with [SHIFT]F1 and look up commands like `DSolve`.

```
mySol = Solve[{y^2 == x^3 - x + 1, 2 y == -3 x}]
MatrixForm[N[mySol]]
```

```
Eliminate[{x == 3 y^2 + b, y x == b}, {y}]
{Solve[{1 / a == 1 / c}, {a}], Reduce[{1 / a == 1 / c}, {a}]}
myY = DSolve[
  {y1 '[x] == y1[x] + y2[x], y2 '[x] == x Sin[x] + 3 y1[x], y1[0] == α}, {y1[x], y2[x]}, x]
FullSimplify[myY]
Simplify[Log[a b] == Log[a] + Log[b], {0 < a < b}]
ListPlot[{Re[x], Im[x]} /. #1 & /@ NSolve[{x^25 + 7 x^21 - 3 x^12 + 7 x^11 + 4 x^8 - 10 x^5 == 1}, {x}]]
```

Various functions concerning **numbers** and **polynomials**.

```
N[π, 80]
BernoulliB[#1] & /@ Range[18]
FactorInteger[Prime[1000]^2 Prime[2000] Prime[3000]]
LegendreP[4, x]
PolynomialGCD[(1 - x)^2 (1 + x)^2 (2 + x), (1 - x) (2 + x) (3 + x)]
```

Basics in **linear algebra**; determinant, inverse, matrix/vector multiplication, eigensystem.
💡 to enter a matrix, place your cursor between two round brackets, like this: (⁝), then ...

• keep hitting CTRL ENTER to add rows • keep hitting CTRL **,** to add columns.
Use the TAB key to toggle between the empty spots □

```
myM = ( a  b
        c  0 )
myM.{d, e}
{d, e}.myM
myI = Inverse[myM]
MF = MatrixForm
myI // MF
myI.{d, e} - LinearSolve[myM, {d, e}]
#1 + #2 x &[6, 7]
myB = Array[#1 (#2 + a #1) &, {3, 3}]
myB // MF
Eigenvalues[myB]
Eigenvectors[myB]
NullSpace[( 1  0  -1  0  0
            0  0   0  1  0 )]
FullSimplify[MatrixExp[myM]] // MF
```

From **logic: boolean expressions**, non-commutative multiplication.
💡 Type ESC=>ESC for ⇒

```
LogicalExpand[! (a || ! (b && ! c) && ! (d || ! d))]
! (a && b) ⇒ ! a || ! b // LogicalExpand
And @@ {0 < 1, b, True, a, a == b}
{a b == b a, a ** b == b ** a}
```

Finite fields of order p^n, multiplication of polynomials is defined by a certain irreducible polynomial. For the purpose of demonstration, we pick $p = 5$ prime and $n = 4$. Elements are adressed by integer coefficients of polynomials.

```
<< Algebra`FiniteFields`
myF = GF[5, 4]
elem = myF[{1, 2}] + myF[{2, 2, 2, 1}]
ElementToPolynomial[elem, x]
GF[5, 4][{1, 2, 1}] GF[5, 4][{2, 2, 2, 1}]
FieldIrreducible[GF[5, 4], x]
```

Tensors are essential in geometry. The following code yields to a given metric $g_{i,j}$ the Christoffel-Tensor $\Gamma^k_{i,j}$ and the Riemannian curvature tensor $\mathcal{R}^l_{i,j,k}$, all expressions depending on coordinates $x_1, \ldots, x_n$. The `Dot` operation for matrices extends to tensors of arbitrary rank.

```
T = Transpose
Md[X_, r_] := T[Array[D[X, x#1] &, {Length[X]}], Flatten[{r + 1, Range[r]}]]
MΓ[g_] := Inverse[g].With[{Q = Md[g, 0]}, -Q + T[Q, {2, 3, 1}] + T[Q, {3, 1, 2}]] / 2
MR[g_] :=
 T[Md[MΓ[g], 1] + MΓ[g].MΓ[g], {1, 4, 3, 2}] - T[Md[MΓ[g], 1] + MΓ[g].MΓ[g], {1, 3, 4, 2}]
```

☿ For illustration we use the induced metric of $S^2 \subset \mathbb{R}^3$ with radius r. We can show that the sectional curvature is constant.

```
g = ( r^2 Cos[x2]^2   0
      0               r^2 )
R = FullSimplify[MR[g]]
X = {x1, x2}
Y = {y1, y2}
FullSimplify[(g.R.X.Y.X.Y) / ((g.X.X) (g.Y.Y) - (g.X.Y)^2)]
```

Formal setup and treatment of tensors.
☿ Produce ⟦ by ESC[[ESC, analogous ⟧

```
dim = Dimensions
myC = Array[c#1,#2,#3 &, {2, 3, 4}]
myD = Array[D[x^5, {x, #1}] + #2 + Sin[#4] &, {4, 5, 2, 2}]
dim[myC]
dim[Transpose[myC, {3, 2, 1}]]
dim[myC.myD]
dim[Transpose[myC, {2, 3, 1}]]
myC⟦2, All, 3⟧
myC⟦1, {3, 2}⟧ = myC⟦1, {2, 1}⟧
```

Some **differential equations** require **numerical treatment**. We also teach the use of `InterpolationFunction`.
⚠ the equations below describe geodesics on the geometric torus with radiuses R and r.

```
geod = {{x1''[t] == (2 r Sin[x2[t]] x1'[t] x2'[t]) / (R + r Cos[x2[t]]), x1'[0] == .1, x1[0] == .2,
    x2''[t] == - (R + r Cos[x2[t]]) Sin[x2[t]] x1'[t]^2 / r, x2'[0] == 1, x2[0] == .3},
   {x1[t], x2[t]}, {t, -10, 10}} /. {R → 2, r → 1.1}
sol = NDSolve @@ geod
sol /. t → 3.4
Off[ParametricPlot::"ppcom"]; (*no message when compilation to plot failed*)
ParametricPlot[{x1[t], x2[t]} /. sol⟦1⟧, {t, -10, 10}];
```

Some expressions do not translate to an explicit form, but *Mathematica* might still know rules, how to manipulate certain entities. This results in **implicit descriptions** via functions like `Polygamma`, `ProductLog`, etc.

```
D[x!, x]
Integrate[Exp[-t^2], t]
Reduce[z == w Exp[w], w]
SphericalHarmonicY[3, 2, θ, ϕ]
DSolve[b x'[t] == a t^α + x[t]^2, x[t], t]
myRec = y[x] /. RSolve[{y[x + 2] == 2 x y[x + 1] + y[x], y[0] == 1, y[1] == 1}, y[x], x][[1]]
Plot3D[Abs[myRec /. x → r + s ⅈ], {r, -2, 5}, {s, -2, 5}, ViewPoint → {-1, 1, 1}]
```

Convolution of **piecewise continuous functions** (*Mathematica* lacks elegance on this topic)

```
bas1 = UnitStep[x + 1 / 2] - UnitStep[x - 1 / 2]
Plot[bas1, {x, -2, 2}];
bas2 = ∫_{-∞}^{∞} bas1 (bas1 /. x → x - s) ⅆx // FullSimplify
Plot[bas2, {s, -2, 2}];
```