The physics package

Leedehai
https://github.com/leedehai/typst-physics

Version 0.5, March 31, 2023
Document updated: March 31, 2023

NOTE (2023-03-31): Typst is in beta and evolving, and this package evolves with it. Also, the package itself is under development and fine-tuning. As a result, no backward compatibility is guaranteed yet, until the major version becomes a positive number.

Contents

1. Introduction

The `physics` package provides handy Typst typesetting functions that make academic writing for physics simpler and faster by simplifying otherwise very complex expressions.

This manual itself was generated using the Typst CLI and the `physics` package, so hopefully the manual source code 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 (TBI). They are listed here just for completeness, as users coming from LATEX's `physics` package 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**
`abs(`*content*`)`		`abs(phi(x))` $\longrightarrow \lvert\varphi(x)\rvert$	absolute, TBI
`norm(`*content*`)`		`norm(phi(x))` $\longrightarrow \lVert\varphi(x)\rVert$	norm, TBI
`order(`*content*`)`		`order(x^2)` $\longrightarrow \mathcal{O}(x^2)$	order of magnitude
`Set`		`Set(a_n)` $\longrightarrow \{a_n\}$ `Set(integral u, forall u)` $\longrightarrow \{\int u \mid \forall u\}$	math set, use `Set` not `set` since the latter is a Typst keyword
`evaluated`	`eval`	`eval(f(x))_0^infinity` $\longrightarrow f(x)\vert_0^\infty$ `eval(f(x)/g(x))_0^1` $\longrightarrow \left.\frac{f(x)}{g(x)}\right\vert_0^1$	attach a vertical bar on the right to denote evaluation boundaries
`expectationvalue`	`expval`	`expval(u)` $\longrightarrow \langle u\rangle$	expectation value

3.2. Vector notations

Symbol	**Abbr.**	**Example**	**Notes**
`vec`		`vec(1,2)` $\longrightarrow \begin{pmatrix}1\\2\end{pmatrix}$	column vector, TBI
`vecrow`		`vecrow(1,2)` $\longrightarrow (1,2)$ `vecrow(sum_0^n a_i, b)` $\longrightarrow \left(\sum_0^n a_i, b\right)$	row vector
`TT`		`v^TT, A^TT` $\longrightarrow v^{\mathsf{T}}, A^{\mathsf{T}}$	transpose (same for matrices)
`vectorbold(`*content*`)`	`vb`	`vb(a),va(mu_1)` $\longrightarrow \boldsymbol{a}, \boldsymbol{\mu}_1$	vector, bold
`vectorarrow(`*content*`)`	`va`	`va(a),va(mu_1)` $\longrightarrow \vec{\boldsymbol{a}}, \overrightarrow{\boldsymbol{\mu}_1}$	vector, arrow
`vectorunit(`*content*`)`	`vu`	`vu(a),vu(mu_1)` $\longrightarrow \hat{\boldsymbol{a}}, \widehat{\boldsymbol{\mu}_1}$	unit vector
`gradient`	`grad`	`grad f` $\longrightarrow \boldsymbol{\nabla} f$	gradient
`divergence`	`div`	`div vb(E)` $\longrightarrow \boldsymbol{\nabla}\cdot\boldsymbol{E}$	divergence
`curl`		`curl vb(B)` $\longrightarrow \boldsymbol{\nabla}\times\boldsymbol{B}$	curl

`laplacian`	`diaer(u) = c^2 laplacian u` $\longrightarrow \ddot{u} = c^2 \nabla^2 u$	laplacian
`dotproduct`	`a dotproduct b` $\longrightarrow a \cdot b$	dot product
`crossproduct`	`a crossproduct b` $\longrightarrow a \times b$	cross product

3.3. Matrix notations

Symbol	Abbr.	Example	Notes
`TT`		`v^TT, A^TT` $\longrightarrow v^{\mathsf{T}}, A^{\mathsf{T}}$	transpose (same for vectors)
`mat`		`mat(1,2;3,4)` $\longrightarrow \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	matrix, TBI
`matrixdet(...)`	`mdet`	`mdet(1,x;1,y)` $\longrightarrow \begin{vmatrix} 1 & x \\ 1 & y \end{vmatrix}$	matrix determinant
`diagonalmatrix(...)`	`dmat`	`dmat(1,2)` $\longrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ `dmat(1,a,xi,delim:"[")` $\longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & \xi \end{bmatrix}$	diagonal matrix
`antidiagonalmatrix(...)`	`admat`	`admat(1,2)` $\longrightarrow \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$ `admat(1,a,xi,delim:"[")` $\longrightarrow \begin{bmatrix} 0 & 0 & 1 \\ 0 & a & 0 \\ \xi & 0 & 0 \end{bmatrix}$	anti-diagonal matrix
`identitymatrix(...)`	`imat`	`imat(#2)` $\longrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ `imat(#3,delim:"[")` $\longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	identity matrix, **note:** Typst needs # to parse input as a number
`zeromatrix(...)`	`zmat`	`zmat(#2)` $\longrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ `zmat(#3,delim:"[")` $\longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	zero matrix, **note:** Typst needs # to parse input as a number

3.4. Dirac braket notations

Symbol	Abbr.	Example	Notes
`bra(`*content*`)`		`bra(u)` $\longrightarrow \langle u \vert$ `bra(limits(sum)_(i=0)^n i)` $\longrightarrow \left\langle \sum_{i=0}^{n} i \right\vert$	bra
`ket(`*content*`)`		`ket(u)` $\longrightarrow \vert u \rangle$ `ket(limits(sum)_(i=0)^n i)` $\longrightarrow \left\vert \sum_{i=0}^{n} i \right\rangle$	ket
`braket(`*a*, *b*`)`		`braket(u, v)` $\longrightarrow \langle u \vert v \rangle$ `braket(limits(sum)_(i=0)^n i, b)` $\longrightarrow \left\langle \sum_{i=0}^{n} i \middle\vert b \right\rangle$	braket
`ketbra(`*a*, *b*`)`		`ketbra(u, v)` $\longrightarrow \vert u \rangle \langle v \vert$	ketbra

		`ketbra(limits(sum)_(i=0)^n i, b)` $\to \left\lvert \sum\limits_{i=0}^{n} i \right\rangle \langle b\rvert$	
`innerproduct(`a, b`)`	`iprod`	`iprod(u, v)` $\to \langle u\lvert v\rangle$ `iprod(limits(sum)_(i=0)^n i, b)` $\to \left\langle \sum\limits_{i=0}^{n} i \middle\vert b \right\rangle$	innerproduct
`outerproduct(`a, b`)`	`oprod`	`oprod(u, v)` $\to \lvert u\rangle\langle v\rvert$ `oprod(limits(sum)_(i=0)^n i,b)` $\to \left\lvert \sum\limits_{i=0}^{n} i \right\rangle \langle b\rvert$	outerproduct
`matrixelement(`n, M, m`)`	`mel`	`mel(n, diff_nu H, m)` $\to \langle n\lvert \partial_\nu H\rvert m\rangle$ `mel(n, limits(sum)_(i=0)^n i, m)` $\to \left\langle n \middle\vert \sum\limits_{i=0}^{n} i \middle\vert m \right\rangle$	matrix element (quantum theory)

3.5. Math functions

Typst built-in (TBI) math operators: source code.

Expressions	**Results**
`sin(x), sinh(x), arcsin(x), asin(x)`	$\sin(x), \sinh(x), \arcsin(x), \mathrm{asin}(x)$
`cos(x), cosh(x), arccos(x), acos(x)`	$\cos(x), \cosh(x), \arccos(x), \mathrm{acos}(x)$
`tan(x), tanh(x), arctan(x), atan(x)`	$\tan(x), \tanh(x), \arctan(x), \mathrm{atan}(x)$
`sec(x), sech(x), arcsec(x), asec(x)`	$\sec(x), \mathrm{sech}(x), \mathrm{arcsec}(x), \mathrm{asec}(x)$
`csc(x), csch(x), arccsc(x), acsc(x)`	$\csc(x), \mathrm{csch}(x), \mathrm{arccsc}(x), \mathrm{acsc}(x)$
`cot(x), coth(x), arccot(x), acot(x)`	$\cot(x), \coth(x), \mathrm{arccot}(x), \mathrm{acot}(x)$

Expressions	**Results**	**Notes**
`Pr(x)`	$\Pr(x)$	probability, TBI
`exp x, log x, lg x, ln x`	$\exp x, \log x, \lg x, \ln x$	exponential and logarithmic, TBI
`det A`	$\det A$	matrix determinant, TBI
`diag(-1,1,1,1)`	$\mathrm{diag}(-1, 1, 1, 1)$	diagonal matrix, compact form (use `dmat` for the “real” matrix form)
`trace A, tr A`	$\mathrm{trace}\, A, \mathrm{tr}\, A$	matrix trace
`Trace A, Tr A`	$\mathrm{Trace}\, A, \mathrm{Tr}\, A$	matrix trace, alt.
`rank A`	$\mathrm{rank}\, A$	matrix rank
`erf(x)`	$\mathrm{erf}(x)$	Gauss error function
`Res A`	$\mathrm{Res}\, A$	residue
`Re z, Im z`	$\mathrm{Re}\, z, \mathrm{Im}\, z$	real, imaginary parts of a complex number
`sgn x`	$\mathrm{sgn}\, x$	sign function

3.6. Differentials and derivatives

Symbol	Abbr.	Example	Notes
`differential(...)`	`dd`	e.g. $\mathrm{d}f, \mathrm{d}x\mathrm{d}y, \mathrm{d}^3x, \mathrm{d}x \wedge \mathrm{d}y$ See Section 3.6.1	differential
`variation(...)`	`var`	`var(f)` $\longrightarrow \delta f$ `var(x,y)` $\longrightarrow \delta x \delta y$	variation, shorthand of `dd(..., d: delta)`
`derivative(...)`	`dv`	e.g. $\frac{\mathrm{d}}{\mathrm{d}x}, \frac{\mathrm{d}f}{\mathrm{d}x}, \frac{\Delta^k f}{\Delta x^k}, \mathrm{d}f/\mathrm{d}x$ 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,
- named *kwargs*:
  - `n`: an order number an order number array [default: `none`],
  - `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 number 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,n:2)` assigns $x \leftarrow 2, y \leftarrow 2$.
- If there is an order number array, then the order numbers therein are assigned to the variables in order, e.g. `dd(f,x,y,n:(2,3))` assigns $x \leftarrow 2, y \leftarrow 3$.
- If the order number array holds fewer numbers than the number of variables, then the orders of the remaining variables are 1, e.g. `dd(x,y,z,n:(2,3))` assigns $x \leftarrow 2, y \leftarrow 3, z \leftarrow 1$.
- If a variable x has order 1, it is rendered as $\mathrm{d}x$ not d^1x.

Examples

(1) `dd(f), dd(x,y), dd(x,y,z,p:and)`

$$\mathrm{d}f, \mathrm{d}x\mathrm{d}y, \mathrm{d}x \wedge \mathrm{d}y \wedge \mathrm{d}z$$

(2) `dd(phi,n:k,d:delta)`

$$\delta^k \varphi$$

(3) `dd(f,n:2), dd(vb(x),t,n:(3,))`

$$\mathrm{d}^2 f, \mathrm{d}^3 \boldsymbol{x} \mathrm{d}t$$

(4) `dd(x,y,n:(2,3)), dd(x,y,z,n:(2,3))`

$$\mathrm{d}^2 x \mathrm{d}^3 y, \mathrm{d}^2 x \mathrm{d}^3 y \mathrm{d}z$$

(5) `dd(x, y, z, n: ((1,1),rho+1,n_1))`

$$\mathrm{d}^{(1,1)} x \mathrm{d}^{\rho+1} y \mathrm{d}^{n_1} z$$

(6) `dd(x,y,d:Delta), dd(x,y,n:2,d:Delta)`

$$\Delta x \Delta y, \Delta^2 x \Delta^2 y$$

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, **optionally** followed by an order number,
- 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 $\frac{\mathrm{d}f}{\mathrm{d}x}$. The most common non-default is slash or simply \/, so as to create a flat form $\mathrm{d}f/\mathrm{d}x$ 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(,x,k+1)`

$$\frac{\mathrm{d}}{\mathrm{d}x}, \frac{\mathrm{d}^2}{\mathrm{d}x^2}, \frac{\mathrm{d}^{k+1}}{\mathrm{d}x^{k+1}}$$

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

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{r}}, \frac{\mathrm{d}^2}{\mathrm{d}\boldsymbol{r}^2}$$

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

$$\frac{\mathrm{d}f}{\mathrm{d}x}, \frac{\mathrm{d}^2 f}{\mathrm{d}x^2}, \frac{\mathrm{d}^{k+1} f}{\mathrm{d}\xi^{k+1}}$$

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

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{r}}, \frac{\mathrm{d}^2 f}{\mathrm{d}\boldsymbol{r}^2}$$

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

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

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

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

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

$$\frac{\mathrm{D}^2 \boldsymbol{u}}{\mathrm{D}t^2}$$

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

$$\mathrm{D}^2 \boldsymbol{u}/\mathrm{D}t^2$$

3.6.3. Partial derivatives

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

- *f*: the function, which can be #none or omitted,
- positional *args*: the variable names, **optionally** followed by an order number e.g. #2, or an order number array e.g. #(2,3).
  - Note # is important, since Typst constructs a number or array in **code mode** – without #, what follows will just be parsed as a sequence of generic symbols (not numbers) that are not operable in the numeric computation of the total order.
- 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.

Order assignment algorithm:

- If there is no order number or order number 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 number array, then the order numbers 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 number array holds fewer numbers 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.

The total order applied to the function differential is automatically calculated by adding the order numbers of the variables. Examples:

- `pdv(f, x)` has total order 1,
- `pdv(f, x, #2)` has total order 2,
- `pdv(f, x, y, #2)` has total order 2 + 2 = 4,
- `pdv(f, x, y, #(2, 3))` has total order 2 + 3 = 5,
- `pdv(f, x, y, z, #(2, 3))` has total order 2 + 3 + 1 = 6,

Examples

(1) `pdv(,x), pdv(,t,#2)`

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

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

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

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

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

3.7. Miscellaneous

3.7.1. Reduced Planck constant (hbar)

Due to the default font, the 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 `planck.reduce`	$E = \hbar \omega$	$\frac{\pi G^2}{\hbar c^4}$	$A e^{\frac{i(px - Et)}{\hbar}}$	$i\hbar \frac{\partial}{\partial t} \psi = -\frac{\hbar^2}{2m} \nabla^2 \psi$
this package's `hbar`	$E = \hbar \omega$	$\frac{\pi G^2}{\hbar c^4}$	$A e^{\frac{i(px - Et)}{\hbar}}$	$i\hbar \frac{\partial}{\partial t} \psi = -\frac{\hbar^2}{2m} \nabla^2 \psi$

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^a_b" is flatly wrong, and `T^(space w)_(i space j)` produces a weird-looking "$T_{i\ j}^{\ w}$" (note w, j vertically overlap).

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

- *symbol*: the tensor symbol,
- positional args: comma-separated list taking the form of $+\ldots$ or $-\ldots$, where a `+` prefix denotes an upper index and a `-` prefix denotes a lower index.

Examples

(1) `tensor(u,+a), tensor(v,-a)`

$$u^a, v_a$$

(2) `tensor(h,+mu,+nu), tensor(g,-mu,-nu)`

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

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

$${T^a}_b, {T_a}^b$$

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

$${{T_i}^w}_j$$

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

$$\left(\mathrm{d}x^\lambda\right)_a$$

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

$${{{{\mathbb{A}^{ab}}_{cd}}^e}_f{}^g}_h$$

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

$${R_{abc}}^d$$

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

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

(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`].

Examples

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

$${}^{127}\mathrm{I}$$

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

$${}_{26}\mathrm{Fe}$$

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

$${}^{211}_{83}\mathrm{Bi} \longrightarrow {}^{207}_{81}\mathrm{Tl} + {}^{4}_{2}\mathrm{He}$$

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

$${}^{207}_{81}\mathrm{Tl} \longrightarrow {}^{207}_{82}\mathrm{Pb} + {}^{0}_{-1}\mathrm{e}$$

4. Acknowledgement

Huge thanks to Sergio C. de la Barrera's LATEX `physics` package and Simon Jensen's LATEX `derivative` package, which lit the way for physics typesetting.