

The physics package

Leedehai

<https://github.com/leedehai/typst-physics>

Version 0.7.2, May 12, 2023

Document updated: May 12, 2023

NOTE (2023-04-02): [Typst](#) is version 0.x and evolving, and this package evolves with it. Also, the package itself is under development and fine-tuning. While the major version stays 0, no backward compatibility is guaranteed.

Contents

1. Introduction	2
2. Using physics	2
3. The symbols	2
3.1. Braces	2
3.2. Vector notations	3
3.3. Matrix notations	3
3.4. Dirac bracket notations	4
3.5. Math functions	5
3.6. Differentials and derivatives	6
3.6.1. Differentials	6
3.6.2. Ordinary derivatives	7
3.6.3. Partial derivatives (incl. mixed orders)	7
3.7. Miscellaneous	8
3.7.1. Reduced Planck constant (\hbar)	8
3.7.2. Tensors	8
3.7.3. Isotopes	9
3.7.4. Signal sequences (digital timing diagrams)	9
3.8. Symbolic addition	10
4. Acknowledgement	11

1. Introduction

[Typst](#) is typesetting framework aiming to become the next generation alternative to LATEX. It excels in its friendly user experience and performance.

The physics package provides handy Typst typesetting functions that make academic writing for physics simpler and faster, by simplifying otherwise very complex and repetitive expressions in the domain of physics.

This manual itself was generated using the Typst CLI and the physics package, so hopefully this document is able to provide you with a sufficiently self evident demonstration of how this package shall be used.

2. Using physics

- To use the physics package, you may import names specifically:

```
#import "physics.typ": curl, grad
```

The expression `$op("curl")(op("grad") f) ident curl (grad f) = 0$` is not foreign to any trained eye in physical mathematics.

- or you may simply import all names:

```
#import "physics.typ": *
```

The expression `$op("curl")(op("grad") f) ident curl (grad f)$` is not foreign to any trained eye in physical mathematics.

- sometimes you may want to import the names under a name space:

```
#import "physics.typ"
```

The expression `$op("curl")(op("grad") f) ident physics.curl (physics.grad f)$` is not foreign to any trained eye in physical mathematics.

3. The symbols

Some symbols are already provided as a Typst built-in. They are listed here just for completeness with annotation like `typst` this, as users coming from LATEX might not know they are already available in Typst out of box.

All symbols need to be used in **math mode** `$...$`.

3.1. Braces

Symbol	Abbr.	Example	Notes
<code>typst abs (content)</code>		$\text{abs}(\text{phi}(x)) \rightarrow \varphi(x) $	absolute
<code>typst norm (content)</code>		$\text{norm}(\text{phi}(x)) \rightarrow \ \varphi(x)\ $	norm
<code>order (content)</code>		$\text{order}(x^2) \rightarrow \mathcal{O}(x^2)$	order of magnitude
<code>Set (content)</code>		$\text{Set}(a_n), \text{Set}(a_i, \text{forall } i)$ $\rightarrow \{a_n\}, \{a_i \forall i\}$ $\text{Set}(\text{vec}(1,n), \text{forall } n)$ $\rightarrow \left\{ \begin{pmatrix} 1 \\ n \end{pmatrix} \middle \forall n \right\}$	math set, use Set not set since the latter is a Typst keyword

evaluated(<i>content</i>)	eval	$\text{eval}(f(x))_{\text{\tiny 0}^\infty}$ $\rightarrow f(x) _0^\infty$ $\text{eval}(f(x)/g(x))_{\text{\tiny 0}^1}$ $\rightarrow \frac{f(x)}{g(x)} _0^1$	attach a vertical bar on the right to denote evaluation boundaries
expectationvalue	expval	$\text{expval}(u) \rightarrow \langle u \rangle$ $\text{expval}(f/N) \rightarrow \langle \frac{f}{N} \rangle$	expectation value

3.2. Vector notations

Symbol	Abbr.	Example	Notes
<code>typst</code> <code>vec(...)</code>		$\text{vec}(1,2) \rightarrow \begin{pmatrix} 1 \\ 2 \end{pmatrix}$	column vector
<code>vecrow(...)</code>		$\text{vecrow}(1,2) \rightarrow (1,2)$ $\text{vecrow}(\text{sum_}\text{\tiny 0}^n \text{ a_i}, b) \rightarrow (\sum_0^n a_i, b)$	row vector
<code>TT</code>		$v^{\text{TT}}, A^{\text{TT}} \rightarrow v^T, A^T$	transpose
<code>vectorbold(content)</code>	<code>vb</code>	$\text{vb}(a), \text{va}(\mu_1) \rightarrow \mathbf{a}, \boldsymbol{\mu}_1$	vector, bold
<code>vectorarrow(content)</code>	<code>va</code>	$\text{va}(a), \text{va}(\mu_1) \rightarrow \vec{a}, \vec{\mu}_1$	vector, arrow
<code>vectorunit(content)</code>	<code>vu</code>	$\text{vu}(a), \text{vu}(\mu_1) \rightarrow \hat{a}, \hat{\mu}_1$	unit vector
<code>gradient</code>	<code>grad</code>	$\text{grad } f \rightarrow \nabla f$	gradient
<code>divergence</code>	<code>div</code>	$\text{div } \text{vb}(E) \rightarrow \nabla \cdot E$	divergence
<code>curl</code>		$\text{curl } \text{vb}(B) \rightarrow \nabla \times B$	curl
<code>laplacian</code>		$\text{diaer}(u) = c^2 \text{laplacian } u$ $\rightarrow \ddot{u} = c^2 \nabla^2 u$	Laplacian, different from <code>typst</code> laplace Δ
<code>dotproduct</code>	<code>dprod</code>	$a \text{ dprod } b \rightarrow a \cdot b$	dot product
<code>crossproduct</code>	<code>cprod</code>	$a \text{ cprod } b \rightarrow a \times b$	cross product

3.3. Matrix notations

Symbol	Abbr.	Example	Notes
<code>TT</code>		$v^{\text{TT}}, A^{\text{TT}} \rightarrow v^T, A^T$	transpose
<code>typst</code> <code>mat(...)</code>		$\text{mat}(1,2;3,4) \rightarrow \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	matrix
<code>matrixdet(...)</code>	<code>mdet</code>	$\text{mdet}(1,x;1,y) \rightarrow \begin{vmatrix} 1 & x \\ 1 & y \end{vmatrix}$	matrix determinant
<code>diagonalmatrix(...)</code>	<code>dmat</code>	$\text{dmat}(1,2) \rightarrow \begin{pmatrix} 1 & \\ & 2 \end{pmatrix}$ $\text{dmat}(1,a,\text{xi},\text{delim}:"[",\text{fill}:\emptyset)$ $\rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & \xi \end{bmatrix}$	diagonal matrix
<code>antidiagonalmatrix(...)</code>	<code>admat</code>	$\text{admat}(1,2) \rightarrow \begin{pmatrix} & 1 \\ 2 & \end{pmatrix}$ $\text{admat}(1,a,\text{xi},\text{delim}:"[",\text{fill}:\text{dot})$ $\rightarrow \begin{bmatrix} \cdot & \cdot & 1 \\ \cdot & a & \cdot \\ \xi & \cdot & \cdot \end{bmatrix}$	anti-diagonal matrix

<code>identitymatrix(...)</code>	<code>imat</code>	$\text{imat}(2) \rightarrow \begin{pmatrix} 1 & \\ & 1 \end{pmatrix}$ $\text{imat}(3, \text{delim: "[", fill: 0}) \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	identity matrix
<code>zeromatrix(...)</code>	<code>zmat</code>	$\text{zmat}(2) \rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ $\text{zmat}(3, \text{delim: "[", fill: 0}) \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	zero matrix
<code>jacobianmatrix(...)</code>	<code>jmat</code>	See below	Jacobian matrix
<code>hessianmatrix(...)</code>	<code>hmat</code>	See below	Hessian matrix
<code>xmatrix(m, n, func, delim)</code>	<code>xmat</code>	See below	Matrix built with an element building function

Jacobian matrix: `jacobianmatrix(...)`, i.e. `jmat(...)`.

$$\begin{array}{cc} \text{jmat}(f_1, f_2; x, y) & \text{jmat}(f_1, f_2; x, y, z; \text{delim: "["}) \\ \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} & \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \end{bmatrix} \end{array}$$

Hessian matrix: `hessianmatrix(...)`, i.e. `hmat(...)`.

$$\begin{array}{cc} \text{hmat}(f; x, y) & \text{hmat}(; x, y, z; \text{delim: "["}) \\ \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} & \begin{bmatrix} \frac{\partial^3}{\partial x^2} & \frac{\partial^3}{\partial x \partial y} & \frac{\partial^3}{\partial x \partial z} \\ \frac{\partial^3}{\partial y \partial x} & \frac{\partial^3}{\partial y^2} & \frac{\partial^3}{\partial y \partial z} \\ \frac{\partial^3}{\partial z \partial x} & \frac{\partial^3}{\partial z \partial y} & \frac{\partial^3}{\partial z^2} \end{bmatrix} \end{array}$$

Matrix built with an element building function: `xmatrix(m, n, func)`, i.e. `xmat(...)`. The element building function `func` takes two integers which are the row and column numbers starting from 1.

$$\begin{array}{cc} \begin{array}{l} \text{\#let } g = (i, j) \Rightarrow \$g^{(\#(i - 1)\#(j - 1))\$} \\ \text{xmat}(2, 2, \text{\#}g) \end{array} & \begin{array}{l} \text{xmat}(2, 3, \text{\#}(r, c) \Rightarrow \{ \\ \quad \$\text{"exp"}(\text{\#}r, \text{\#}c) = \text{\#calc.pow}(r, c)\$ \\ \}, \text{delim: "["}) \end{array} \\ \begin{pmatrix} g^{00} & g^{01} \\ g^{10} & g^{11} \end{pmatrix} & \begin{bmatrix} \exp(1, 1) = 1 & \exp(1, 2) = 1 & \exp(1, 3) = 1 \\ \exp(2, 1) = 2 & \exp(2, 2) = 4 & \exp(2, 3) = 8 \end{bmatrix} \end{array}$$

3.4. Dirac bracket notations

Symbol	Abbr. Example	Notes
<code>bra(content)</code>	$\text{bra}(u) \rightarrow \langle u $ $\text{bra}(\text{vec}(1, 2)) \rightarrow \left\langle \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right $	bra
<code>ket(content)</code>	$\text{ket}(u) \rightarrow u\rangle$ $\text{ket}(\text{vec}(1, 2)) \rightarrow \left \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\rangle$	ket

<code>expval(content)</code>	<code>expval(u)</code> $\rightarrow \langle u \rangle$ <code>expval(vec(1,2))</code> $\rightarrow \left\langle \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\rangle$	expectation
<code>braket(a, b)</code>	<code>braket(a), braket(u, v)</code> $\rightarrow \langle a a \rangle, \langle u v \rangle$ <code>braket(vec(1,2), b)</code> $\rightarrow \left\langle \begin{pmatrix} 1 \\ 2 \end{pmatrix} \middle b \right\rangle$	braket
<code>ketbra(a, b)</code>	<code>ketbra(a), ketbra(u, v)</code> $\rightarrow a\rangle\langle a , u\rangle\langle v $ <code>ketbra(vec(1,2), b)</code> $\rightarrow \left \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\rangle \left\langle b \right $	ketbra
<code>innerproduct(a, b)</code>	<code>iproduct iprod(a), iprod(u, v)</code> $\rightarrow \langle a a \rangle, \langle u v \rangle$ <code>iproduct(a, vec(1,2))</code> $\rightarrow \left\langle a \middle \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right\rangle$	innerproduct = braket
<code>outerproduct(a, b)</code>	<code>oproduct oprod(a), oprod(u, v)</code> $\rightarrow a\rangle\langle a , u\rangle\langle v $ <code>oproduct(a, vec(1,2))</code> $\rightarrow \left a \right\rangle \left\langle \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right $	outerproduct = ketbra
<code>matricelement(n, M, m)</code>	<code>mel mel(n, diff_nu H, m)</code> $\rightarrow \langle n \partial_\nu H m \rangle$ <code>mel(n, vec(U, V), m)</code> $\rightarrow \left\langle n \middle \begin{pmatrix} U \\ V \end{pmatrix} \middle m \right\rangle$	matrix element

3.5. Math functions

Typst built-in math operators: [source code](#).

Expressions

`sin(x), sinh(x), arcsin(x), asin(x)`
`cos(x), cosh(x), arccos(x), acos(x)`
`tan(x), tanh(x), arctan(x), atan(x)`
`sec(x), sech(x), arcsec(x), asec(x)`
`csc(x), csch(x), arccsc(x), acsc(x)`
`cot(x), coth(x), arccot(x), acot(x)`

Results

`sin(x), sinh(x), arcsin(x), asin(x)`
`cos(x), cosh(x), arccos(x), acos(x)`
`tan(x), tanh(x), arctan(x), atan(x)`
`sec(x), sech(x), arcsec(x), asec(x)`
`csc(x), csch(x), arccsc(x), acsc(x)`
`cot(x), coth(x), arccot(x), acot(x)`

Expressions

`typst Pr(x)`
`typst exp x`
`typst log x, lg x, ln x`
`typst det A`
`diag(-1,1,1,1)`

`trace A, tr A`
`Trace A, Tr A`
`rank A`
`erf(x)`
`Res A`

Results

`Pr(x)`
`exp x`
`log x, lg x, ln x`
`det A`
`diag(-1,1,1,1)`

`trace A, tr A`
`Trace A, Tr A`
`rank A`
`erf(x)`
`Res A`

Notes

probability
exponential
logarithmic
matrix determinant
diagonal matrix, compact form (use `dmat` for the “real” matrix form)
matrix trace
matrix trace, alt.
matrix rank
Gauss error function
residue (complex analysis)

$\text{Re } z, \text{Im } z$	$\text{Re } z, \text{Im } z$	real, imaginary (complex analysis)
$\text{sgn } x$	$\text{sgn } x$	sign function

3.6. Differentials and derivatives

Symbol	Abbr.	Example	Notes
differential(...)	dd	e.g. $df, dx dy, d^3 x, dx \wedge dy$ See Section 3.6.1	differential
variation(...)	var	$\text{var}(f) \rightarrow \delta f$ $\text{var}(x, y) \rightarrow \delta x \delta y$	variation, shorthand of <code>dd(..., d: delta)</code>
difference(...)		$\text{difference}(f) \rightarrow \Delta f$ $\text{difference}(x, y) \rightarrow \Delta x \Delta y$	difference, shorthand of <code>dd(..., d: Delta)</code>
derivative(...)	dv	e.g. $\frac{d}{dx}, \frac{df}{dx}, \frac{\Delta^k f}{\Delta x^k}, df/dx$ See Section 3.6.2	derivative
partialderivative(...)	pdv	e.g. $\frac{\partial}{\partial x}, \frac{\partial f}{\partial x}, \frac{\partial^4 f}{\partial x^2 \partial y^2}, \frac{\partial^5 f}{\partial x^2 \partial y^3}, \partial f / \partial x$ See Section 3.6.3	partial derivative, could be mixed order

3.6.1. Differentials

Functions: `differential(*args, **kwargs)`, abbreviated as `dd(...)`.

- positional *args*: the variable names, then at the last **optionally** followed by an order number e.g. 2, or an order array e.g. [2,3], [k], [m n, lambda+1].
- named *kwargs*:
 - d*: the differential symbol [default: `upright(d)`].
 - p*: the product symbol connecting the components [default: `none`].

Order assignment algorithm:

- If there is no order number or order array, all variables has order 1.
- If there is an order number (not an array), then this order number is assigned to *every* variable, e.g. `dd(x, y, 2)` assigns $x \leftarrow 2, y \leftarrow 2$.
- If there is an order array, then the orders therein are assigned to the variables in order, e.g. `dd(f, x, y, [2, 3])` assigns $x \leftarrow 2, y \leftarrow 3$.
- If the order array holds fewer elements than the number of variables, then the orders of the remaining variables are 1, e.g. `dd(x, y, z, [2, 3])` assigns $x \leftarrow 2, y \leftarrow 3, z \leftarrow 1$.
- If a variable x has order 1, it is rendered as dx not $d^1 x$.

Examples

(1) <code>dd(f), dd(x, y)</code>	(2) <code>dd(x, 3), dd(f, [k]), dd(f, [k], d:delta)</code>
$df, dx dy$	$d^3 x, d^k f, \delta^k f$
(3) <code>dd(f, 2), dd(vb(x), t, [3, 1])</code>	(4) <code>dd(x, y, [2, 3]), dd(x, y, z, [2, 3])</code>
$d^2 f, d^3 x dt$	$d^2 x d^3 y, d^2 x d^3 y dz$
(5) <code>dd(x, y, z, [[1, 1], rho+1, n_1])</code>	(6) <code>dd(x, y, d:Delta), dd(x, y, 2, d:Delta)</code>
$d^{[1, 1]} x d^{\rho+1} y d^{n_1} z$	$\Delta x \Delta y, \Delta^2 x \Delta^2 y$
(7) <code>dd(t, x_1, x_2, x_3, p:and)</code>	(7) <code>dd(t, x_1, x_2, x_3, d:upright(D))</code>

$$dt \wedge dx_1 \wedge dx_2 \wedge dx_3$$

$$DtDx_1Dx_2Dx_3$$

3.6.2. Ordinary derivatives

Function: `derivative(f, *args, **kwargs)`, abbreviated as `dv(...)`.

- *f*: the function, which can be `#none` or omitted,
- positional *args*: the variable name, then at the last **optionally** followed by an order number e.g. 2,
- named *kwargs*:
 - *d*: the differential symbol [default: `upright(d)`].
 - *s*: the “slash” separating the numerator and denominator [default: `none`], by default it produces the normal fraction form $\frac{df}{dx}$. The most common non-default is `slash` or simply `\`, so as to create a flat form df/dx that fits inline.

Order assignment algorithm: there is just one variable, so the assignment is trivial: simply assign the order number (default to 1) to the variable. If a variable x has order 1, it is rendered as x not x^1 .

Examples

(1) `dv(,x), dv(,x,2), dv(f,x,k+1)`

$$\frac{d}{dx}, \frac{d^2}{dx^2}, \frac{d^{k+1}f}{dx^{k+1}}$$

(2) `dv(, vb(r)), dv(f, vb(r)_e, 2)`

$$\frac{d}{dr}, \frac{d^2}{dr_e^2}$$

(3) `dv(f,x,2,s:\/), dv(f,xi,k+1,s:slash)`

$$d^2f/dx^2, d^{k+1}f/d\xi^{k+1}$$

(4) `dv(, x, d:delta), dv(, x, 2, d:Delta)`

$$\frac{\delta}{\delta x}, \frac{\Delta^2}{\Delta x^2}$$

(5) `dv(vb(u), t, 2, d: upright(D))`

$$\frac{D^2u}{Dt^2}$$

(6) `dv(vb(u), t, 2, d:upright(D), s:slash)`

$$D^2u/Dt^2$$

3.6.3. Partial derivatives (incl. mixed orders)

Function: `partialderivative(f, *args, **kwargs)`, abbreviated as `pdv(...)`.

- *f*: the function, which can be `#none` or omitted,
- positional *args*: the variable names, then at last **optionally** followed by an order number e.g. 2, or an order array e.g. [2,3], [k], [m n, lambda+1].
- named *kwargs*:
 - *s*: the “slash” separating the numerator and denominator [default: `none`], by default it produces the normal fraction form $\frac{\partial f}{\partial x}$. The most common non-default is `slash` or simply `\`, so as to create a flat form $\partial f/\partial x$ that fits inline.
 - *total*: the user-specified total order.
 - If it is absent, then (1) if the orders assigned to all variables are numeric, the total order number will be **automatically computed**; (2) if non-number symbols are present, computation will be attempted with minimum effort, and a user override with argument *total* may be necessary.

Order assignment algorithm:

- If there is no order number or order array, all variables has order 1.
- If there is an order number (not an array), then this order number is assigned to *every* variable, e.g. `pdv(f,x,y,2)` assigns $x \leftarrow 2, y \leftarrow 2$.
- If there is an order array, then the orders therein are assigned to the variables in order, e.g. `pdv(f,x,y,[2,3])` assigns $x \leftarrow 2, y \leftarrow 3$.

- If the order array holds fewer elements than the number of variables, then the orders of the remaining variables are 1, e.g. `pdv(f,x,y,z,[2,3])` assigns $x \leftarrow 2, y \leftarrow 3, z \leftarrow 1$.
- If a variable x has order 1, it is rendered as x , not x^1 .

Examples

(1) `pdv(,x), pdv(,t,2), pdv(,lambda,[k])`

$$\frac{\partial}{\partial x}, \frac{\partial^2}{\partial t^2}, \frac{\partial^k}{\partial \lambda^k}$$

(2) `pdv(f,vb(r)), pdv(phi,vb(r)_e,2)`

$$\frac{\partial \varphi}{\partial r}, \frac{\partial^2 \varphi}{\partial r_e^2}$$

(3) `pdv(,x,y), pdv(,x,y,2)`

$$\frac{\partial^2}{\partial x \partial y}, \frac{\partial^4}{\partial x^2 \partial y^2}$$

(4) `pdv(f,x,y,2), pdv(f,x,y,3)`

$$\frac{\partial^4 \varphi}{\partial x^2 \partial y^2}, \frac{\partial^6 \varphi}{\partial x^3 \partial y^3}$$

(5) `pdv(,x,y,[2,1]), pdv(,x,y,[1,2])`

$$\frac{\partial^3}{\partial x^2 \partial y}, \frac{\partial^3}{\partial x \partial y^2}$$

(6) `pdv(,t,2,s:\), pdv(f,x,y,s:slash)`

$$\partial^2 / \partial t^2, \partial^2 f / \partial x \partial y$$

(7) `pdv(, (x^1), (x^2), (x^3), [1,3])`

$$\frac{\partial^5}{\partial (x^1) \partial (x^2)^3 \partial (x^3)}$$

(8) `pdv(phi,x,y,z,tau, [2,2,2,1])`

$$\frac{\partial^7 \varphi}{\partial x^2 \partial y^2 \partial z^2 \partial \tau}$$

(9) `pdv(,x,y,z,t,[1,xi,2,eta+2])`

$$\frac{\partial^{\eta+\xi+5}}{\partial x \partial y^\xi \partial z^2 \partial t^{\eta+2}}$$

(10) `pdv(,x,y,z,[xi n,n-1],total:(xi+1)n)`

$$\frac{\partial^{(\xi+1)n}}{\partial x^{\xi n} \partial y^{n-1} \partial z}$$

(11) `integral_V dd(V) (pdv(cal(L), phi) - diff_mu (pdv(cal(L), (diff_mu phi)))) = 0`

$$\int_V dV \left(\frac{\partial \mathcal{L}}{\partial \varphi} - \partial_\mu \left(\frac{\partial \mathcal{L}}{\partial (\partial_\mu \varphi)} \right) \right) = 0$$

3.7. Miscellaneous

3.7.1. Reduced Planck constant (hbar)

In the default font, the Typst built-in symbol `planck.reduce` \hbar looks a bit off: on letter “h” there is a slash instead of a horizontal bar, contrary to the symbol’s colloquial name “h-bar”. This package offers `hbar` to render the symbol in the familiar form: \hbar . Contrast:

Typst’s <code>planck.reduce</code>	$E = \hbar \omega$	$\frac{\pi G^2}{\hbar c^4}$	$A e^{\frac{i(px-Et)}{\hbar}}$	$i\hbar \frac{\partial}{\partial t} \psi = -\frac{\hbar^2}{2m} \nabla^2 \psi$
this package’s <code>hbar</code>	$E = \hbar \omega$	$\frac{\pi G^2}{\hbar c^4}$	$A e^{\frac{i(px-Et)}{\hbar}}$	$i\hbar \frac{\partial}{\partial t} \psi = -\frac{\hbar^2}{2m} \nabla^2 \psi$

3.7.2. Tensors

Tensors are often expressed using the [abstract index notation](#), which makes the contravariant and covariant “slots” explicit. The intuitive solution of using superscripts and subscripts do not suffice if

both upper (contravariant) and lower (covariant) indices exist, because the notation rules require the indices be vertically separated: e.g. T^a_b and T_a^b , which are of different shapes. “ T_b^a ” is flatly wrong, and $T^{(space\ w)}_{(i\ space\ j)}$ produces a weird-looking “ T_i^w ” (note w, j vertically overlap).

Function: `tensor(symbol, *args)`.

- *symbol*: the tensor symbol,
- positional *args*: each argument takes the form of $+...$ or $-...$, where a $+$ prefix denotes an upper index and a $-$ prefix denotes a lower index.

Examples

(1) `tensor(u,+a), tensor(v,-a)` (2) `tensor(h,+mu,+nu), tensor(g,-mu,-nu)`

$$u^a, v_a$$

$$h^{\mu\nu}, g_{\mu\nu}$$

(3) `tensor(T,+a,-b), tensor(T,-a,+b)`

(4) `tensor(T, -i, +w, -j)`

$$T^a_b, T_a^b$$

$$T_i^w_j$$

(5) `tensor((dd(x^lambda)), -a)`

(6) `tensor(AA,+a,+b,-c,-d,+e,-f,+g,-h)`

$$(dx^\lambda)_a$$

$$\mathbb{A}^{ab\ e\ g}_{cd\ f\ h}$$

(7) `tensor(R, -a, -b, -c, +d)`

(8) `tensor(T,+1,-I(1,-1),+a_bot,-+,-+)`

$$R_{abc}^d$$

$$T^1_{I(1,-1)} a_\perp^-$$

(9) `grad_mu A^nu = diff_mu A^nu + tensor(Gamma,+nu,-mu,-lambda) A^lambda`

$$\nabla_\mu A^\nu = \partial_\mu A^\nu + \Gamma^\nu_{\mu\lambda} A^\lambda$$

3.7.3. Isotopes

Function: `isotope(element, a: ..., z: ...)`.

- *element*: the chemical element (use “.” for multi-letter symbols)
- *a*: the mass number A [default: none].
- *z*: the atomic number Z [default: none].

Change log: Typst merged my [PR](#), which fixed a misalignment issue with the surrounding text.

Examples

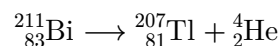
(1) `isotope(I, a:127)`

(2) `isotope("Fe", z:26)`

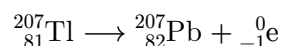
$$^{127}\text{I}$$

$$^{26}_{26}\text{Fe}$$


(3) `isotope("Bi",a:211,z:83) --> isotope("Tl",a:207,z:81) + isotope("He",a:4,z:2)`



(4) `isotope("Tl",a:207,z:81) --> isotope("Pb",a:207,z:82) + isotope(e,a:0,z:-1)`



3.7.4. Signal sequences (digital timing diagrams)

In engineering, people often need to draw digital timing diagrams for signals, like .

Function: `signals(str, step:..., style:...)`.

- `str`: a string representing the signals. Each character represents an glyph (see below).
- `step` (optional): step width, i.e. how wide each glyph is [default: `#1em`].
- `color` (optional): the stroke color [default: `#black`].

Glyph characters

HLM \Leftrightarrow "10-" full step	hlm $\wedge v$ 1/2 step, 1/10 step	' , (edge) 0 step	= # empty, shaded
R (rise)	F (fall)	C (charge)	D (drain)
<	>	X	
ingore: (blankspace)	repeat: . (dot)		
separate: &			

Examples

(1) `signals("10.1")`, `signals("1|0|1|0R")`, `signals("CD")`, `signals("CD", step: #2em)`



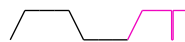
(2) `signals("M'H|L|h|l|^|v,&H'M'H|l,m,l|")` (the ampersand & serves as a separator)



(3) `signals("-|=|-", step: #2em)`, `signals("-|#|-")`, `signals("-<=>-<=")`

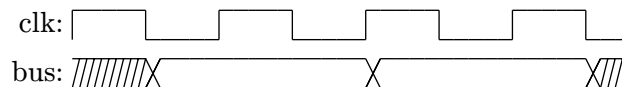


(4) `signals("R1..F0..", step: #.5em)``signals("R1.|v|1", step: #.5em, color:#fuchsia)`



(5)

`"clk:" & signals("|1....|0....|1....|0....|1....|0....|1....|0..", step: #0.5em) \`
`"bus:" & signals(" #.... X=... .. X=... .. X#.", step: #0.5em)`



3.8. Symbolic addition

This package implements a very rudimentary, **bare-minimum-effort** symbolic addition function to aid the automatic computation of a partial derivative's total order in the absence of user override (see Section 3.6.3). Though rudimentary and unsophisticated, this should suffice for most use cases in partial derivatives.

Function: `BMEsymadd([...])`.

- `...`: symbols that need to be added up e.g. `[1,2]`, `[a+1,b^2+1,2]`.

Examples

(1) `BMEsymadd([1])`, `BMEsymadd([2, 3])` \rightarrow 1, 5

(2) `BMEsymadd([a, b^2, 1])` \rightarrow $a + b^2 + 1$

(3) <code>BMEsymadd([a+1,2c,b,2,b])</code>	\rightarrow	$a + 2b + 2c + 3$
(4) <code>BMEsymadd([a+1,2(b+1),1,b+1,15])</code>	\rightarrow	$a + b + 2(b + 1) + 18$
(5) <code>BMEsymadd([a+1,2(b+1),1,(b+1),15])</code>	\rightarrow	$a + 3(b + 1) + 17$
(6) <code>BMEsymadd([a+1,2(b+1),1,3(b+1),15])</code>	\rightarrow	$a + 5(b + 1) + 17$
(7) <code>BMEsymadd([2a+1,xi,b+1,a xi + 2b+a,2b+1])</code>	\rightarrow	$3a + 5b + \xi + a\xi + 3$

4. Acknowledgement

Huge thanks to these LATEX packages, for lighting the way of physics typesetting.

- physics by Sergio C. de la Barrera,
- derivatives by Simon Jensen,
- tensor by Philip G. Ratcliffe et al.