The physicx package

Wenjian Chern (Longaster*)

April 27, 2022, version v0.3

Abstract

physicx

1 Implementation

```
⟨*package⟩
⟨@@=physicx⟩
\cs_generate_variant:Nn \keys_set:nn { nx , on , ox }
\cs_generate_variant:Nn \use:nnnn { nnno }
\cs_generate_variant:Nn \seq_set_split:Nnn { Non, NVV, c, cnV, cVV }
\cs_generate_variant:Nn \tl_replace_all:Nnn { Non, Nox }
\cs_new:Npn \PHYSICXIGNORE
  { \exp_end: \exp_not:N \PHYSICXIGNORE }
\bool_new:N \g__physicx_mathtools_bool
\bool_new:N \g__physicx_physics_bool
\bool_new:N \g__physicx_compat_bool
\bool_new:N \g__physicx_short_bool
\bool_new:N \g__physicx_reqty_bool
\prg_new_conditional:Npnn \physicx_compat: { T, F, TF }
  {
    \bool_if:NTF \g__physicx_compat_bool
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_new_conditional:Npnn \physicx_short: { T, F, TF }
  {
    \bool_if:NTF \g__physicx_short_bool
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_new_conditional:Npnn \physicx_mathtools: { T, F, TF }
  {
    \bool_if:NTF \g__physicx_mathtools_bool
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_new_conditional:Npnn \physicx_option_or:nn #1#2 { T, F, TF }
  {
    \bool_lazy_or:nnTF
      { \cs:w g__physicx_ #1 _bool \cs_end: }
      { \cs:w g__physicx_ #2 _bool \cs_end: }
```

*Email: longaster@163.com

```
      { \prg_return_true: }
      { \prg_return_false: }
  }
\cs_if_exist_use:NF \hook_gput_code:nnn { \use_none:nnn }
  { package/unicode-math/after } { ./package }
  {
    \cs_gset_eq:NN \physicx_unimath:TF \use_i:nn
    \cs_gset_eq:NN \physicx_unimath:T  \use:n
    \cs_gset_eq:NN \physicx_unimath:F  \use_none:n
  }
\prg_set_conditional:Npnn \physicx_unimath: { T, F, TF }
  {
    \tl_if_exist:cTF { ver @ unicode-math . \@pkgextension }
      { \prg_return_true: } { \prg_return_false: }
  }

\clist_new:N \l__physicx_tmpa_clist
\bool_new:N \l__physicx_tmpa_bool
\int_new:N \l__physicx_tmpa_int
\int_new:N \l__physicx_tmpb_int
\msg_new:nnnn { physicx } { unknown-key }
  { The~key~‘#1’~is~unknown~and~is~being~ignored. }
  {
    The~module~#2~does~not~have~a~key~called~#1.\\
    Check~that~you~have~spelled~the~key~name~correctly.
  }
\msg_new:nnn { physicx } { diag-key }
  { The~value~‘#1’~of~diag~key~is~unknown~and~is~being~ignored. }
```

1.1 Utils functions

\physicx_parse_range:nnnN
\physicx_parse_range_check:
\physicx_parse_range_nocheck:

Parse range, such as -3,6-8,9,10-.

```
\int_new:N \l__physicx_begin_int
\int_new:N \l__physicx_end_int
\int_new:N \l__physicx_max_int
\int_new:N \l__physicx_min_int
\bool_new:N \l__physicx_invalid_range_bool
\cs_new_protected:Npn \physicx_parse_range_check:
  {
    \cs_set_eq:NN \__physicx_parse_range_single:n \__physicx_parse_range_single_check:n
    \cs_set_eq:NN \__physicx_parse_range_range: \__physicx_parse_range_range_check:
  }
\cs_new_protected:Npn \physicx_parse_range_nocheck:
  {
    \cs_set_eq:NN \__physicx_parse_range_single:n \__physicx_parse_range_single_nocheck:n
    \cs_set_eq:NN \__physicx_parse_range_range: \__physicx_parse_range_range_nocheck:
  }
\cs_new_protected:Npn \physicx_parse_range:nnnN #1#2#3#4
  {
    \seq_set_eq:NN #4 \c_empty_seq
    \int_set:Nn \l__physicx_min_int {#1}
    \int_set:Nn \l__physicx_max_int {#2}
    \clist_map_inline:nn {#3}
      {
        \__physicx_parse_range_aux:n {##1}
        \bool_if:NF \l__physicx_invalid_range_bool
          { \seq_concat:NNN #4 #4 \l__physicx_tmpa_seq }
      }
  }
\cs_generate_variant:Nn \physicx_parse_range:nnnN { nnvN, nnxN }
\cs_new_protected:Npn \physicx_parse_range:nnN
  { \physicx_parse_range:nnnN { 1 } }
\cs_generate_variant:Nn \physicx_parse_range:nnN { nvN, nxN }
\cs_new_protected:Npn \__physicx_parse_range_aux:n #1
  {
    \bool_set_false:N \l__physicx_invalid_range_bool
    \seq_clear:N \l__physicx_tmpa_seq
    \tl_if_in:nnTF {#1} { - }
      {
        \seq_set_split:Nnn \l__physicx_tmpb_seq { - } {#1}
        \seq_pop_left:NN \l__physicx_tmpb_seq \l__physicx_tmpa_tl
        \tl_if_empty:NTF \l__physicx_tmpa_tl
          { \int_set_eq:NN \l__physicx_begin_int \l__physicx_min_int }
          {
            \int_set:Nn \l__physicx_begin_int { \l__physicx_tmpa_tl }
            \int_compare:nNnT \l__physicx_begin_int < \l__physicx_min_int
              {
                \int_set_eq:NN \l__physicx_begin_int \l__physicx_min_int
              }
          }
        \seq_pop_left:NN \l__physicx_tmpb_seq \l__physicx_tmpa_tl
        \tl_if_empty:NTF \l__physicx_tmpa_tl
          { \int_set_eq:NN \l__physicx_end_int \l__physicx_max_int }
          {
            \int_set:Nn \l__physicx_end_int { \l__physicx_tmpa_tl }
            \int_compare:nNnT \l__physicx_end_int > \l__physicx_max_int
              {
                \int_set_eq:NN \l__physicx_end_int \l__physicx_max_int
              }
          }
        \__physicx_parse_range_range:
      }
      { \__physicx_parse_range_single:n {#1} }
  }
\cs_new:Npn \__physicx_parse_range_single_check:n #1
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#1} > \l__physicx_max_int }
      { \int_compare_p:nNn {#1} < \l__physicx_min_int }
      { \bool_set_true:N \l__physicx_invalid_range_bool }
      { \seq_put_right:Nn \l__physicx_tmpa_seq {#1} }
  }
\cs_new:Npn \__physicx_parse_range_single_nocheck:n #1
  { \seq_put_right:Nn \l__physicx_tmpa_seq {#1} }
\cs_new_eq:NN \__physicx_parse_range_single:n \__physicx_parse_range_single_check:n
\cs_new:Npn \__physicx_parse_range_range_check:
  {
    \bool_lazy_or:nnTF
```

```
      { \int_compare_p:nNn \l__physicx_begin_int > \l__physicx_max_int }
      { \int_compare_p:nNn \l__physicx_begin_int > \l__physicx_end_int }
      { \bool_set_true:N \l__physicx_invalid_range_bool }
      {
        \int_step_inline:nnn
          { \l__physicx_begin_int } { \l__physicx_end_int }
          { \seq_put_right:Nn \l__physicx_tmpa_seq {##1} }
      }
  }
\cs_new:Npn \__physicx_parse_range_range_nocheck:
  {
    \int_compare:nNnTF \l__physicx_begin_int > \l__physicx_end_int
      { \bool_set_true:N \l__physicx_invalid_range_bool }
      {
        \int_step_inline:nnn
          { \l__physicx_begin_int } { \l__physicx_end_int }
          { \seq_put_right:Nn \l__physicx_tmpa_seq {##1} }
      }
  }
\cs_new_eq:NN \__physicx_parse_range_range: \__physicx_parse_range_range_check:
```

(*End definition for \physicx_parse_range:nnnN, \physicx_parse_range_check:, and \physicx_parse_range_nocheck:. These functions are documented on page* **??***.*)

```
\cs_new:Npn \__physicx_if_keyval:nTF #1
  { \tl_if_in:nnTF {#1} { = } }
\prg_new_conditional:Npnn \physicx_if_num:n #1 { T, F, TF }
  {
    \regex_match:nnTF { \A [[:digit:]]+ \Z } {#1}
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_new_conditional:Npnn \physicx_if_num_sign:n #1 { T, F, TF }
  {
    \regex_match:nnTF { \A [\+\-]* [[:digit:]]+ \Z } {#1}
      { \prg_return_true: } { \prg_return_false: }
  }
\cs_new:Npn \physicx_search_also:nn #1#2
  {
    \clist_map_inline:nn {#1}
      {
        \exp_args:Nno \keys_if_exist:nnT {##1} { \l_keys_key_str }
          {
            \clist_map_break:n
              { \keys_set:no {##1} { \l_keys_key_str = {#2} } }
          }
      }
  }
\prg_new_conditional:Npnn \physicx_search_also:nn #1#2 { T, F, TF }
  {
    \bool_set_false:N \l__physicx_tmpa_bool
    \clist_map_inline:nn {#1}
      {
        \exp_args:Nno \keys_if_exist:nnT {##1} { \l_keys_key_str }
          {
            \clist_map_break:n
```

```
              {
                \bool_set_true:N \l__physicx_tmpa_bool
                \keys_set:no {##1} { \l_keys_key_str = {#2} }
              }
          }
      }
    \bool_if:NTF \l__physicx_tmpa_bool
      { \prg_return_true: } { \prg_return_false: }
  }
\cs_generate_variant:Nn \physicx_search_also:nn { no , oo }
\prg_generate_conditional_variant:Nnn \physicx_search_also:nn { no , oo } { T , F , TF }
\cs_new_protected:Npn \physicx_new_type:nnn #1#2#3
  { \keys_define:nn { physicx/#1 } { type / #2 .meta:n = {#3} } }

\tl_const:Nn \c_physicx_order_tl { \mathcal{o} }
\tl_const:Nn \c_physicx_Order_tl { \mathcal{O} }
\cs_new:Npn \physicx_use_amssymb_type:
  {
    \cs_set_eq:NN \physicx_bf: \boldsymbol
  }
\cs_new:Npn \physicx_use_uni_bfit_type:
  {
    \cs_set_eq:NN \physicx_bf: \symbfit
  }
\cs_new:Npn \physicx_use_uni_bf_type:
  {
    \cs_set_eq:NN \physicx_bf: \symbf
  }
\cs_new:Npn \physicx_left: { \mathopen{}\mathclose\bgroup\left }
\cs_new:Npn \physicx_right: { \aftergroup\egroup\right }
\cs_new:Npn \physicx_left:N { \mathopen{}\mathclose\bgroup }
\cs_new:Npn \physicx_right:N { \egroup }

\cs_new:Npn \__physicx_loadpackage_options:nnn #1#2#3
  {
    \clist_if_empty:nF {#1} { \PassOptionsToPackage {#1} {#3} }
    \RequirePackage {#3}
  }
\keys_define:nn { physicx }
  {
    compat .bool_set:N = \g__physicx_compat_bool ,
    compat .default:n = true ,
    short .bool_set:N = \g__physicx_short_bool ,
    short .default:n = true ,
    physics .code:n = \__physicx_loadpackage_options:nnn {#1} { } {physics} ,
    physics .default:n = { } ,
    mathtools .code:n = \__physicx_loadpackage_options:nnn {#1} { } {mathtools} ,
    mathtools .default:n = { } ,
    unimath .code:n = \__physicx_loadpackage_options:nnn {#1} { } { unicode-math } ,
    unimath .default:n = { } ,
    reqty .bool_set:N = \g__physicx_reqty_bool ,
    reqty .default:n = true ,
    reqty .initial:n = true ,
    noqty .meta:n = { reqty = false } ,
  }
```

```
%
\ProcessKeysPackageOptions { physicx }
%
\@ifpackageloaded{physics}
  { \bool_set_true:N \g__physicx_compat_bool }
  { }
\@ifpackageloaded{mathtools}
  { \bool_set_true:N \g__physicx_mathtools_bool }
  { \bool_set_false:N \g__physicx_mathtools_bool }
%
\physicx_compat:T
  {
    \tl_set_eq:NN \ordersymbol \c_physicx_order_tl
    \tl_set_eq:NN \Ordersymbol \c_physicx_Order_tl
  }
%
\@ifpackageloaded {unicode-math}
  { \physicx_use_uni_bfit_type: }
  { \physicx_use_amssymb_type: }
\physicx_unimath:T { %% TODO:
  \AtBeginDocument{
    \DeclareDocumentCommand\vectorbold{ s m }
      { \IfBooleanTF{#1} { \physicx_bf:{#2} } { \mathbf{#2} } }
    \DeclareDocumentCommand\vectorarrow{ s m }
      { \IfBooleanTF{#1} { \vec{\physicx_bf:{#2}} } { \vec{\mathbf{#2}} } }
    \DeclareDocumentCommand\vectorunit{ s m }
      {\IfBooleanTF{#1} { \physicx_bf:{\hat{#2}} } { \hat{\mathbf{#2}} } }
    \setmathfont[range={"2219}]{STIX~Two~Math}
    \DeclareDocumentCommand \dotproduct { } { \vysmblkcircle }
    \DeclareDocumentCommand \crossproduct { } { \vectimes }
    \DeclareDocumentCommand \vnabla { } { \symbf \nabla }
    \let\divisionsymbol\div
    \let\div\divergence
  }
}
```

`\physicxset` physicx setup command.

```
\NewDocumentCommand \physicxset { s m }
  {
    \IfBooleanTF {#1}
      { \keys_set:nn { physicx/#2 } }
      { \keys_set:nn { physicx } {#2} }
  }
```

(*End definition for* `\physicxset`. *This function is documented on page* **??**.)

1.2 Quantity things

1.2.1 New quantity interfaces

```
\tl_new:N \l__physicx_quantity_args_tl
\tl_new:N \l__physicx_quantity_code_tl
\tl_new:N \l__physicx_quantity_left_size_tl
\tl_new:N \l__physicx_quantity_left_tl
```

```
\tl_new:N \l__physicx_quantity_post_tl
\tl_new:N \l__physicx_quantity_pre_tl
\tl_new:N \l__physicx_quantity_right_size_tl
\tl_new:N \l__physicx_quantity_right_tl
\keys_define:nn { physicx }
  { quantity .code:n = \keys_set:nn { physicx/quantity } {#1} }
\keys_define:nn { physicx/quantity }
  {
    pre   .tl_set:N = \l__physicx_quantity_pre_tl  ,
    post  .tl_set:N = \l__physicx_quantity_post_tl ,
    left  .tl_set:N = \l__physicx_quantity_left_tl ,
    right .tl_set:N = \l__physicx_quantity_right_tl ,
    left-size  .code:n = { \tl_set_eq:NN \l__physicx_quantity_left_size_tl  #1 } ,
    right-size .code:n = { \tl_set_eq:NN \l__physicx_quantity_right_size_tl #1 } ,
    size  .meta:n = { left-size = {#1} , right-size = {#1} } ,
    noauto .meta:n =  { left-size = \c_empty_tl , right-size = \c_empty_tl } ,
    noauto .value_required:n = false ,
    args  .code:n =
      \tl_set:Nn \l__physicx_quantity_args_tl { [#1] } ,
    args* .tl_set:N = \l__physicx_quantity_args_tl ,
    code  .tl_set:N = \l__physicx_quantity_code_tl ,
    type .multichoice: ,

    settype .code:n = \setquantitytype #1 ,

    unknown .code:n =
      \tl_set:Nx \l__physicx_tmpa_tl { \tl_head:N \l_keys_key_str }
      \token_if_eq_meaning:NNTF \l__physicx_tmpa_tl \c_backslash_str
        { \use:n } { \use_ii:nn }
      {
        \cs_if_exist:cTF { \tl_tail:N \l_keys_key_str }
          {
            \keys_set:nx { physicx/quantity }
              { size = \exp_not:c { \tl_tail:N \l_keys_key_str } }
            \use_none:n
          }
          { \use:n }
      }
      {
        \physicx_search_also:nnF
          {
            physicx/quantity/type ,
          }
          {#1}
          {
            \msg_error:nnxx { physicx } { unknown-key }
              \l_keys_path_str { physicx/quantity }
          }
      } ,
  }
\NewDocumentCommand \setquantitytype { >{ \TrimSpaces } m }
  { \physicx_new_type:nnn { quantity } {#1} }
\setquantitytype { b } { left={[} , right={]} , }
\setquantitytype { B } { left={\{} , right={\}} , }
```

```
\setquantitytype { p } { left={(} , right={)} , }
\setquantitytype { v } { left=\vert , right=\vert , }
\setquantitytype { V } { left=\Vert , right=\Vert , }
\setquantitytype { a } { left=\langle , right=\rangle , }
\setquantitytype { m } { left=\begin{matrix} , right=\end{matrix} , noauto }
\setquantitytype { bm } { left=\begin{bmatrix} , right=\end{bmatrix} , noauto }
\setquantitytype { Bm } { left=\begin{Bmatrix} , right=\end{Bmatrix} , noauto }
\setquantitytype { pm } { left=\begin{pmatrix} , right=\end{pmatrix} , noauto }
\setquantitytype { vm } { left=\begin{vmatrix} , right=\end{vmatrix} , noauto }
\setquantitytype { Vm } { left=\begin{Vmatrix} , right=\end{Vmatrix} , noauto }
\setquantitytype { sm } { left=\begin{smallmatrix} , right=\end{smallmatrix} , noauto }
\physicx_mathtools:T
  {
    \setquantitytype { m* } { left=\begin{matrix*} , right=\end{matrix*} , noauto }
    \setquantitytype { bm* } { left=\begin{bmatrix*} , right=\end{bmatrix*} , noauto }
    \setquantitytype { Bm* } { left=\begin{Bmatrix*} , right=\end{Bmatrix*} , noauto }
    \setquantitytype { pm* } { left=\begin{pmatrix*} , right=\end{pmatrix*} , noauto }
    \setquantitytype { vm* } { left=\begin{vmatrix*} , right=\end{vmatrix*} , noauto }
    \setquantitytype { Vm* } { left=\begin{Vmatrix*} , right=\end{Vmatrix*} , noauto }
    \setquantitytype { sm* } { left=\begin{smallmatrix*} , right=\end{smallmatrix*} , noauto
    \setquantitytype { sbm } { left=\begin{bsmallmatrix} , right=\end{bsmallmatrix} , noauto
    \setquantitytype { sBm } { left=\begin{Bsmallmatrix} , right=\end{Bsmallmatrix} , noauto
    \setquantitytype { spm } { left=\begin{psmallmatrix} , right=\end{psmallmatrix} , noauto
    \setquantitytype { svm } { left=\begin{vsmallmatrix} , right=\end{vsmallmatrix} , noauto
    \setquantitytype { sVm } { left=\begin{Vsmallmatrix} , right=\end{Vsmallmatrix} , noauto
    \setquantitytype { sbm* } { left=\begin{bsmallmatrix*} , right=\end{bsmallmatrix*} , noa
    \setquantitytype { sBm* } { left=\begin{Bsmallmatrix*} , right=\end{Bsmallmatrix*} , noa
    \setquantitytype { spm* } { left=\begin{psmallmatrix*} , right=\end{psmallmatrix*} , noa
    \setquantitytype { svm* } { left=\begin{vsmallmatrix*} , right=\end{vsmallmatrix*} , noa
    \setquantitytype { sVm* } { left=\begin{Vsmallmatrix*} , right=\end{Vsmallmatrix*} , noa
  }
\keys_set:nn { physicx/quantity }
  {
    left-size = \left ,
    right-size = \right ,
    type = p ,
  }
```

\physicx_xquantity:nn
\newxquantity
\NewXQuantity

```
\cs_new:Npn \physicx_xquantity:nn #1#2
  {
    \group_begin:
    \keys_set:nn { physicx/quantity } {#1}
    \tl_if_empty:nF {#2} { \tl_set:Nn \l__physicx_quantity_code_tl {#2} }
    \__physicx_xquantity_aux:oooo
      { \l__physicx_quantity_left_tl }
      { \l__physicx_quantity_args_tl }
      { \l__physicx_quantity_code_tl }
      { \l__physicx_quantity_right_tl }
    \group_end:
  }
\cs_new:Npn \__physicx_xquantity_aux:nnnn #1#2#3#4
  {
    \l__physicx_quantity_pre_tl
    \bool_lazy_or:nnTF
      { \tl_if_empty_p:N \l__physicx_quantity_left_size_tl }
      { \tl_if_empty_p:N \l__physicx_quantity_right_size_tl }
      { #1 #2 #3 #4 }
      {
        \bool_lazy_or:nnTF
          { \token_if_eq_meaning_p:NN \l__physicx_quantity_left_size_tl \left }
          { \token_if_eq_meaning_p:NN \l__physicx_quantity_right_size_tl \right }
          { \physicx_left: #1 #2 #3 \physicx_right: #4 }
          {
            \physicx_left:N \l__physicx_quantity_left_size_tl #1 #2
            #3
            \physicx_right:N \l__physicx_quantity_right_size_tl #4
          }
      }
    \l__physicx_quantity_post_tl
  }
\NewDocumentCommand \xquantity { } { \physicx_xquantity:nn }
\cs_generate_variant:Nn \__physicx_xquantity_aux:nnnn { oooo }
\NewDocumentCommand \newxquantity { m o o m m }
  {
    \IfNoValueTF {#2}
      {
        \cs_set:Npn \__physicx_new_xquantity_aux:w ##1
          { \newcommand ##1 }
      }
      {
        \IfNoValueTF {#3}
          {
            \cs_set:Npn \__physicx_new_xquantity_aux:w ##1
              { \newcommand ##1 [#2] }
          }
          {
            \cs_set:Npn \__physicx_new_xquantity_aux:w ##1
              { \newcommand ##1 [#2] [#3] }
          }
      }
    \exp_args:Nc \__physicx_new_xquantity_aux:w
      { \cs_to_str:N #1~star }
      { \physicx_xquantity:nn { #4 , noauto } {#5} }
    \exp_args:Nc \__physicx_new_xquantity_aux:w
      { \cs_to_str:N #1~unstar }
      { \physicx_xquantity:nn { #4 } {#5} }
    \exp_args:NNx \newcommand #1
      {
        \exp_not:N \@ifstar
        \exp_not:c { \cs_to_str:N #1~star }
        \exp_not:c { \cs_to_str:N #1~unstar }
      }
  }
\NewDocumentCommand \NewXQuantity { m m m m }
  {
    \NewDocumentCommand #1 { s #2 }
      {
        \IfBooleanTF {##1}
          { \physicx_xquantity:nn { #3 , noauto } {#4} }
          { \physicx_xquantity:nn { #3 } {#4} }
      }
  }
\NewXQuantity \qxqty { O{} m } { #2 } {#3}
\NewXQuantity \txqty { O{p} O{} m } { type={#2}, #3 } {#4}
```

(*End definition for* `\physicx_xquantity:nn`*,* `\newxquantity`*, and* `\NewXQuantity`*. These functions are documented on page* **??**.)

1.2.2 Legacy quantity

\physicx_declare_legacy_quantity:nnNn

\@declarequantitycmd

```
\tl_new:N \physicxtmp
\tl_new:N \l__physicx_cmd_noauto_body_tl
\bool_new:N \l__physicx_cmd_noauto_body_bool
\tl_new:N \l__physicx_cmd_auto_body_tl
\bool_new:N \l__physicx_cmd_auto_body_bool
\tl_new:N \l__physicx_cmd_arg_spec_tl
\int_new:N \l__physicx_cmd_arg_int
\cs_new:Npn \__physicx_declare_init:nnn #1#2#3
  {
    \tl_clear:N \l__physicx_cmd_noauto_body_tl
    \tl_clear:N \l__physicx_cmd_auto_body_tl
    \tl_clear:N \l__physicx_cmd_arg_spec_tl
    \int_set:Nn \l__physicx_cmd_arg_int {#1}
    \bool_set:Nn \l__physicx_cmd_noauto_body_bool {#2}
    \bool_set:Nn \l__physicx_cmd_auto_body_bool {#3}
  }
% noauto, auto, cmd, body
\cs_new:Npn \physicx_declare_legacy_quantity:nnNn #1#2#3#4
  {
    \__physicx_declare_init:nnn { 3 } {#1} {#2}
    \__physicx_declare_legacy_quantity_aux:nw #4
      \q_recursion_tail \q_recursion_tail \q_recursion_stop
    \__physicx_declare_legacy_quantity_aux:NcVVV
      #3 { \cs_to_str:N #3 ~ body }
      \l__physicx_cmd_arg_spec_tl
      \l__physicx_cmd_noauto_body_tl
      \l__physicx_cmd_auto_body_tl
  }
% arg spec, pre, body to replace(start from #4), post
\cs_new:Npn \__physicx_declare_legacy_quantity_aux:nnnn #1#2#3#4
  {
    \int_incr:N \l__physicx_cmd_arg_int
    \if_int_compare:w \l__physicx_cmd_arg_int < 10 \exp_stop_f:
      \tl_put_right:Nn \l__physicx_cmd_arg_spec_tl {#1}
      \tl_set:Nx \l__physicx_tmp_tl
        {
          {
          \exp_not:N \tl_if_novalue_p:n
          {
            \if_case:w \l__physicx_cmd_arg_int \exp_stop_f:
```

```
      \or: \or: \or:
      \or: \exp_not:n {##4} \or: \exp_not:n {##5} \or: \exp_not:n {##6}
      \or: \exp_not:n {##7} \or: \exp_not:n {##8} \or: \exp_not:n {##9}
      \fi:
     }
     }
    }
   \if_bool:N \l__physicx_cmd_noauto_body_bool
    \tl_put_right:No \l__physicx_cmd_noauto_body_tl { \l__physicx_tmp_tl }
    \tl_put_right:Nn \l__physicx_cmd_noauto_body_tl
     {
      {
       % if is ‘.’, use none
       \str_if_eq:nnTF {#2} {.} {} {#2}
       #3
       \str_if_eq:nnTF {#4} {.} {} {#4}
      }
     }
   \fi:
   \if_bool:N \l__physicx_cmd_auto_body_bool
    \tl_put_right:No \l__physicx_cmd_auto_body_tl { \l__physicx_tmp_tl }
    \tl_put_right:Nn \l__physicx_cmd_auto_body_tl
     { { ##1 #2 #3 ##2 #4 } }
   \fi:
  \fi:
 }
\cs_new:Npn \__physicx_declare_legacy_quantity_aux:nw #1#2
 {
  \quark_if_recursion_tail_stop:n {#1}
  \quark_if_recursion_tail_stop:n {#2}
  \__physicx_declare_legacy_quantity_aux:nnnn {#1} #2
  \__physicx_declare_legacy_quantity_aux:nw
 }
\cs_new:Npn \__physicx_declare_legacy_quantity_aux:NNnnn #1#2#3#4#5
 {
  \__physicx_nauto_case:nnnn
   { \use_i:nn } { \use_ii:nn } { \use_i:nn } { \use_i:nn }
   {
    \cs_set_protected:Npn #1
     {
      \peek_charcode_ignore_spaces:NTF \let
       { #2 } { #2 [ \physicx_left: ] \physicx_right: }
     }
    \DeclareDocumentCommand #2 { O{##2} m s #3 }
     {
      \IfBooleanTF { ##3 }
       { \bool_case_false:n {#4} }
       { \bool_case_false:n {#5} }
     }
   }
   {
    \cs_set_protected:Npn #1
     { #2 \c_empty_tl \c_empty_tl }
    \DeclareDocumentCommand #2 { m m s #3 }
```

```
          { \bool_case_false:n {#4} }
      }
  }
\cs_generate_variant:Nn \__physicx_declare_legacy_quantity_aux:NNnnn { NcVVV }
\cs_new:Npn \__physicx_nauto_case:nnnn #1#2#3#4
  {
    \bool_if:NTF \l__physicx_cmd_noauto_body_bool
      {
        \bool_if:NTF \l__physicx_cmd_auto_body_bool
          {#1} {#2}
      }
      {
        \bool_if:NTF \l__physicx_cmd_auto_body_bool
          {#3} {#4}
      }
  }
\cs_set_protected:Npn \@declarequantitycmd
  { \physicx_declare_legacy_quantity:nnNn }
```

(*End definition for* \physicx_declare_legacy_quantity:nnNn *and* \@declarequantitycmd. *These functions are documented on page* **??**.)

\quantity
\evaluated
\matrixquantity
\smallmatrixquantity

Redefine some macros in physics package.

```
\if_bool:N \g__physicx_reqty_bool
\physicx_declare_legacy_quantity:nnNn
  \c_true_bool \c_true_bool \quantity
  {
    { !g   } { { \{      } { #4 } { \}      } }
    { !o   } { { [       } { #5 } { ]       } }
    { !d() } { { (       } { #6 } { )       } }
    { !d|| } { { \vert   } { #7 } { \vert   } }
    { !d<> } { { \langle } { #8 } { \rangle } }
    { !d== } { { \Vert   } { #9 } { \Vert   } }
  }
\physicx_declare_legacy_quantity:nnNn
  \c_true_bool \c_true_bool \evaluated
  {
    { !g   } { { . } { #4 \nobreak } { \vert } }
    { !d[| } { { [ } { #5 \nobreak } { \vert } }
    { !d(| } { { ( } { #6 \nobreak } { \vert } }
  }
\physicx_declare_legacy_quantity:nnNn
  \c_true_bool \c_false_bool \matrixquantity
  {
    { !g }
      {
        { \IfBooleanT{#3}{\left\{} }
        { \begin{matrix} #4 \end{matrix} }
        { \IfBooleanT{#3}{\right\}} }
      }
    { !o }   { {\begin{bmatrix} } {#5} { \end{bmatrix} } }
    { !d() }
      {
        { \IfBooleanTF{#3}{\left\lgroup}{\left(} }
        { \begin{matrix} #6 \end{matrix} }
        { \IfBooleanTF{#3}{\right\rgroup}{\right)} }
      }
    { !d|| } { { \begin{vmatrix} } {#7} { \end{vmatrix} } }
    { !d<> } { { \left\langle } { \begin{matrix} #8 \end{matrix} } { \right\rangle } }
    { !d== } { { \begin{Vmatrix} } {#9} { \end{Vmatrix} } }
  }
\physicx_declare_legacy_quantity:nnNn
  \c_true_bool \c_false_bool \smallmatrixquantity
  {
    { !g } { { \left\{ } { \begin{smallmatrix} #4 \end{smallmatrix} } { \right\} } }
    { !o } { {\left[} { \begin{smallmatrix} #5 \end{smallmatrix} } {\right]} }
    { !d() }
      {
        { \IfBooleanTF{#3}{\left\lgroup}{\left(} }
        { \begin{smallmatrix} #6 \end{smallmatrix} }
        { \IfBooleanTF{#3}{\right\rgroup}{\right)} }
      }
    { !d|| } { {\left\vert} { \begin{smallmatrix} #7 \end{smallmatrix} } {\right\vert} }
    { !d<> } { {\left\langle} { \begin{smallmatrix} #8 \end{smallmatrix} } {\right\rangle} }
    { !d== } { {\left\Vert} { \begin{smallmatrix} #9 \end{smallmatrix} } {\right\Vert} }
  }
\fi:
```

(*End definition for* `\quantity` *and others. These functions are documented on page* **??**.)

\physicx_declare_legacy_paren:NnnnNNn
\@declareparencmd

```
%% cmd, arg spec, replace(start from #6), pre, left, right, post
\cs_new:Npn \physicx_declare_legacy_paren:NnnnNNn #1#2#3#4#5#6#7
  {
    \DeclareDocumentCommand #1 { s t\big t\Big t\bigg t\Bigg #2 }
      {
        \bool_case_true:nF
          {
            { \bool_if_p:n {##2} } { #4 \physicx_left:N \bigl  #5 #3 \physicx_right:N \bigr
            { \bool_if_p:n {##3} } { #4 \physicx_left:N \Bigl  #5 #3 \physicx_right:N \Bigr
            { \bool_if_p:n {##4} } { #4 \physicx_left:N \biggl #5 #3 \physicx_right:N \biggr
            { \bool_if_p:n {##5} } { #4 \physicx_left:N \Biggl #5 #3 \physicx_right:N \Biggr
          }
          {
            \IfBooleanTF {##1}
              { #4       #5 #3        #6 #7 }
              { #4 \physicx_left: #5 #3 \physicx_right: #6 #7 }
          }
      }
  }
\cs_set_protected:Npn \@declareparencmd
  { \physicx_declare_legacy_paren:NnnnNNn }
```

(*End definition for* `\physicx_declare_legacy_paren:NnnnNNn` *and* `\@declareparencmd`*. These functions are documented on page* **??**.)

\qty
\mqty
\smqty
\pqty
\bqty
\vqty
\Bqty
\absolutevalue
\eval
\abs
\norm
\order
\oorder

Redefine some macros in physics package.

```
\if_bool:N \g__physicx_reqty_bool
\physicx_option_or:nnT { compat } { short }
```

```
  {
    \cs_set:Npn \qty { \quantity }
    \physicx_declare_legacy_paren:NnnnNNn \pqty { m } {#6} { } ( ) { }
    \physicx_declare_legacy_paren:NnnnNNn \bqty { m } {#6} { } [ ] { }
    \physicx_declare_legacy_paren:NnnnNNn \vqty { m } {#6} { } \vert \vert { }
    \physicx_declare_legacy_paren:NnnnNNn \Bqty { m } {#6} { } \{ \} { }
  }
\physicx_declare_legacy_paren:NnnnNNn \absolutevalue
  { m } {#6} { } \vert \vert { }
\physicx_option_or:nnT { compat } { short }
  {
    \cs_set:Npn \eval { \evaluated }
    \cs_set:Npn \abs { \absolutevalue }
  }
\physicx_declare_legacy_paren:NnnnNNn \norm
  { m } {#6} { } \lVert \rVert { }
\physicx_compat:TF
  {
    \physicx_declare_legacy_paren:NnnnNNn \order
      { m } {#6} { \c_physicx_Order_tl } ( ) { }
  }
  {
    \physicx_declare_legacy_paren:NnnnNNn \order
      { m } {#6} { \c_physicx_order_tl } ( ) { }
  }
\physicx_declare_legacy_paren:NnnnNNn \commutator
  { m m } { #6 , #7 } { } [ ] { }
\physicx_option_or:nnT { compat } { short }
  { \cs_set:Npn \comm { \commutator } }
\physicx_declare_legacy_paren:NnnnNNn \poissonbracket
  { m m } { #6 , #7 } { } \{ \} { }
\physicx_option_or:nnT { compat } { short }
  {
    \cs_set:Npn \pb { \poissonbracket }
    \cs_set:Npn \anticommutator { \poissonbracket }
    \cs_set:Npn \acomm { \poissonbracket }
  }
\fi:
\physicx_declare_legacy_paren:NnnnNNn \OOrder
  { m } {#6} { \c_physicx_Order_tl } ( ) { }
\physicx_declare_legacy_paren:NnnnNNn \oorder
  { m } {#6} { \c_physicx_order_tl } ( ) { }
```

(*End definition for \qty and others. These functions are documented on page* **??**.)

1.3 Matrix things

1.3.1 Matrix auxillary functions

```
\cs_new_nopar:Npn \__physicx_matrix_calc:nn #1#2
  {
    \int_set:Nn \l__physicx_matrix_rows_int
      { \int_max:nn {#1} \l__physicx_matrix_rows_int }
    \int_set:Nn \l__physicx_matrix_cols_int
      { \int_max:nn {#2} \l__physicx_matrix_cols_int }
  }
% use matrix element
\cs_new_nopar:Npn \physicx_matrix_use_r_c:nn #1#2
  {
    \if_cs_exist:w l__physicx_matrix_r@#1_c@#2_tl \cs_end:
      \exp_not:v { l__physicx_matrix_r@#1_c@#2_tl }
    \else:
      \exp_not:o { \physicxempty }
    \fi:
  }
% set matrix element, check or not
\cs_new_nopar:Npn \__physicx_matrix_set_r_c_nock:nnn #1#2
  { \tl_set:cn { l__physicx_matrix_r@#1_c@#2_tl } }
\cs_new_nopar:Npn \__physicx_matrix_set_r_c_ckig:nnn #1#2#3
  {
    \tl_if_eq:nnF {#3} { \PHYSICXIGNORE }
      { \tl_set:cn { l__physicx_matrix_r@#1_c@#2_tl } {#3} }
  }
\cs_new_nopar:Npn \__physicx_matrix_set_r_c_ckep:nnn #1#2#3
  {
    \tl_if_empty:nTF {#3}
      { \tl_set:co { l__physicx_matrix_r@#1_c@#2_tl } { \physicxempty } }
      { \tl_set:cn { l__physicx_matrix_r@#1_c@#2_tl } {#3} }
  }
\cs_new_nopar:Npn \__physicx_matrix_set_r_c_ckigep:nnn #1#2#3
  {
    \tl_if_eq:nnF {#3} { \PHYSICXIGNORE }
      {
        \tl_if_empty:nTF {#3}
          { \tl_set:co { l__physicx_matrix_r@#1_c@#2_tl } { \physicxempty } }
          { \tl_set:cn { l__physicx_matrix_r@#1_c@#2_tl } {#3} }
      }
  }
\cs_set_eq:NN \__physicx_matrix_set_r_c_ckall:nnn
  \__physicx_matrix_set_r_c_ckigep:nnn
\cs_new_eq:NN \physicx_matrix_set_r_c:nnn
  \__physicx_matrix_set_r_c_nock:nnn
% align, cr, sep symbol
\str_const:Nn \physicx@align { , }
\str_const:Nn \physicx@cr { ; }
\str_const:Nn \physicx@sep { , }
\bool_new:N \l__physicx_matrix_infinite_bool
\bool_new:N \l__physicx_matrix_dotrow_bool
\bool_new:N \l__physicx_matrix_dotcol_bool
\tl_new:N \l__physicx_matrix_array_tl
\tl_new:N \l__physicx_matrix_body_tl
\int_new:N \l__physicx_matrix_rows_int
\int_new:N \l__physicx_matrix_cols_int
\tl_new:N \l__physicx_matrix_main_tl
\clist_new:N \l__physicx_matrix_diag_clist
\clist_new:N \l__physicx_matrix_item_clist
\bool_new:N \l__physicx_matrix_diag_bool
\seq_new:N \l__physicx_row_list_seq
\seq_new:N \l__physicx_col_list_seq
```

```
% expand input
\cs_new_eq:NN \__physicx_expand:w \exp_not:o
%% main, row iterate, col iterate
\cs_new_nopar:Npn \physicx@matrixelement #1#2#3 { #1 \sb { #2 #3 } }
\cs_new_nopar:Npn \__physicx_matrix_row_iterate:n #1 { #1 }
\tl_new:N \l__physicx_matrix_last_row_tl
\tl_new:N \l__physicx_matrix_last_col_tl
\cs_new_nopar:Npn \__physicx_matrix_col_iterate:n #1 { #1 }
\cs_new_nopar:Npn \__physicx_matrix_begin:w { }
\cs_new_nopar:Npn \__physicx_matrix_end:w { }
\cs_new_eq:NN \__physicx_matrix_autocalc:nn \use_none:nn
\bool_new:N \l__physicx_matrix_expand_element_bool
% when element is empty use \physicxempty
\tl_new:N \physicxempty
% save ‘element-except’ key’s value
\tl_new:N \physicxexcept
\tl_new:N \l__physicx_matrix_args_tl
\tl_new:N \l__physicx_matrix_after_begin_tl
\tl_new:N \l__physicx_matrix_after_end_tl
\bool_new:N \l__physicx_matrix_transpose_bool
\bool_new:N \l__physicx_matrix_enhanced_bool
\dim_new:N \l__physicx_matrix_sep_dim
\cs_new:Npn \__physicx_adi:nnn #1#2#3 { #1#2#3 }
\tl_new:N \l__physicx_matrix_beginning_tl
\tl_new:N \l__physicx_matrix_ending_tl
```

1.3.2 Matrix keys

```
\keys_define:nn { physicx }
  { matrix .code:n = \keys_set:nn { physicx/matrix } {#1} }
\keys_define:nn { physicx/matrix }
  {
    array .tl_set:N = \l__physicx_matrix_array_tl ,
    expand .choice: ,
    expand / none .code:n =
      \cs_set_eq:NN \__physicx_expand:w \exp_not:o ,
    expand / text-expand .code:n =
      \cs_set_eq:NN \__physicx_expand:w \text_expand:n ,
    expand / f .code:n =
      \cs_set_eq:NN \__physicx_expand:w \exp_not:f ,
    expand / romanual .meta:n = { expand = f } ,
    expand / x .code:n =
      \cs_set_eq:NN \__physicx_expand:w \use:n ,
    expand / edef .meta:n = { expand = x } ,
    rows .int_set:N = \l__physicx_matrix_rows_int ,
    cols .int_set:N = \l__physicx_matrix_cols_int ,
    auto-update .choice: ,
    auto-update / true .code:n =
      \cs_set_eq:NN \__physicx_matrix_autocalc:nn \__physicx_matrix_calc:nn ,
    auto-update / false .code:n =
      \cs_set_eq:NN \__physicx_matrix_autocalc:nn \use_none:nn ,
    auto-update .default:n = true ,
    main .tl_set:N = \l__physicx_matrix_main_tl ,
    row-list .code:n =
      \seq_set_split:Non \l__physicx_row_list_seq { \physicx@sep } {#1} ,
    col-list .code:n =
      \seq_set_split:Non \l__physicx_col_list_seq { \physicx@sep } {#1} ,
    infinite .bool_set:N = \l__physicx_matrix_infinite_bool ,
    infinite .default:n = true ,
    !infinite .code:n =
      \bool_set_inverse:N \l__physicx_matrix_infinite_bool ,
    element-code .cs_set:Np = \physicx@matrixelement #1#2#3 ,
    element-code* .choice: ,
    element-code* / except-empty .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \tl_if_empty:nTF {##1}
            {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
        } ,
    element-code* / except-blank .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \tl_if_blank:nTF {##1}
            {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
        } ,
    element-code* / except-dots .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \tl_if_in:nnTF { \cdots\vdots\ldots\ddots } {##1}
            {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
        } ,
    element-code* / except-tl .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \tl_if_in:onTF { \physicxexcept } {##1}
            {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
        } ,
    element-code* / except-regex .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \exp_args:No \regex_match:nnTF { \physicxexcept } {##1}
            {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
        } ,
    element-code* / only-regex .code:n =
      \cs_set_eq:NN \__physicx_matrix_element_aux:nnn
        \physicx@matrixelement
      \cs_set:Npn \physicx@matrixelement ##1##2##3
        {
          \exp_args:No \regex_match:nnTF { \physicxexcept } {##1}
            { \__physicx_matrix_element_aux:nnn {##1} {##2} {##3} }
            {##1}
        } ,
    element-code* / unknown .code:n =
      \cs_set:Npx \physicx@matrixelement { \exp_not:c {#1} },
    element-except .tl_set:N = \physicxexcept ,
    element-except+  .code:n =
      \tl_put_right:Nn \physicxexcept {#1} ,
    expand-element .bool_set:N = \l__physicx_matrix_expand_element_bool ,
    expand-element .default:n = true ,
    empty .tl_set:N = \physicxempty ,
    check .choice: ,
    check / none .code:n =
      \cs_set_eq:NN \physicx_matrix_set_r_c:nnn
        \__physicx_matrix_set_r_c_nock:nnn ,
    check / empty .code:n =
      \cs_set_eq:NN \physicx_matrix_set_r_c:nnn
        \__physicx_matrix_set_r_c_ckep:nnn ,
    check / ignore .code:n =
      \cs_set_eq:NN \physicx_matrix_set_r_c:nnn
        \__physicx_matrix_set_r_c_ckig:nnn ,
    check / igep .code:n =
      \cs_set_eq:NN \physicx_matrix_set_r_c:nnn
        \__physicx_matrix_set_r_c_ckigep:nnn ,
    check / all .code:n =
      \cs_set_eq:NN \physicx_matrix_set_r_c:nnn
        \__physicx_matrix_set_r_c_ckall:nnn ,
    check .default:n = all ,
    row-iterate .cs_set:Np = \__physicx_matrix_row_iterate:n #1 ,
    col-iterate .cs_set:Np = \__physicx_matrix_col_iterate:n #1 ,
    last-row .tl_set:N = \l__physicx_matrix_last_row_tl ,
    last-col .tl_set:N = \l__physicx_matrix_last_col_tl ,
    diag .clist_set:N = \l__physicx_matrix_diag_clist ,
    diag+ .code:n =
      \clist_put_right:Nn \l__physicx_matrix_diag_clist {#1} ,
    diag-now .code:n = \physicx_matrix_diag_parse:n {#1} ,
    diag-data .code:n = \__physicx_matrix_set_data:nn { diag } {#1} ,
    diag-data+ .code:n = \__physicx_matrix_add_data:nn { diag } {#1} ,
    item .clist_set:N = \l__physicx_matrix_item_clist ,
    item+ .code:n =
    \clist_put_right:Nn \l__physicx_matrix_item_clist {#1} ,
    item-now .code:n = \physicx_matrix_item_parse:n {#1} ,
    item-data .code:n = \__physicx_matrix_set_data:nn { item } {#1} ,
    item-data+ .code:n = \__physicx_matrix_add_data:nn { item } {#1} ,
    check-range .choice: ,
    check-range / true .code:n = \physicx_parse_range_check: ,
    check-range / false .code:n = \physicx_parse_range_nocheck: ,
    check-range .default:n = true ,
    begin .tl_set:N = \__physicx_matrix_begin:w ,
    end   .tl_set:N = \__physicx_matrix_end: ,
    args    .code:n =
      \tl_set:Nn \l__physicx_matrix_args_tl { [#1] } ,
    args* .tl_set:N = \l__physicx_matrix_args_tl ,
    after-begin .tl_set:N = \l__physicx_matrix_after_begin_tl ,
    after-begin+  .code:n =
      { \tl_put_right:Nn \l__physicx_matrix_after_begin_tl {#1} } ,
    after-end   .tl_set:N = \l__physicx_matrix_after_end_tl ,
    after-end+    .code:n =
      { \tl_put_right:Nn \l__physicx_matrix_after_end_tl {#1} } ,
    sepdim .dim_set:N = \l__physicx_matrix_sep_dim ,
    type .multichoice: ,
    saveto .tl_set:N = \l__physicx_matrix_save_tl ,
    saveto* .code:n =
      \tl_set:No \l__physicx_matrix_save_tl { \cs:w #1 \cs_end: } ,
    transpose .bool_set:N = \l__physicx_matrix_transpose_bool ,
    transpose .default:n = true ,
    ' .meta:n = { transpose = true } ,
    T .meta:n = { transpose = true } ,
    MaxMatrixCols .int_set:N = \c@MaxMatrixCols ,
    enhanced .bool_set:N = \l__physicx_matrix_enhanced_bool ,
    enhanced .default:n = true ,
    !enhanced .code:n =
      \bool_set_inverse:N \l__physicx_matrix_enhanced_bool ,
    cr .tl_set:N = \physicx@cr ,
    align .tl_set:N = \physicx@align ,
    sep .tl_set:N = \physicx@sep ,
    adi-order .choice: ,
    adi-order / adi .code:n = \cs_set:Nn \__physicx_adi:nnn {##1##2##3} ,
    adi-order / dia .code:n = \cs_set:Nn \__physicx_adi:nnn {##2##3##1} ,
    adi-order / iad .code:n = \cs_set:Nn \__physicx_adi:nnn {##3##1##2} ,
    adi-order / aid .code:n = \cs_set:Nn \__physicx_adi:nnn {##1##3##2} ,
    adi-order / ida .code:n = \cs_set:Nn \__physicx_adi:nnn {##3##2##1} ,
    adi-order / dai .code:n = \cs_set:Nn \__physicx_adi:nnn {##2##1##3} ,
    beginning .tl_set:N = \l__physicx_matrix_beginning_tl ,
    beginning+ .code:n =
      \tl_put_right:Nn \l__physicx_matrix_beginning_tl {#1} ,
    ending .tl_set:N = \l__physicx_matrix_ending_tl ,
    ending+ .code:n =
      \tl_put_right:Nn \l__physicx_matrix_ending_tl {#1} ,

    settype .code:n = \setmatrixtype #1 ,

    unknown .code:n =
      \physicx_search_also:nnF
        {
          physicx/matrix/type ,
          physicx/matrix/expand ,
          physicx/matrix/element-code* ,
        }
        {#1}
        {
          \exp_args:No \physicx_if_num:nTF { \l_keys_key_str }
            {
              \keys_set:nx { physicx/matrix }
                { MaxMatrixCols = \l_keys_key_str }
            }
            {
              \msg_error:nnxx { physicx } { unknown-key }
                \l_keys_path_str { physicx/matrix }
            }
        } ,
  }
```

\physicx_matrix_new_type:nnn
\physicx_matrix_new_type:nn
\setmatrixtype

```
\cs_new:Npn \physicx_matrix_new_type:nnn #1#2#3
  { \physicx_new_type:nnn { matrix } {#1} { begin={#2} , end={#3} } }
\cs_new:Npn \physicx_matrix_new_type:nn
  { \physicx_new_type:nnn { matrix } }
\NewDocumentCommand \setmatrixtype { s >{ \TrimSpaces } m }
  {
    \IfBooleanTF {#1}
      { \physicx_matrix_new_type:nn {#2} }
      { \physicx_matrix_new_type:nnn {#2} }
  }
```

(*End definition for* \physicx_matrix_new_type:nnn *,* \physicx_matrix_new_type:nn *, and* \setmatrixtype*. These functions are documented on page* **??***.*)

A few types.

```
\setmatrixtype {m} {\begin{matrix}} {\end{matrix}}
\setmatrixtype {p} {\begin{pmatrix}} {\end{pmatrix}}
\setmatrixtype {b} {\begin{bmatrix}} {\end{bmatrix}}
\setmatrixtype {B} {\begin{Bmatrix}} {\end{Bmatrix}}
\setmatrixtype {v} {\begin{vmatrix}} {\end{vmatrix}}
\setmatrixtype {V} {\begin{Vmatrix}} {\end{Vmatrix}}
\setmatrixtype {sm} {\begin{smallmatrix}} {\end{smallmatrix}}
\physicx_mathtools:T
  {
    \setmatrixtype {m*} {\begin{matrix*}} {\end{matrix*}}
    \setmatrixtype {p*} {\begin{pmatrix*}} {\end{pmatrix*}}
    \setmatrixtype {b*} {\begin{bmatrix*}} {\end{bmatrix*}}
    \setmatrixtype {B*} {\begin{Bmatrix*}} {\end{Bmatrix*}}
    \setmatrixtype {v*} {\begin{vmatrix*}} {\end{vmatrix*}}
    \setmatrixtype {V*} {\begin{Vmatrix*}} {\end{Vmatrix*}}
    \setmatrixtype {sm*} {\begin{smallmatrix*}} {\end{smallmatrix*}}
    \setmatrixtype {sp} {\begin{psmallmatrix}} {\end{psmallmatrix}}
    \setmatrixtype {sb} {\begin{bsmallmatrix}} {\end{bsmallmatrix}}
    \setmatrixtype {sB} {\begin{Bsmallmatrix}} {\end{Bsmallmatrix}}
    \setmatrixtype {sv} {\begin{vsmallmatrix}} {\end{vsmallmatrix}}
    \setmatrixtype {sV} {\begin{Vsmallmatrix}} {\end{Vsmallmatrix}}
    \setmatrixtype {sp*} {\begin{psmallmatrix*}} {\end{psmallmatrix*}}
    \setmatrixtype {sb*} {\begin{bsmallmatrix*}} {\end{bsmallmatrix*}}
    \setmatrixtype {sB*} {\begin{Bsmallmatrix*}} {\end{Bsmallmatrix*}}
    \setmatrixtype {sv*} {\begin{vsmallmatrix*}} {\end{vsmallmatrix*}}
    \setmatrixtype {sV*} {\begin{Vsmallmatrix*}} {\end{Vsmallmatrix*}}
  }
```

\setmatrixdata Set matrix data, one can use '...-data' key to use it.

```
\cs_new_protected_nopar:Npn \setmatrixdata #1#2
  { \clist_set:cn { physicx@ #1 data@ #2 } }
\cs_new_protected_nopar:Npn \__physicx_matrix_set_data:nn #1#2
  {
    \clist_clear:c { l__physicx_matrix_ #1 _clist }
    \__physicx_matrix_add_data:nn {#1} {#2}
  }
\cs_new_protected_nopar:Npn \__physicx_matrix_add_data:nn #1#2
  {
    \clist_map_inline:nn {#2}
      {
        \clist_concat:ccc
          { l__physicx_matrix_ #1 _clist }
          { l__physicx_matrix_ #1 _clist }
          { physicx@ #1 data@ #2 }
      }
  }
```

(*End definition for* `\setmatrixdata`. *This function is documented on page* **??**.)

Initial settings.

```
\keys_set:nn { physicx/matrix }
  {
    type = m ,
    saveto = ? ,
  }
```

`\qxmatrix`

```
%% basicly, https://tex.stackexchange.com/questions/486154/is-there-a-way-to-define-
   xmatmnm-in-the-physics-package, but changed some
% #1 = boolean, saveto matrix
% #2 = star, infinite
% #3 = options
% #4 = letter for the entries
% #5 = number of rows
% #6 = number of explicit rows, default = 3
% #7 = number of columns
% #8 = number of explicit columns, default = 3
\DeclareDocumentCommand \qxmatrix { t= s O{type=p} m m O{3} m O{3} }
  {
    \group_begin:
    \IfBooleanTF { #2 }
      { \bool_set_true:N \l__physicx_matrix_infinite_bool }
      { \bool_set_false:N \l__physicx_matrix_infinite_bool }
    \int_set:Nn \l__physicx_matrix_rows_int {#6}
    \int_set:Nn \l__physicx_matrix_cols_int {#8}
    \IfBooleanTF {#1}
      { \keys_set:nn { physicx/matrix } { #3 , saveto = \physicxtmp } }
      { \keys_set:nn { physicx/matrix } {#3} }
    \physicx_qxmatrix:nnn {#4} {#5} {#7}
    \__physicx_matrix_save_or_print:
    \group_end:
  }
\cs_new_protected:Nn \physicx_qxmatrix:nnn
  {
    \bool_if:NTF \l__physicx_matrix_expand_element_bool
      {
        \cs_set_eq:NN \__physicx_qxmatrix_appto_body:nnn
          \__physicx_matrix_appto_body_e:nnn
      }
      {
        \cs_set_eq:NN \__physicx_qxmatrix_appto_body:nnn
          \__physicx_matrix_appto_body_ne:nnn
      }
    % clear the variable containing the body of the matrix
    \tl_clear:N \l__physicx_matrix_body_tl
    % set the tentative number of explicit rows
    \physicx_if_num:nTF { #2 }
      {% number of rows is an integer
        \int_compare:nTF { #2 <= \l__physicx_matrix_rows_int }
        {% if #2 <= rows, we don’t want a row of dots
          \bool_set_false:N \l__physicx_matrix_dotrow_bool
          \int_set:Nn \l__physicx_matrix_rows_int { #2 }
        }
        {% we want a row of dots
          \bool_set_true:N \l__physicx_matrix_dotrow_bool
        }
      }
      {% number of rows is symbolic, we want a row of dots
        \bool_set_true:N \l__physicx_matrix_dotrow_bool
      }
    % set the tentative number of explicit columns
    \physicx_if_num:nTF { #3 }
      {% number of cols is an integer
        \int_compare:nTF { #3 <= \l__physicx_matrix_cols_int }
          {% if #3 <= cols, we don’t want a column of dots
            \bool_set_false:N \l__physicx_matrix_dotcol_bool
            \int_set:Nn \l__physicx_matrix_cols_int { #3 }
          }
          {% we want a column of dots
            \bool_set_true:N \l__physicx_matrix_dotcol_bool
          }
      }
      {% number of columns is symbolic, we want a column of dots
        \bool_set_true:N \l__physicx_matrix_dotcol_bool
      }
    % loop through the rows
    \int_step_inline:nn { \l__physicx_matrix_rows_int }
      {
        % add the first entry in the row
        %%\tl_put_right:Nn \l__physicx_matrix_body_tl { #1\sb{##1 1} }
        \__physicx_qxmatrix_appto_body:nnn {#1} {##1} { 1 }
        % add the further entries in the explicit columns
        \int_step_inline:nnn { 2 } { \l__physicx_matrix_cols_int }
          {
            %%\tl_put_right:Nn \l__physicx_matrix_body_tl { & #1\sb{##1 ####1} }
            \tl_put_right:Nn \l__physicx_matrix_body_tl { & }
            \__physicx_qxmatrix_appto_body:nnn {#1} {##1} {####1}
          }
        % if we have a column of dots, add \cdots and the last entry
        \bool_if:NT \l__physicx_matrix_dotcol_bool
          {
            %%\tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots & #1\sb{##1 #3} }
            \tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots & }
            \__physicx_qxmatrix_appto_body:nnn {#1} {##1} {#3}
          }
        % infinite matrix, add \cdots
        \bool_if:NT \l__physicx_matrix_infinite_bool
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots } }
        \if_int_compare:w ##1 = \l__physicx_matrix_rows_int
          \scan_stop:
        \else:
          % finish up the row
          \tl_put_right:Nx \l__physicx_matrix_body_tl { \\[\dim_use:N \l__physicx_matrix_sep
        \fi:
      }
    % finish up the rows
    \bool_if:NT \l__physicx_matrix_dotrow_bool
      {
        % finish up the row
        \tl_put_right:Nx \l__physicx_matrix_body_tl { \\[\dim_use:N \l__physicx_matrix_sep_d
        % if we have a row of dots, fill it in
        \tl_put_right:Nn \l__physicx_matrix_body_tl { \vdots }
        \prg_replicate:nn { \l__physicx_matrix_cols_int - 1 }
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { & \vdots } }
        \bool_if:NT \l__physicx_matrix_dotcol_bool
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { & \ddots & \vdots } }
        \tl_put_right:Nx \l__physicx_matrix_body_tl { \\[\dim_use:N \l__physicx_matrix_sep_d
        % fill the last row
        %%\tl_put_right:Nn \l__physicx_matrix_body_tl { #1\sb{#2 1} }
        \__physicx_qxmatrix_appto_body:nnn {#1} {#2} { 1 }
        \int_step_inline:nnn { 2 } { \l__physicx_matrix_cols_int }
          {
            %%\tl_put_right:Nn \l__physicx_matrix_body_tl { & #1\sb{#2 ##1} }
            \tl_put_right:Nn \l__physicx_matrix_body_tl { & }
            \__physicx_qxmatrix_appto_body:nnn {#1} {#2} {##1}
          }
        \bool_if:NT \l__physicx_matrix_dotcol_bool
          {
            %%\tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots & #1\sb{#2 #3} }
            \tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots & }
            \__physicx_qxmatrix_appto_body:nnn {#1} {#2} {#3}
          }
        % if the matrix is infinite, add a further column with \cdots
        \bool_if:NT \l__physicx_matrix_infinite_bool
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { & \cdots } }
      }
    % if the matrix is infinite, add a final row
    \bool_if:NT \l__physicx_matrix_infinite_bool
      {
        % finish up the row
        \tl_put_right:Nx \l__physicx_matrix_body_tl { \\[\dim_use:N \l__physicx_matrix_sep_d
        \tl_put_right:Nn \l__physicx_matrix_body_tl { \vdots }
        \prg_replicate:nn { \l__physicx_matrix_cols_int - 1 }
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { & \vdots } }
        \bool_if:NT \l__physicx_matrix_dotcol_bool
          { \tl_put_right:Nn \l__physicx_matrix_body_tl { &  & \vdots } }
        \tl_put_right:Nn \l__physicx_matrix_body_tl { & \ddots }
        % update cols
        \bool_if:NTF \l__physicx_matrix_dotcol_bool
          { \tex_advance:D \l__physicx_matrix_cols_int by 3 }
          { \tex_advance:D \l__physicx_matrix_cols_int by 2 }
      }
  }
```

(*End definition for* `\qxmatrix`*. This function is documented on page* **??**.)

`\physicx_matrix_diag_parse:n`
`\physicx_matrix_diag_parse:o`

Parse ‘diag...’ keys.

```
\cs_new:Npn \physicx_matrix_diag_parse:n #1
  {
    \keyval_parse:nnn
      \__physicx_matrix_diag_parse_aux:n
      \__physicx_matrix_diag_parse_aux:nn
      {#1}
  }
\cs_generate_variant:Nn \physicx_matrix_diag_parse:n { o }
\cs_new:Npn \__physicx_matrix_diag_parse_aux:n #1
  {
    \str_case_e:nnF {#1}
      {
        { auto-update }
          {
            \cs_set_eq:NN \__physicx_matrix_diag_calc:nn
              \__physicx_matrix_calc:nn
          }
        { noauto-update }
          {
            \cs_set_eq:NN \__physicx_matrix_diag_calc:nn \use_none:nn
          }
        { true }
          {
            \bool_set_true:N \l__physicx_matrix_diag_bool
            \cs_set_eq:NN \__physicx_diagonalmatrix_diag_main:
              \__physicx_diagonalmatrix_set_diag:
          }
        { false }
          {
            \bool_set_false:N \l__physicx_matrix_diag_bool
            \cs_set_eq:NN \__physicx_diagonalmatrix_diag_main:
              \__physicx_diagonalmatrix_no_diag:
          }
      }
      { \msg_error:nnn { physicx } { diag-key } {#1} }
  }
\cs_new:Npn \__physicx_matrix_diag_parse_aux:nn #1#2
  {
    \tl_set:Nn \l__physicx_tmpdiag_tl {#2}
    \tl_set:Nx \l__physicx_tmpdiag_tl
      { \__physicx_expand:w \l__physicx_tmpdiag_tl }
    \seq_set_split:NVV \l__physicx_tmpdiag_seq \physicx@sep \l__physicx_tmpdiag_tl
    \tl_if_head_eq_charcode:nNTF {#1} '
      {
        \exp_args:Nf \__physicx_matrix_diag_parse_aux_anti:n
          { \tl_tail:n {#1} }
      }
      { \__physicx_matrix_diag_parse_aux_regu:n {#1} }
  }
\cs_new:Npn \__physicx_diagonalmatrix_set_diag:
  {
    \int_zero:N \l__physicx_matrix_cols_int
    \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
      {
        \int_incr:N \l__physicx_matrix_cols_int
        \physicx_matrix_set_r_c:nnn {##1} {##1} {##2}
      }
    \int_set_eq:NN \l__physicx_matrix_rows_int
      \l__physicx_matrix_cols_int
  }
\cs_new:Npn \__physicx_diagonalmatrix_no_diag:
  {
    \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
      { \physicx_matrix_set_r_c:nnn {##1} {##1} {##2} }
    \__physicx_matrix_diag_calc:nn
      { \seq_count:N \l__physicx_tmpdiag_seq }
      { \seq_count:N \l__physicx_tmpdiag_seq }
  }
\cs_new_eq:NN \__physicx_diagonalmatrix_diag_main:
  \__physicx_diagonalmatrix_no_diag:
\cs_new:Npn \__physicx_matrix_diag_parse_aux_regu:n #1
  {
    \if_int_compare:w #1 = 0 \exp_stop_f:
      \__physicx_diagonalmatrix_diag_main:
    \else:
      \if_int_compare:w #1 > 0 \exp_stop_f:
        \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
          {
            \physicx_matrix_set_r_c:nnn
              {##1} { \int_eval:n { ##1 + #1 } } {##2}
          }
        \__physicx_matrix_diag_calc:nn
          { \seq_count:N \l__physicx_tmpdiag_seq }
          { \seq_count:N \l__physicx_tmpdiag_seq + #1 }
      \else:
        \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
          {
            \physicx_matrix_set_r_c:nnn
              { \int_eval:n { ##1 - #1 } } {##1} {##2}
          }
        \__physicx_matrix_diag_calc:nn
          { \seq_count:N \l__physicx_tmpdiag_seq - #1 }
          { \seq_count:N \l__physicx_tmpdiag_seq }
      \fi:
    \fi:
  }
\cs_new:Npn \__physicx_matrix_diag_parse_aux_anti:n #1
  {
    \if_int_compare:w #1 = 0 \exp_stop_f:
      \__physicx_matrix_diag_calc:nn
        { \seq_count:N \l__physicx_tmpdiag_seq }
        { \seq_count:N \l__physicx_tmpdiag_seq }
      \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
        {
          \physicx_matrix_set_r_c:nnn
            {##1}
            { \int_eval:n { \l__physicx_matrix_cols_int - ##1 + 1 } }
            {##2}
        }
    \else:
      \if_int_compare:w #1 > 0 \exp_stop_f:
        \__physicx_matrix_diag_calc:nn
          { \seq_count:N \l__physicx_tmpdiag_seq }
          { \seq_count:N \l__physicx_tmpdiag_seq + #1 }
        \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
          {
            \physicx_matrix_set_r_c:nnn
              {##1}
              { \int_eval:n { \l__physicx_matrix_cols_int - ##1 - #1 + 1 } }
              {##2}
          }
      \else:
        \__physicx_matrix_diag_calc:nn
          { \seq_count:N \l__physicx_tmpdiag_seq - #1 }
          { \seq_count:N \l__physicx_tmpdiag_seq }
        \seq_map_indexed_inline:Nn \l__physicx_tmpdiag_seq
          {
            \physicx_matrix_set_r_c:nnn
              { \int_eval:n { ##1 - #1 } }
              { \int_eval:n { \l__physicx_matrix_cols_int - ##1 + 1 } }
              {##2}
          }
      \fi:
    \fi:
  }
\cs_new:Npn \__physicx_matrix_diag_calc:nn
  { \__physicx_matrix_autocalc:nn }
```

(*End definition for* `\physicx_matrix_diag_parse:n`*. This function is documented on page* **??**.)

`\physicx_matrix_item_parse:n`
`\physicx_matrix_item_parse:o`

Parse ‘item...’ keys.

```
\cs_new:Npn \physicx_matrix_item_parse:n #1
  {
    \clist_set_eq:NN \l__physicx_item_ignore_clist \c_empty_clist
    \keyval_parse:NNn
      \__physicx_matrix_item_parse_aux:n
      \__physicx_matrix_item_parse_aux:nn
      {#1}
  }
\cs_generate_variant:Nn \physicx_matrix_item_parse:n { o }
\cs_new:Npn \__physicx_matrix_item_parse_aux:n #1 { }
\cs_new:Npn \__physicx_matrix_item_parse_aux:nn #1#2
  {
    \tl_set:Nn \l__physicx_tmpitem_tl {#2}
    \tl_set:Nx \l__physicx_tmpitem_tl
      { \__physicx_expand:w \l__physicx_tmpitem_tl }
    \physicx_parse_range:nxN \l__physicx_matrix_rows_int
      { \use_i:nn #1 } \l__physicx_tmp_rownum_seq
    \physicx_parse_range:nxN \l__physicx_matrix_cols_int
      { \use_ii:nn #1 } \l__physicx_tmp_colnum_seq
    \exp_args:No \tl_if_eq:nnTF
      { \l__physicx_tmpitem_tl } { \PHYSICXIGNORE }
      {
        \seq_map_inline:Nn \l__physicx_tmp_rownum_seq
          {
            \seq_map_inline:Nn \l__physicx_tmp_colnum_seq
              {
                \clist_put_right:Nn \l__physicx_item_ignore_clist { [##1][####1] }
              }
          }
      }
      {
        \seq_map_inline:Nn \l__physicx_tmp_rownum_seq
          {
            \seq_map_inline:Nn \l__physicx_tmp_colnum_seq
              {
                \clist_if_in:NnF \l__physicx_item_ignore_clist { [##1][####1] }
                  {
                    \exp_args:Nnno \physicx_matrix_set_r_c:nnn
                      {##1} {####1} { \l__physicx_tmpitem_tl }
                  }
              }
          }
      }
  }
```

(*End definition for* `\physicx_matrix_item_parse:n`*. This function is documented on page* **??**.)

\physicx_matrix_array_parse:n
\physicx_matrix_array_parse:o

Parse 'array...' keys.

```
\cs_new:Npn \physicx_matrix_array_parse:n #1
  {
    \tl_set:Nn \l__physicx_tmparr_tl {#1}
    \tl_set:Nx \l__physicx_tmparr_tl
      { \__physicx_expand:w \l__physicx_tmparr_tl }
    \seq_set_split:NVV \l__physicx_matrix_tmparr_r_sep \physicx@cr \l__physicx_tmparr_tl
    \__physicx_matrix_autocalc:nn
      { \seq_count:N \l__physicx_matrix_tmparr_r_sep }
      { 0 }
    \seq_map_indexed_inline:Nn \l__physicx_matrix_tmparr_r_sep
      {
        \seq_set_split:Non \l__physicx_matrix_tmparr_c_sep { \physicx@align } {##2}
        \__physicx_matrix_autocalc:nn
          { 0 }
          { \seq_count:N \l__physicx_matrix_tmparr_c_sep }
        \seq_map_indexed_inline:Nn \l__physicx_matrix_tmparr_c_sep
          {
            \physicx_matrix_set_r_c:nnn {##1} {####1} {####2}
          }
      }
  }
\cs_generate_variant:Nn \physicx_matrix_array_parse:n { o }
```

(*End definition for* `\physicx_matrix_array_parse:n`*. This function is documented on page* **??***.*)

`\physicx_matrix_array_parse_main:`

Process ‘main’ key.

```
\cs_new:Npn \physicx_matrix_array_parse_main:
  {
    \int_step_inline:nn \l__physicx_matrix_rows_int
      {
        \int_step_inline:nn \l__physicx_matrix_cols_int
          {
            \exp_args:Nnno \physicx_matrix_set_r_c:nnn
              {##1} {####1} \l__physicx_matrix_main_tl
          }
      }
  }
```

(*End definition for* `\physicx_matrix_array_parse_main:`*. This function is documented on page* **??***.*)

`\__physicx_if_can_num:n`

Test if can num, one can use `\int_eval:n`, `\fp_eval:n`, and `\inteval`, `\fpeval` in xfp package (if loaded).

```
\prg_new_conditional:Npnn \__physicx_if_can_num:n #1 { T, F, TF }
  {
    \physicx_if_num:nTF {#1}
      { \prg_return_true: }
      {
        \bool_case_true:nTF
          {
            { \tl_if_head_eq_meaning_p:nN {#1} \int_eval:n } { }
            { \tl_if_head_eq_meaning_p:nN {#1} \fp_eval:n } { }
            {
              \bool_lazy_and_p:nn
                { \cs_if_exist_p:N \inteval }
                { \tl_if_head_eq_meaning_p:nN {#1} \inteval }
            } { }
            {
              \bool_lazy_and_p:nn
                { \cs_if_exist_p:N \fpeval }
                { \tl_if_head_eq_meaning_p:nN {#1} \fpeval }
            } { }
          }
          { \prg_return_true: }
          { \prg_return_false: }
      }
  }
```

(*End definition for* `\__physicx_if_can_num:n`.)

`\diagonalmatrix` Define `\diagonalmatrix`.

```
\DeclareDocumentCommand \diagonalmatrix { t= t+ O{} m }
  {
    \group_begin:
    \IfBooleanTF {#1}
      { \keys_set:nn { physicx/matrix } { #3 , saveto = \physicxtmp } }
      { \keys_set:nn { physicx/matrix } { #3 } }
    \physicx_construct:nnn { }
      {
        \physicx_matrix_diag_parse:o \l__physicx_matrix_diag_clist
        \tl_if_empty:nF {#4}
          {
            \__physicx_if_keyval:nTF {#4}
              { \physicx_matrix_diag_parse:n { true, #4 } }
              { \physicx_matrix_diag_parse:n { true, 0 = {#4} } }
          }
      }
      { \physicx_matrix_item_parse:o \l__physicx_matrix_item_clist }
    \bool_lazy_or:nnTF
      { \bool_if_p:n {#2} }
      { \bool_if_p:N \l__physicx_matrix_enhanced_bool }
      {
        \bool_if:NTF \l__physicx_matrix_expand_element_bool
          {
            \cs_set_eq:NN \__physicx_diagonalmatrix_enhanced:nnn
              \__physicx_matrix_appto_body_e:off
          }
          {
            \cs_set_eq:NN \__physicx_diagonalmatrix_enhanced:nnn
              \__physicx_matrix_appto_body_ne:off
          }
        \use_i_ii:nnn
      }
      { \use_i:nn }
      \__physicx_matrix_transpose:N
        \__physicx_diagonalmatrix_generate_enhanced_body:NNN
        \__physicx_diagonalmatrix_generate_body:NNN
    \__physicx_matrix_save_or_print:
    \group_end:
  }
\cs_new:Npn \__physicx_diagonalmatrix_generate_enhanced_body:NNN #1#2#3
  {
    \__physicx_matrix_generate_body:NNNN #1#2#3
      \__physicx_diagonalmatrix_enhanced:nnn
  }
\cs_new:Npn \__physicx_diagonalmatrix_generate_body:NNN #1#2#3
  {
    \int_step_inline:nn { #1 - 1 }
      {
        \int_step_inline:nn { #2 - 1 }
          {
            \tl_put_right:Nx \l__physicx_matrix_body_tl
```

```
              {
                \exp_after:wN
                \physicx_matrix_use_r_c:nn
                #3 {{##1}} {{####1}} &
              }
          }
        \tl_put_right:Nx \l__physicx_matrix_body_tl
          {
            \exp_after:wN
            \physicx_matrix_use_r_c:nn
            #3 {{##1}} {{ \int_use:N #2 }} \\[\dim_use:N \l__physicx_matrix_sep_dim]
          }
      }
    \int_step_inline:nn { #2 - 1 }
      {
        \tl_put_right:Nx \l__physicx_matrix_body_tl
          {
            \exp_after:wN
            \physicx_matrix_use_r_c:nn
            #3 {{ \int_use:N #1 }} {{##1}} &
          }
      }
    \tl_put_right:Nx \l__physicx_matrix_body_tl
      {
        \exp_after:wN
        \physicx_matrix_use_r_c:nn
        #3 {{ \int_use:N #1 }} {{ \int_use:N #2 }}
      }
  }
```

(*End definition for* `\diagonalmatrix`*. This function is documented on page* **??**.)

__physicx_declare_init:

```
\cs_new:Npn \__physicx_matrix_enhanced_init:
  {
    \seq_if_empty:NF \l__physicx_row_list_seq
      {
        \bool_set_true:N \l__physicx_matrix_expand_element_bool
        \cs_set_nopar:Npn \__physicx_matrix_row_iterate:n ##1
          { \seq_item:Nn \l__physicx_row_list_seq {##1} }
      }
    \seq_if_empty:NF \l__physicx_col_list_seq
      {
        \bool_set_true:N \l__physicx_matrix_expand_element_bool
        \cs_set_nopar:Npn \__physicx_matrix_col_iterate:n ##1
          { \seq_item:Nn \l__physicx_col_list_seq {##1} }
      }
  }
```

(*End definition for* `\__physicx_declare_init:`.)

\commamatrix Define \commamatrix.

```
\DeclareDocumentCommand \commamatrix { t= t+ O{} m }
  {
    \group_begin:
```

```
    \keys_set:nn { physicx/matrix } {#3}
    \tl_if_empty:nF {#4}
      { \keys_set:nn { physicx/matrix } { array = {#4} } }
    \IfBooleanT {#1}
      { \keys_set:nn { physicx/matrix } { saveto = \physicxtmp } }
    \tl_set:Nx \l__physicx_matrix_array_tl
      { \__physicx_expand:w \l__physicx_matrix_array_tl }
    \bool_lazy_or:nnTF
      { \bool_if_p:n {#2} }
      { \bool_if_p:N \l__physicx_matrix_enhanced_bool }
      { \__physicx_commamatrix_enhanced: }
      {
        \tl_replace_all:Nox \l__physicx_matrix_array_tl
          { \physicx@cr } { \\[\dim_use:N \l__physicx_matrix_sep_dim] }
        \tl_replace_all:Non \l__physicx_matrix_array_tl
          { \physicx@align } { & }
        \tl_set_eq:NN \l__physicx_matrix_body_tl
          \l__physicx_matrix_array_tl
      }
    \__physicx_matrix_save_or_print:
    \group_end:
  }
\cs_new_nopar:Npn \__physicx_matrix_save_or_print:
  {
    \exp_after:wN \token_if_cs:NTF \l__physicx_matrix_save_tl
      {
        \exp_after:wN \tl_gset_eq:NN
          \l__physicx_matrix_save_tl
          \l__physicx_matrix_body_tl
      }
      {
        \if_int_compare:w \c@MaxMatrixCols < \l__physicx_matrix_cols_int
          \int_set_eq:NN \c@MaxMatrixCols \l__physicx_matrix_cols_int
        \fi:
        \exp_after:wN \__physicx_matrix_begin:w \l__physicx_matrix_args_tl \l__physicx_matri
        \l__physicx_matrix_body_tl
        \__physicx_matrix_end: \l__physicx_matrix_after_end_tl
      }
  }
\cs_new:Npn \__physicx_commamatrix_enhanced:
  {
    \tl_clear:N \l__physicx_matrix_body_tl
    \int_zero:N \l__physicx_tmpa_int
    \seq_set_split:NVV \l__physicx_tmp_seq \physicx@cr
      \l__physicx_matrix_array_tl
    \int_set:Nn \l__physicx_matrix_rows_int
      { \seq_count:N \l__physicx_tmp_seq }
    \__physicx_matrix_enhanced_init:
    \bool_if:NTF \l__physicx_matrix_expand_element_bool
      {
        \seq_map_tokens:Nn \l__physicx_tmp_seq
          {
            \int_incr:N \l__physicx_tmpa_int
            \exp_args:NV \__physicx_commamatrix_enhanced_aux:nNn
```

```
              \l__physicx_tmpa_int \__physicx_commamatrix_enhanced_aux_e:nnn
          }
      }
      {
        \seq_map_tokens:Nn \l__physicx_tmp_seq
          {
            \int_incr:N \l__physicx_tmpa_int
            \exp_args:NV \__physicx_commamatrix_enhanced_aux:nNn
              \l__physicx_tmpa_int \__physicx_commamatrix_enhanced_aux_ne:nnn
          }
      }
  }
\cs_new:Npn \__physicx_commamatrix_enhanced_aux:nNn #1#2#3
  {
    \seq_set_split:Non \l__physicx_tmp_col_seq
      { \physicx@align } {#3}
    \seq_set_eq:NN \l__physicx_tmp_coled_seq \c_empty_seq
    \seq_map_indexed_inline:Nn \l__physicx_tmp_col_seq
      { #2 {##2} {#1} {##1} }
    \tl_put_right:Nx \l__physicx_matrix_body_tl
      {
        \seq_use:Nn \l__physicx_tmp_coled_seq { & }
        \if_int_compare:w \l__physicx_matrix_rows_int = #1
          \scan_stop:
        \else:
          \\[\dim_use:N \l__physicx_matrix_sep_dim]
        \fi:
      }
  }
\cs_new:Npn \__physicx_commamatrix_enhanced_aux_e:nnn #1#2#3
  {
    \seq_put_right:Nx \l__physicx_tmp_coled_seq
      {
        \text_expand:n % \text_expand:n do the magic thing, but slower
          {
            \physicx@matrixelement { #1 }
              { \__physicx_matrix_row_iterate:n {#2} }
              { \__physicx_matrix_col_iterate:n {#3} }
          }
      }
  }
\cs_new:Npn \__physicx_commamatrix_enhanced_aux_ne:nnn #1#2#3
  {
    \seq_put_right:No \l__physicx_tmp_coled_seq
      {
        \physicx@matrixelement {#1}
          { \__physicx_matrix_row_iterate:n {#2} }
          { \__physicx_matrix_col_iterate:n {#3} }
      }
  }
```

(*End definition for* `\commamatrix`. *This function is documented on page* **??**.)

`\generalmatrix` Define `\generalmatrix`.

```
\DeclareDocumentCommand \generalmatrix { t= t+ s m }
  {
    \IfBooleanTF {#2}
      {
        \group_begin:
        \IfBooleanTF {#1}
          { \keys_set:nn { physicx/matrix } { #4 , saveto = \physicxtmp } }
          { \keys_set:nn { physicx/matrix } {#4} }
        \bool_set:Nn \l__physicx_matrix_infinite_bool {#3}
        \physicx_construct:nnn
          {
            \tl_if_empty:NTF \l__physicx_matrix_main_tl
              {
                \physicx_matrix_array_parse:o \l__physicx_matrix_array_tl
              }
              { \physicx_matrix_array_parse_main: }
          }
          { \physicx_matrix_diag_parse:o \l__physicx_matrix_diag_clist }
          { \physicx_matrix_item_parse:o \l__physicx_matrix_item_clist }
        \__physicx_generalmatrix:
        \__physicx_matrix_save_or_print:
        \group_end:
      }
      {
        \IfBooleanTF {#1}
          { \IfBooleanTF {#3} { } { \use_i_ii:nnn } }
          { \IfBooleanTF {#3} { \use_i:nn } { \use_i:nnn } }
        \qxmatrix = * [#4]
      }
  }
\cs_new:Npn \__physicx_generalmatrix:
  {
    \bool_if:NTF \l__physicx_matrix_expand_element_bool
      {
        \cs_set_eq:NN \__physicx_generalmatrix_generate:nnn
          \__physicx_matrix_appto_body_e:off
      }
      {
        \cs_set_eq:NN \__physicx_generalmatrix_generate:nnn
          \__physicx_matrix_appto_body_ne:off
      }
    \__physicx_matrix_transpose:N
      \__physicx_matrix_generate_body:NNNN
      \__physicx_generalmatrix_generate:nnn
  }
```

(*End definition for* `\generalmatrix`. *This function is documented on page* **??**.)

__physicx_matrix_generate_body:NNNN

```
% row, col, \use:nn or \use_ii_i:nn, appto body cmd
\cs_new:Npn \__physicx_matrix_generate_body:NNNN #1#2#3#4
  {
    \__physicx_matrix_enhanced_init:
    \int_step_inline:nn { #1 - 1 }
```

```
      {
        \int_step_inline:nn { #2 - 1 }
          {
            \tl_set:Nx \l__physicx_tmp_tl
              {
                \exp_after:wN
                \physicx_matrix_use_r_c:nn
                #3 {{##1}} {{####1}}
              }
            #4 \l__physicx_tmp_tl {##1} {####1}
            \tl_put_right:Nn \l__physicx_matrix_body_tl { & }
          }
        \tl_set:Nx \l__physicx_tmp_tl
          {
            \exp_after:wN
            \physicx_matrix_use_r_c:nn
            #3 {{##1}} {{ \int_use:N #2 }}
          }
        #4 \l__physicx_tmp_tl {##1} { \int_use:N #2 }
        \tl_put_right:Nx \l__physicx_matrix_body_tl
          { \\[\dim_use:N \l__physicx_matrix_sep_dim] }
      }
    \int_step_inline:nn { #2 - 1 }
      {
        \tl_set:Nx \l__physicx_tmp_tl
          {
            \exp_after:wN
            \physicx_matrix_use_r_c:nn
            #3 {{ \int_use:N #1 }} {{##1}}
          }
        #4 \l__physicx_tmp_tl { \int_use:N #1 } {##1}
        \tl_put_right:Nn \l__physicx_matrix_body_tl { & }
      }
    \tl_set:Nx \l__physicx_tmp_tl
      {
        \exp_after:wN
        \physicx_matrix_use_r_c:nn
        #3 {{ \int_use:N #1 }} {{ \int_use:N #2 }}
      }
    #4 \l__physicx_tmp_tl { \int_use:N #1 } { \int_use:N #2 }
  }
```

(*End definition for* __*physicx_matrix_generate_body:NNNN*.)

__physicx_matrix_appto_body_e:nnn
__physicx_matrix_appto_body_e:off
__physicx_matrix_appto_body_e:xff
__physicx_matrix_appto_body_ne:nnn
__physicx_matrix_appto_body_ne:off
__physicx_matrix_appto_body_ne:xff

```
\cs_new:Npn \__physicx_matrix_appto_body_e:nnn #1#2#3
  {
    \tl_put_right:Nx \l__physicx_matrix_body_tl
      {
        \text_expand:n
          {
            \physicx@matrixelement {#1}
              { \__physicx_matrix_row_iterate:n {#2} }
              { \__physicx_matrix_col_iterate:n {#3} }
          }
      }
  }
\cs_generate_variant:Nn \__physicx_matrix_appto_body_e:nnn { off, xff }
\cs_new:Npn \__physicx_matrix_appto_body_ne:nnn #1#2#3
  {
    \tl_put_right:No \l__physicx_matrix_body_tl
      {
        \physicx@matrixelement {#1}
          { \__physicx_matrix_row_iterate:n {#2} }
          { \__physicx_matrix_col_iterate:n {#3} }
      }
  }
\cs_generate_variant:Nn \__physicx_matrix_appto_body_ne:nnn { off, xff }
```

(*End definition for* `\__physicx_matrix_appto_body_e:nnn` *and* `\__physicx_matrix_appto_body_ne:nnn`.)

__physicx_matrix_transpose:N

```
\cs_new:Npn \__physicx_matrix_transpose:N #1 % generate body command
  {
    \bool_if:NTF \l__physicx_matrix_transpose_bool
      {
        #1
          \l__physicx_matrix_cols_int
          \l__physicx_matrix_rows_int
          \use_ii_i:nn
      }
      {
        #1
          \l__physicx_matrix_rows_int
          \l__physicx_matrix_cols_int
          \use:nn
      }
  }
```

(*End definition for* `\__physicx_matrix_transpose:N`.)

\physicx_construct:nnn Final construct. First is adi (array, diag, item), then 'last-col', 'last-row' and dots, then infinite, then 'ending' key.

```
\cs_new:Npn \physicx_construct:nnn #1#2#3
  {
    \l__physicx_matrix_beginning_tl
    \__physicx_adi:nnn {#1} {#2} {#3}
    \tl_if_empty:NF \l__physicx_matrix_last_col_tl
      {
        \int_incr:N \l__physicx_matrix_cols_int
        \__physicx_matrix_last_aux_c:
        \int_incr:N \l__physicx_matrix_cols_int
      }
    \tl_if_empty:NF \l__physicx_matrix_last_row_tl
      {
        \int_incr:N \l__physicx_matrix_rows_int
        \__physicx_matrix_last_aux_r:
        \int_incr:N \l__physicx_matrix_rows_int
      }
    \bool_lazy_or:nnF
      { \tl_if_empty_p:N \l__physicx_matrix_last_row_tl }
      { \tl_if_empty_p:N \l__physicx_matrix_last_col_tl }
      {
        \physicx_matrix_set_r_c:nnn
          { \int_eval:n { \l__physicx_matrix_rows_int - 1 } }
          { \int_eval:n { \l__physicx_matrix_cols_int - 1 } }
          { \ddots }
      }
    \bool_if:NT \l__physicx_matrix_infinite_bool
      {
        \int_incr:N \l__physicx_matrix_rows_int
        \int_incr:N \l__physicx_matrix_cols_int
        \__physicx_matrix_last_aux_c:
        \__physicx_matrix_last_aux_r:
        \physicx_matrix_set_r_c:nnn
          { \int_use:N \l__physicx_matrix_rows_int }
          { \int_use:N \l__physicx_matrix_cols_int }
          { \ddots }
      }
    \l__physicx_matrix_ending_tl
  }
\cs_new:Npn \__physicx_matrix_last_aux_c:
  {
    \int_step_inline:nn \l__physicx_matrix_rows_int
      {
        \physicx_matrix_set_r_c:nnn
          {##1} { \int_use:N \l__physicx_matrix_cols_int }
          { \cdots }
      }
  }
\cs_new:Npn \__physicx_matrix_last_aux_r:
  {
    \int_step_inline:nn \l__physicx_matrix_cols_int
      {
        \physicx_matrix_set_r_c:nnn
          { \int_use:N \l__physicx_matrix_rows_int } {##1}
          { \vdots }
      }
  }
```

(*End definition for* `\physicx_construct:nnn`. *This function is documented on page* **??**.)

1.3.3 Define new matrix command

__physicx_new_matrix_cmd:NNN
\newgeneralmatrix
\NewGeneralMatrix
\newdiagonalmatrix
\NewDiagonalMatrix
\newcommamatrix
\NewCommaMatrix

```
\cs_new:Npn \__physicx_new_matrix_cmd:NNN #1#2#3
  {
    \NewDocumentCommand #2 { t+ m o o m m }
      {
        \IfBooleanTF {##1}
          {
            \IfNoValueTF {##3}
```

```
              { \newcommand ##2 { #1 + [##5] {##6} } }
              {
                \IfNoValueTF {##4}
                  { \newcommand ##2 [##3] { #1 + [##5] {##6} } }
                  { \newcommand ##2 [##3] [##4] { #1 + [##5] {##6} } }
              }
          }
          {
            \IfNoValueTF {##3}
              { \newcommand ##2 { #1 [##5] {##6} } }
              {
                \IfNoValueTF {##4}
                  { \newcommand ##2 [##3] { #1 [##5] {##6} } }
                  { \newcommand ##2 [##3] [##4] { #1 [##5] {##6} } }
              }
          }
      }
    \NewDocumentCommand #3 { t+ m m m m }
      {
        \IfBooleanTF {##1}
          { \NewDocumentCommand ##2 {##3} { #1 + [##4] {##5} } }
          { \NewDocumentCommand ##2 {##3} { #1   [##4] {##5} } }
      }
  }
\__physicx_new_matrix_cmd:NNN \diagonalmatrix \newdiagonalmatrix \NewDiagonalMatrix
\__physicx_new_matrix_cmd:NNN \commamatrix \newcommamatrix \NewCommaMatrix
\NewDocumentCommand \newgeneralmatrix { t+ m o o m }
  {
    \IfBooleanTF {#1}
      {
        \IfNoValueTF {#3}
          { \newcommand #2 { \generalmatrix + {#5} } }
          {
            \IfNoValueTF {#4}
              { \newcommand #2 [#3] { \generalmatrix + {#5} } }
              { \newcommand #2 [#3] [#4] { \generalmatrix + {#5} } }
          }
      }
      {
        \IfNoValueTF {#3}
          { \newcommand #2 { \generalmatrix {#5} } }
          {
            \IfNoValueTF {#4}
              { \newcommand #2 [#3] { \generalmatrix {#5} } }
              { \newcommand #2 [#3] [#4] { \generalmatrix {#5} } }
          }
      }
  }
\NewDocumentCommand \NewGeneralMatrix { t+ m m m }
  {
    \IfBooleanTF {#1}
      { \NewDocumentCommand #2 {#3} { \generalmatrix + {#4} } }
      { \NewDocumentCommand #2 {#3} { \generalmatrix   {#4} } }
  }
```

(*End definition for* `\__physicx_new_matrix_cmd:NNN` *and others. These functions are documented on page* **??**.)

```
⟨/package⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.