# RKHS

1.0

Generated by Doxygen 1.8.10

Thu Dec 15 2016 13:35:50

# Contents

# 1 Modules Index

## 1.1 Modules List

Here is a list of all modules with brief descriptions:

# 2 Data Type Index

## 2.1 Data Types List

Here are the data types with brief descriptions:

# 3 Module Documentation

## 3.1 file_io Module Reference

File input/output unit.

### Variables

- integer, parameter io_unit = 30

### 3.1.1 Detailed Description

File input/output unit.

### Author

Oliver T. Unke, University of Basel

This module contains the standard file i/o unit that is used by the RKHS module. In case the file unit conflicts with any existing code, it can be easily changed here.

### 3.1.2 Variable Documentation

#### 3.1.2.1 integer parameter file_io::io_unit = 30

Definition at line 48 of file RKHS.f90.

```
00048     integer, parameter :: io_unit = 30
```

## 3.2 kernel_bernoulli_2 Module Reference

Bernoulli polynomial kernel with n = 2.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)

### Variables

- integer, parameter m2 = 12
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 1d0/12d0
- real(kind(0d0)), parameter p22 = -1d0/24d0
- real(kind(0d0)), parameter p23 = -1d0/24d0
- real(kind(0d0)), parameter p24 = -1d0/12d0
- real(kind(0d0)), parameter p25 = 1d0/12d0
- real(kind(0d0)), parameter p26 = -1d0/4d0
- real(kind(0d0)), parameter p27 = 1d0/4d0
- real(kind(0d0)), parameter p28 = -1d0/24d0
- real(kind(0d0)), parameter p29 = -1d0/24d0
- real(kind(0d0)), parameter p210 = 1d0/6d0
- real(kind(0d0)), parameter p211 = 1d0/6d0
- real(kind(0d0)), parameter p212 = -1d0/4d0

#### 3.2.1 Detailed Description

Bernoulli polynomial kernel with n = 2.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the Bernoulli kernel with n = 2. Note that this kernel is only defined for values in the interval [0,1]. It is only applicable for periodic functions, such as sin(2∗pi∗x).

#### 3.2.2 Function/Subroutine Documentation

##### 3.2.2.1 pure real(kind(0d0)) function kernel_bernoulli_2::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 261 of file RKHS.f90.

```
00261         implicit none
00262         real(kind(0d0)), intent(in) :: x
00263         real(kind(0d0)), dimension(:), intent(in) :: par
00264         d2f21 = 0d0
```

##### 3.2.2.2 pure real(kind(0d0)) function kernel_bernoulli_2::d2f210 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 639 of file RKHS.f90.

```
00639         implicit none
00640         real(kind(0d0)), intent(in) :: x
00641         real(kind(0d0)), dimension(:), intent(in) :: par
00642         d2f210 = 6d0*x
```

##### 3.2.2.3 pure real(kind(0d0)) function kernel_bernoulli_2::d2f211 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 681 of file RKHS.f90.

```
00681         implicit none
00682         real(kind(0d0)), intent(in) :: x
00683         real(kind(0d0)), dimension(:), intent(in) :: par
00684         d2f211 = 0d0
```

#### 3.2.2.4 pure real(kind(0d0)) function kernel_bernoulli_2::d2f212 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 723 of file RKHS.f90.

```
00723         implicit none
00724         real(kind(0d0)), intent(in) :: x
00725         real(kind(0d0)), dimension(:), intent(in) :: par
00726         d2f212 = 2d0
```

#### 3.2.2.5 pure real(kind(0d0)) function kernel_bernoulli_2::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 303 of file RKHS.f90.

```
00303         implicit none
00304         real(kind(0d0)), intent(in) :: x
00305         real(kind(0d0)), dimension(:), intent(in) :: par
00306         d2f22 = 2d0
```

#### 3.2.2.6 pure real(kind(0d0)) function kernel_bernoulli_2::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 345 of file RKHS.f90.

```
00345         implicit none
00346         real(kind(0d0)), intent(in) :: x
00347         real(kind(0d0)), dimension(:), intent(in) :: par
00348         d2f23 = 0d0
```

#### 3.2.2.7 pure real(kind(0d0)) function kernel_bernoulli_2::d2f24 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 387 of file RKHS.f90.

```
00387         implicit none
00388         real(kind(0d0)), intent(in) :: x
00389         real(kind(0d0)), dimension(:), intent(in) :: par
00390         d2f24 = 6d0*x
```

#### 3.2.2.8 pure real(kind(0d0)) function kernel_bernoulli_2::d2f25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 429 of file RKHS.f90.

```
00429         implicit none
00430         real(kind(0d0)), intent(in) :: x
00431         real(kind(0d0)), dimension(:), intent(in) :: par
00432         d2f25 = 0d0
```

#### 3.2.2.9 pure real(kind(0d0)) function kernel_bernoulli_2::d2f26 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 471 of file RKHS.f90.

```
00471         implicit none
00472         real(kind(0d0)), intent(in) :: x
00473         real(kind(0d0)), dimension(:), intent(in) :: par
00474         d2f26 = 0d0
```

#### 3.2.2.10 pure real(kind(0d0)) function kernel_bernoulli_2::d2f27 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 513 of file RKHS.f90.

```
00513         implicit none
00514         real(kind(0d0)), intent(in) :: x
00515         real(kind(0d0)), dimension(:), intent(in) :: par
00516         d2f27 = 2d0
```

#### 3.2.2.11 pure real(kind(0d0)) function kernel_bernoulli_2::d2f28 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 555 of file RKHS.f90.

```
00555         implicit none
00556         real(kind(0d0)), intent(in) :: x
00557         real(kind(0d0)), dimension(:), intent(in) :: par
00558         d2f28 = 12d0*x**2
```

#### 3.2.2.12 pure real(kind(0d0)) function kernel_bernoulli_2::d2f29 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 597 of file RKHS.f90.

```
00597         implicit none
00598         real(kind(0d0)), intent(in) :: x
00599         real(kind(0d0)), dimension(:), intent(in) :: par
00600         d2f29 = 0d0
```

#### 3.2.2.13 pure real(kind(0d0)) function kernel_bernoulli_2::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 282 of file RKHS.f90.

```
00282         implicit none
00283         real(kind(0d0)), intent(in) :: x
00284         real(kind(0d0)), dimension(:), intent(in) :: par
00285         d2f31 = 0d0
```

#### 3.2.2.14 pure real(kind(0d0)) function kernel_bernoulli_2::d2f310 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 660 of file RKHS.f90.

```
00660         implicit none
00661         real(kind(0d0)), intent(in) :: x
00662         real(kind(0d0)), dimension(:), intent(in) :: par
00663         d2f310 = 0d0
```

#### 3.2.2.15 pure real(kind(0d0)) function kernel_bernoulli_2::d2f311 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 702 of file RKHS.f90.

```
00702         implicit none
00703         real(kind(0d0)), intent(in) :: x
00704         real(kind(0d0)), dimension(:), intent(in) :: par
00705         d2f311 = 6d0*x
```

#### 3.2.2.16 pure real(kind(0d0)) function kernel_bernoulli_2::d2f312 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 744 of file RKHS.f90.

```
00744         implicit none
00745         real(kind(0d0)), intent(in) :: x
00746         real(kind(0d0)), dimension(:), intent(in) :: par
00747         d2f312 = 2d0
```

#### 3.2.2.17 pure real(kind(0d0)) function kernel_bernoulli_2::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 324 of file RKHS.f90.

```
00324         implicit none
00325         real(kind(0d0)), intent(in) :: x
00326         real(kind(0d0)), dimension(:), intent(in) :: par
00327         d2f32 = 0d0
```

#### 3.2.2.18 pure real(kind(0d0)) function kernel_bernoulli_2::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 366 of file RKHS.f90.

```
00366         implicit none
00367         real(kind(0d0)), intent(in) :: x
00368         real(kind(0d0)), dimension(:), intent(in) :: par
00369         d2f33 = 2d0
```

#### 3.2.2.19 pure real(kind(0d0)) function kernel_bernoulli_2::d2f34 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 408 of file RKHS.f90.

```
00408         implicit none
00409         real(kind(0d0)), intent(in) :: x
00410         real(kind(0d0)), dimension(:), intent(in) :: par
00411         d2f34 = 0d0
```

#### 3.2.2.20 pure real(kind(0d0)) function kernel_bernoulli_2::d2f35 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 450 of file RKHS.f90.

```
00450         implicit none
00451         real(kind(0d0)), intent(in) :: x
00452         real(kind(0d0)), dimension(:), intent(in) :: par
00453         d2f35 = 6d0*x
```

#### 3.2.2.21 pure real(kind(0d0)) function kernel_bernoulli_2::d2f36 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 492 of file RKHS.f90.

```
00492         implicit none
00493         real(kind(0d0)), intent(in) :: x
00494         real(kind(0d0)), dimension(:), intent(in) :: par
00495         d2f36 = 2d0
```

#### 3.2.2.22 pure real(kind(0d0)) function kernel_bernoulli_2::d2f37 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 534 of file RKHS.f90.

```
00534         implicit none
00535         real(kind(0d0)), intent(in) :: x
00536         real(kind(0d0)), dimension(:), intent(in) :: par
00537         d2f37 = 0d0
```

#### 3.2.2.23 pure real(kind(0d0)) function kernel_bernoulli_2::d2f38 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 576 of file RKHS.f90.

```
00576         implicit none
00577         real(kind(0d0)), intent(in) :: x
00578         real(kind(0d0)), dimension(:), intent(in) :: par
00579         d2f38 = 0d0
```

#### 3.2.2.24 pure real(kind(0d0)) function kernel_bernoulli_2::d2f39 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 618 of file RKHS.f90.

```
00618         implicit none
00619         real(kind(0d0)), intent(in) :: x
00620         real(kind(0d0)), dimension(:), intent(in) :: par
00621         d2f39 = 12d0*x**2
```

#### 3.2.2.25 pure real(kind(0d0)) function kernel_bernoulli_2::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 231 of file RKHS.f90.

```
00231         implicit none
00232         real(kind(0d0)), intent(in) :: x1,x2
00233         real(kind(0d0)), dimension(:), intent(in) :: par
00234         !find larger/smaller of x1 and x2
00235         if(x1 <= x2) then
00236             d2k = -0.1d1 / 0.12d2 - x1 ** 2 / 0.2d1 + x2 * x1 - x2 ** 2 / 0.2d1 &
00237                   - x1 / 0.2d1 + x2 / 0.2d1
00238
00239         else
00240             d2k = -0.1d1 / 0.12d2 + x1 / 0.2d1 - x2 / 0.2d1 - x1 ** 2 / 0.2d1 &
00241                   + x2 * x1 - x2 ** 2 / 0.2d1
00242         end if
```

#### 3.2.2.26 pure real(kind(0d0)) function kernel_bernoulli_2::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 254 of file RKHS.f90.

```
00254         implicit none
00255         real(kind(0d0)), intent(in) :: x
00256         real(kind(0d0)), dimension(:), intent(in) :: par
00257         df21 = 1d0
```

#### 3.2.2.27 pure real(kind(0d0)) function kernel_bernoulli_2::df210 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 632 of file RKHS.f90.

```
00632         implicit none
00633         real(kind(0d0)), intent(in) :: x
00634         real(kind(0d0)), dimension(:), intent(in) :: par
00635         df210 = 3d0*x**2
```

#### 3.2.2.28 pure real(kind(0d0)) function kernel_bernoulli_2::df211 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 674 of file RKHS.f90.

```
00674         implicit none
00675         real(kind(0d0)), intent(in) :: x
00676         real(kind(0d0)), dimension(:), intent(in) :: par
00677         df211 = 1d0
```

#### 3.2.2.29 pure real(kind(0d0)) function kernel_bernoulli_2::df212 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 716 of file RKHS.f90.

```
00716         implicit none
00717         real(kind(0d0)), intent(in) :: x
00718         real(kind(0d0)), dimension(:), intent(in) :: par
00719         df212 = 2d0*x
```

#### 3.2.2.30 pure real(kind(0d0)) function kernel_bernoulli_2::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 296 of file RKHS.f90.

```
00296         implicit none
00297         real(kind(0d0)), intent(in) :: x
00298         real(kind(0d0)), dimension(:), intent(in) :: par
00299         df22 = 2d0*x
```

#### 3.2.2.31 pure real(kind(0d0)) function kernel_bernoulli_2::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 338 of file RKHS.f90.

```
00338         implicit none
00339         real(kind(0d0)), intent(in) :: x
00340         real(kind(0d0)), dimension(:), intent(in) :: par
00341         df23 = 0d0
```

#### 3.2.2.32 pure real(kind(0d0)) function kernel_bernoulli_2::df24 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 380 of file RKHS.f90.

```
00380         implicit none
00381         real(kind(0d0)), intent(in) :: x
00382         real(kind(0d0)), dimension(:), intent(in) :: par
00383         df24 = 3d0*x**2
```

#### 3.2.2.33 pure real(kind(0d0)) function kernel_bernoulli_2::df25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 422 of file RKHS.f90.

```
00422         implicit none
00423         real(kind(0d0)), intent(in) :: x
00424         real(kind(0d0)), dimension(:), intent(in) :: par
00425         df25 = 0d0
```

#### 3.2.2.34 pure real(kind(0d0)) function kernel_bernoulli_2::df26 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 464 of file RKHS.f90.

```
00464         implicit none
00465         real(kind(0d0)), intent(in) :: x
00466         real(kind(0d0)), dimension(:), intent(in) :: par
00467         df26 = 1d0
```

#### 3.2.2.35 pure real(kind(0d0)) function kernel_bernoulli_2::df27 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 506 of file RKHS.f90.

```
00506         implicit none
00507         real(kind(0d0)), intent(in) :: x
00508         real(kind(0d0)), dimension(:), intent(in) :: par
00509         df27 = 2d0*x
```

#### 3.2.2.36 pure real(kind(0d0)) function kernel_bernoulli_2::df28 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 548 of file RKHS.f90.

```
00548         implicit none
00549         real(kind(0d0)), intent(in) :: x
00550         real(kind(0d0)), dimension(:), intent(in) :: par
00551         df28 = 4d0*x**3
```

#### 3.2.2.37 pure real(kind(0d0)) function kernel_bernoulli_2::df29 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 590 of file RKHS.f90.

```
00590         implicit none
00591         real(kind(0d0)), intent(in) :: x
00592         real(kind(0d0)), dimension(:), intent(in) :: par
00593         df29 = 0d0
```

#### 3.2.2.38 pure real(kind(0d0)) function kernel_bernoulli_2::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 275 of file RKHS.f90.

```
00275         implicit none
00276         real(kind(0d0)), intent(in) :: x
00277         real(kind(0d0)), dimension(:), intent(in) :: par
00278         df31 = 1d0
```

#### 3.2.2.39 pure real(kind(0d0)) function kernel_bernoulli_2::df310 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 653 of file RKHS.f90.

```
00653         implicit none
00654         real(kind(0d0)), intent(in) :: x
00655         real(kind(0d0)), dimension(:), intent(in) :: par
00656         df310 = 1d0
```

#### 3.2.2.40 pure real(kind(0d0)) function kernel_bernoulli_2::df311 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 695 of file RKHS.f90.

```
00695         implicit none
00696         real(kind(0d0)), intent(in) :: x
00697         real(kind(0d0)), dimension(:), intent(in) :: par
00698         df311 = 3d0*x**2
```

#### 3.2.2.41 pure real(kind(0d0)) function kernel_bernoulli_2::df312 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 737 of file RKHS.f90.

```
00737         implicit none
00738         real(kind(0d0)), intent(in) :: x
00739         real(kind(0d0)), dimension(:), intent(in) :: par
00740         df312 = 2d0*x
```

#### 3.2.2.42 pure real(kind(0d0)) function kernel_bernoulli_2::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 317 of file RKHS.f90.

```
00317         implicit none
00318         real(kind(0d0)), intent(in) :: x
00319         real(kind(0d0)), dimension(:), intent(in) :: par
00320         df32 = 0d0
```

#### 3.2.2.43 pure real(kind(0d0)) function kernel_bernoulli_2::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 359 of file RKHS.f90.

```
00359         implicit none
00360         real(kind(0d0)), intent(in) :: x
00361         real(kind(0d0)), dimension(:), intent(in) :: par
00362         df33 = 2d0*x
```

#### 3.2.2.44 pure real(kind(0d0)) function kernel_bernoulli_2::df34 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 401 of file RKHS.f90.

```
00401         implicit none
00402         real(kind(0d0)), intent(in) :: x
00403         real(kind(0d0)), dimension(:), intent(in) :: par
00404         df34 = 0d0
```

#### 3.2.2.45 pure real(kind(0d0)) function kernel_bernoulli_2::df35 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 443 of file RKHS.f90.

```
00443         implicit none
00444         real(kind(0d0)), intent(in) :: x
00445         real(kind(0d0)), dimension(:), intent(in) :: par
00446         df35 = 3d0*x**2
```

#### 3.2.2.46 pure real(kind(0d0)) function kernel_bernoulli_2::df36 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 485 of file RKHS.f90.

```
00485         implicit none
00486         real(kind(0d0)), intent(in) :: x
00487         real(kind(0d0)), dimension(:), intent(in) :: par
00488         df36 = 2d0*x
```

#### 3.2.2.47 pure real(kind(0d0)) function kernel_bernoulli_2::df37 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 527 of file RKHS.f90.

```
00527         implicit none
00528         real(kind(0d0)), intent(in) :: x
00529         real(kind(0d0)), dimension(:), intent(in) :: par
00530         df37 = 1d0
```

#### 3.2.2.48 pure real(kind(0d0)) function kernel_bernoulli_2::df38 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 569 of file RKHS.f90.

```
00569         implicit none
00570         real(kind(0d0)), intent(in) :: x
00571         real(kind(0d0)), dimension(:), intent(in) :: par
00572         df38 = 0d0
```

#### 3.2.2.49 pure real(kind(0d0)) function kernel_bernoulli_2::df39 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 611 of file RKHS.f90.

```
00611         implicit none
00612         real(kind(0d0)), intent(in) :: x
00613         real(kind(0d0)), dimension(:), intent(in) :: par
00614         df39 = 4d0*x**3
```

#### 3.2.2.50 pure real(kind(0d0)) function kernel_bernoulli_2::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 215 of file RKHS.f90.

```
00215         implicit none
00216         real(kind(0d0)), intent(in) :: x1,x2
00217         real(kind(0d0)), dimension(:), intent(in) :: par
00218         !find larger/smaller of x1 and x2
00219         if(x1 <= x2) then
00220             dk = -x1 / 0.12d2 + x2 / 0.12d2 - x1 ** 3 / 0.6d1 + x1 ** 2 * x2 &
00221                  / 0.2d1 - x1 * x2 ** 2 / 0.2d1 + x2 ** 3 / 0.6d1 - x1 ** 2 / 0.4d1 &
00222                   + x2 * x1 / 0.2d1 - x2 ** 2 / 0.4d1
00223         else
00224             dk = x2 / 0.12d2 - x1 / 0.12d2 + x1 ** 2 / 0.4d1 - x2 * x1 / 0.2d1 + &
00225                  x2 ** 2 / 0.4d1 - x1 ** 3 / 0.6d1 + x2 ** 3 / 0.6d1 + x1 ** 2 * x2 &
00226                  / 0.2d1 - x1 * x2 ** 2 / 0.2d1
00227         end if
```

#### 3.2.2.51 pure real(kind(0d0)) function kernel_bernoulli_2::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 247 of file RKHS.f90.

```
00247         implicit none
00248         real(kind(0d0)), intent(in) :: x
00249         real(kind(0d0)), dimension(:), intent(in) :: par
00250         f21 = x
```

#### 3.2.2.52 pure real(kind(0d0)) function kernel_bernoulli_2::f210 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 625 of file RKHS.f90.

```
00625         implicit none
00626         real(kind(0d0)), intent(in) :: x
00627         real(kind(0d0)), dimension(:), intent(in) :: par
00628         f210 = x**3
```

#### 3.2.2.53 pure real(kind(0d0)) function kernel_bernoulli_2::f211 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 667 of file RKHS.f90.

```
00667         implicit none
00668         real(kind(0d0)), intent(in) :: x
00669         real(kind(0d0)), dimension(:), intent(in) :: par
00670         f211 = x
```

#### 3.2.2.54 pure real(kind(0d0)) function kernel_bernoulli_2::f212 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 709 of file RKHS.f90.

```
00709         implicit none
00710         real(kind(0d0)), intent(in) :: x
00711         real(kind(0d0)), dimension(:), intent(in) :: par
00712         f212 = x**2
```

#### 3.2.2.55 pure real(kind(0d0)) function kernel_bernoulli_2::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 289 of file RKHS.f90.

```
00289         implicit none
00290         real(kind(0d0)), intent(in) :: x
00291         real(kind(0d0)), dimension(:), intent(in) :: par
00292         f22 = x**2 - 1d0/30d0
```

#### 3.2.2.56 pure real(kind(0d0)) function kernel_bernoulli_2::f23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 331 of file RKHS.f90.

```
00331         implicit none
00332         real(kind(0d0)), intent(in) :: x
00333         real(kind(0d0)), dimension(:), intent(in) :: par
00334         f23 = 1d0
```

#### 3.2.2.57 pure real(kind(0d0)) function kernel_bernoulli_2::f24 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 373 of file RKHS.f90.

```
00373         implicit none
00374         real(kind(0d0)), intent(in) :: x
00375         real(kind(0d0)), dimension(:), intent(in) :: par
00376         f24 = x**3
```

#### 3.2.2.58 pure real(kind(0d0)) function kernel_bernoulli_2::f25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 415 of file RKHS.f90.

```
00415         implicit none
00416         real(kind(0d0)), intent(in) :: x
00417         real(kind(0d0)), dimension(:), intent(in) :: par
00418         f25 = 1d0
```

#### 3.2.2.59 pure real(kind(0d0)) function kernel_bernoulli_2::f26 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 457 of file RKHS.f90.

```
00457         implicit none
00458         real(kind(0d0)), intent(in) :: x
00459         real(kind(0d0)), dimension(:), intent(in) :: par
00460         f26 = x
```

#### 3.2.2.60 pure real(kind(0d0)) function kernel_bernoulli_2::f27 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 499 of file RKHS.f90.

```
00499         implicit none
00500         real(kind(0d0)), intent(in) :: x
00501         real(kind(0d0)), dimension(:), intent(in) :: par
00502         f27 = x**2
```

#### 3.2.2.61 pure real(kind(0d0)) function kernel_bernoulli_2::f28 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 541 of file RKHS.f90.

```
00541         implicit none
00542         real(kind(0d0)), intent(in) :: x
00543         real(kind(0d0)), dimension(:), intent(in) :: par
00544         f28 = x**4
```

#### 3.2.2.62 pure real(kind(0d0)) function kernel_bernoulli_2::f29 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 583 of file RKHS.f90.

```
00583         implicit none
00584         real(kind(0d0)), intent(in) :: x
00585         real(kind(0d0)), dimension(:), intent(in) :: par
00586         f29 = 1d0
```

#### 3.2.2.63 pure real(kind(0d0)) function kernel_bernoulli_2::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 268 of file RKHS.f90.

```
00268         implicit none
00269         real(kind(0d0)), intent(in) :: x
00270         real(kind(0d0)), dimension(:), intent(in) :: par
00271         f31 = x
```

#### 3.2.2.64 pure real(kind(0d0)) function kernel_bernoulli_2::f310 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 646 of file RKHS.f90.

```
00646         implicit none
00647         real(kind(0d0)), intent(in) :: x
00648         real(kind(0d0)), dimension(:), intent(in) :: par
00649         f310 = x
```

#### 3.2.2.65 pure real(kind(0d0)) function kernel_bernoulli_2::f311 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 688 of file RKHS.f90.

```
00688         implicit none
00689         real(kind(0d0)), intent(in) :: x
00690         real(kind(0d0)), dimension(:), intent(in) :: par
00691         f311 = x**3
```

#### 3.2.2.66 pure real(kind(0d0)) function kernel_bernoulli_2::f312 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 730 of file RKHS.f90.

```
00730         implicit none
00731         real(kind(0d0)), intent(in) :: x
00732         real(kind(0d0)), dimension(:), intent(in) :: par
00733         f312 = x**2
```

#### 3.2.2.67 pure real(kind(0d0)) function kernel_bernoulli_2::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 310 of file RKHS.f90.

```
00310         implicit none
00311         real(kind(0d0)), intent(in) :: x
00312         real(kind(0d0)), dimension(:), intent(in) :: par
00313         f32 = 1d0
```

#### 3.2.2.68 pure real(kind(0d0)) function kernel_bernoulli_2::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 352 of file RKHS.f90.

```
00352         implicit none
00353         real(kind(0d0)), intent(in) :: x
00354         real(kind(0d0)), dimension(:), intent(in) :: par
00355         f33 = x**2
```

#### 3.2.2.69 pure real(kind(0d0)) function kernel_bernoulli_2::f34 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 394 of file RKHS.f90.

```
00394         implicit none
00395         real(kind(0d0)), intent(in) :: x
00396         real(kind(0d0)), dimension(:), intent(in) :: par
00397         f34 = 1d0
```

#### 3.2.2.70 pure real(kind(0d0)) function kernel_bernoulli_2::f35 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 436 of file RKHS.f90.

```
00436         implicit none
00437         real(kind(0d0)), intent(in) :: x
00438         real(kind(0d0)), dimension(:), intent(in) :: par
00439         f35 = x**3
```

#### 3.2.2.71 pure real(kind(0d0)) function kernel_bernoulli_2::f36 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 478 of file RKHS.f90.

```
00478         implicit none
00479         real(kind(0d0)), intent(in) :: x
00480         real(kind(0d0)), dimension(:), intent(in) :: par
00481         f36 = x**2
```

#### 3.2.2.72 pure real(kind(0d0)) function kernel_bernoulli_2::f37 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 520 of file RKHS.f90.

```
00520         implicit none
00521         real(kind(0d0)), intent(in) :: x
00522         real(kind(0d0)), dimension(:), intent(in) :: par
00523         f37 = x
```

#### 3.2.2.73 pure real(kind(0d0)) function kernel_bernoulli_2::f38 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 562 of file RKHS.f90.

```
00562         implicit none
00563         real(kind(0d0)), intent(in) :: x
00564         real(kind(0d0)), dimension(:), intent(in) :: par
00565         f38 = 1d0
```

#### 3.2.2.74 pure real(kind(0d0)) function kernel_bernoulli_2::f39 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 604 of file RKHS.f90.

```
00604         implicit none
00605         real(kind(0d0)), intent(in) :: x
00606         real(kind(0d0)), dimension(:), intent(in) :: par
00607         f39 = x**4
```

#### 3.2.2.75 pure real(kind(0d0)) function kernel_bernoulli_2::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 196 of file RKHS.f90.

```
00196         implicit none
00197         real(kind(0d0)), intent(in) :: x1,x2
00198         real(kind(0d0))             :: xs,xl
00199         real(kind(0d0)), dimension(:), intent(in) :: par
00200         !find larger/smaller of x1 and x2
00201         if(x1 <= x2) then
00202             xs = x1
00203             xl = x2
00204         else
```

```
00205             xs = x2
00206             xl = xl
00207         end if
00208         k = 0.1d1 / 0.720d3 + xs * xl / 0.12d2 - xs ** 2 / 0.24d2 - xl ** 2 &
00209             / 0.24d2 - xs ** 3 / 0.12d2 + xl ** 3 / 0.12d2 - xs * xl ** 2 / &
00210             0.4d1 + xs ** 2 * xl / 0.4d1 - xs ** 4 / 0.24d2 - xl ** 4 / 0.24d2 &
00211             + xs ** 3 * xl / 0.6d1 + xs * xl ** 3 / 0.6d1 - xs ** 2 * xl ** 2 / 0.4d1
```

### 3.2.3 Variable Documentation

#### 3.2.3.1 integer, parameter kernel_bernoulli_2::m2 = 12

Definition at line 178 of file RKHS.f90.

```
00178     integer, parameter :: m2   = 12
```

#### 3.2.3.2 integer, parameter kernel_bernoulli_2::npar = 0

Definition at line 179 of file RKHS.f90.

```
00179     integer, parameter :: npar = 0
```

#### 3.2.3.3 real(kind(0d0)), parameter kernel_bernoulli_2::p21 = 1d0/12d0

Definition at line 180 of file RKHS.f90.

```
00180     real(kind(0d0)), parameter :: p21  =  1d0/12d0,&
00181                                   p22  = -1d0/24d0,&
00182                                   p23  = -1d0/24d0,&
00183                                   p24  = -1d0/12d0,&
00184                                   p25  =  1d0/12d0,&
00185                                   p26  = -1d0/4d0 ,&
00186                                   p27  =  1d0/4d0 ,&
00187                                   p28  = -1d0/24d0,&
00188                                   p29  = -1d0/24d0,&
00189                                   p210 =  1d0/6d0,&
00190                                   p211 =  1d0/6d0,&
00191                                   p212 = -1d0/4d0
```

#### 3.2.3.4 real(kind(0d0)), parameter kernel_bernoulli_2::p210 = 1d0/6d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.5 real(kind(0d0)), parameter kernel_bernoulli_2::p211 = 1d0/6d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.6 real(kind(0d0)), parameter kernel_bernoulli_2::p212 = -1d0/4d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.7 real(kind(0d0)), parameter kernel_bernoulli_2::p22 = -1d0/24d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.8 real(kind(0d0)), parameter kernel_bernoulli_2::p23 = -1d0/24d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.9 real(kind(0d0)), parameter kernel_bernoulli_2::p24 = -1d0/12d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.10 real(kind(0d0)), parameter kernel_bernoulli_2::p25 = 1d0/12d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.11 real(kind(0d0)), parameter kernel_bernoulli_2::p26 = -1d0/4d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.12 real(kind(0d0)), parameter kernel_bernoulli_2::p27 = 1d0/4d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.13 real(kind(0d0)), parameter kernel_bernoulli_2::p28 = -1d0/24d0

Definition at line 180 of file RKHS.f90.

#### 3.2.3.14 real(kind(0d0)), parameter kernel_bernoulli_2::p29 = -1d0/24d0

Definition at line 180 of file RKHS.f90.

## 3.3 kernel_exp_2 Module Reference

Exponential decay kernel with n = 2.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 1
- real(kind(0d0)), parameter p21 = 4d0
- real(kind(0d0)), parameter p22 = 4d0

### 3.3.1 Detailed Description

Exponential decay kernel with n = 2.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the exponential decay kernel with n = 2. For values larger than the greatest point in the grid, the kernel decays exponentially with exp(-beta∗x). Note that beta can be set to any positive value, but is initialized automatically with the value 1. This type of kernel is recommended for short-range intermolecular interactions, which often decay exponentially. The kernel is only defined for values in the interval [0,infinity).

### 3.3.2 Function/Subroutine Documentation

#### 3.3.2.1 pure real(kind(0d0)) function kernel_exp_2::d2f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 834 of file RKHS.f90.

```
00834         implicit none
00835         real(kind(0d0)), intent(in) :: x
00836         real(kind(0d0)), dimension(:), intent(in) :: par
00837         d2f21 = 0d0
```

#### 3.3.2.2 pure real(kind(0d0)) function kernel_exp_2::d2f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 855 of file RKHS.f90.

```
00855         implicit none
00856         real(kind(0d0)), intent(in) :: x
00857         real(kind(0d0)), dimension(:), intent(in) :: par
00858         d2f22 = 0d0
```

#### 3.3.2.3 pure real(kind(0d0)) function kernel_exp_2::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 876 of file RKHS.f90.

```
00876         implicit none
00877         real(kind(0d0)), intent(in) :: x
00878         real(kind(0d0)), dimension(:), intent(in) :: par
00879         d2f31 = exp(-par(1)*x)/par(1)
```

#### 3.3.2.4 pure real(kind(0d0)) function kernel_exp_2::d2f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 897 of file RKHS.f90.

```
00897         implicit none
00898         real(kind(0d0)), intent(in) :: x
00899         real(kind(0d0)), dimension(:), intent(in) :: par
00900         d2f32 = exp(-par(1)*x)/par(1) * (x - 2d0/par(1))
```

#### 3.3.2.5 pure real(kind(0d0)) function kernel_exp_2::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 807 of file RKHS.f90.

```
00807         implicit none
00808         real(kind(0d0)), intent(in) :: x1,x2
00809         real(kind(0d0)), dimension(:), intent(in) :: par
00810         !find larger/smaller of x1 and x2
00811         if(x1 <= x2) then
00812             d2k = 0d0
00813         else
00814             d2k = 4d0*exp(-par(1)*x1) * (x1 - x2)
00815         end if
```

#### 3.3.2.6 pure real(kind(0d0)) function kernel_exp_2::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 827 of file RKHS.f90.

```
00827         implicit none
00828         real(kind(0d0)), intent(in) :: x
00829         real(kind(0d0)), dimension(:), intent(in) :: par
00830         df21 = -par(1)
```

#### 3.3.2.7 pure real(kind(0d0)) function kernel_exp_2::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 848 of file RKHS.f90.

```
00848         implicit none
00849         real(kind(0d0)), intent(in) :: x
00850         real(kind(0d0)), dimension(:), intent(in) :: par
00851         df22 = 0d0
```

#### 3.3.2.8 pure real(kind(0d0)) function kernel_exp_2::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 869 of file RKHS.f90.

```
00869         implicit none
00870         real(kind(0d0)), intent(in) :: x
00871         real(kind(0d0)), dimension(:), intent(in) :: par
00872         df31 = -exp(-par(1)*x)/par(1)**2
```

#### 3.3.2.9 pure real(kind(0d0)) function kernel_exp_2::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 890 of file RKHS.f90.

```
00890         implicit none
00891         real(kind(0d0)), intent(in) :: x
00892         real(kind(0d0)), dimension(:), intent(in) :: par
00893         df32 = exp(-par(1)*x)/par(1)**2 * (1d0/par(1) - x)
```

#### 3.3.2.10 pure real(kind(0d0)) function kernel_exp_2::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 794 of file RKHS.f90.

```
00794         implicit none
00795         real(kind(0d0)), intent(in) :: x1,x2
00796         real(kind(0d0)), dimension(:), intent(in) :: par
00797         !find larger/smaller of x1 and x2
00798         if(x1 <= x2) then
00799             dk = -4d0*exp(-par(1)*x2)/par(1)**2
00800         else
00801             dk = -4d0*exp(-par(1)*x1)/par(1)**2 * &
00802                   (par(1)*(x1-x2) + 1d0)
00803         end if
```

#### 3.3.2.11 pure real(kind(0d0)) function kernel_exp_2::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 820 of file RKHS.f90.

```
00820         implicit none
00821         real(kind(0d0)), intent(in) :: x
00822         real(kind(0d0)), dimension(:), intent(in) :: par
00823         f21 = 2d0-par(1)*x
```

#### 3.3.2.12 pure real(kind(0d0)) function kernel_exp_2::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 841 of file RKHS.f90.

```
00841         implicit none
00842         real(kind(0d0)), intent(in) :: x
00843         real(kind(0d0)), dimension(:), intent(in) :: par
00844         f22 = par(1)
```

#### 3.3.2.13 pure real(kind(0d0)) function kernel_exp_2::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 862 of file RKHS.f90.

```
00862         implicit none
00863         real(kind(0d0)), intent(in) :: x
00864         real(kind(0d0)), dimension(:), intent(in) :: par
00865         f31 = exp(-par(1)*x)/par(1)**3
```

#### 3.3.2.14 pure real(kind(0d0)) function kernel_exp_2::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 883 of file RKHS.f90.

```
00883         implicit none
00884         real(kind(0d0)), intent(in) :: x
00885         real(kind(0d0)), dimension(:), intent(in) :: par
00886         f32 = x*exp(-par(1)*x)/par(1)**3
```

#### 3.3.2.15 pure real(kind(0d0)) function kernel_exp_2::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 777 of file RKHS.f90.

```
00777         implicit none
00778         real(kind(0d0)), intent(in) :: x1,x2
00779         real(kind(0d0))             :: xs,xl
00780         real(kind(0d0)), dimension(:), intent(in) :: par
00781         !find larger/smaller of x1 and x2
00782         if(x1 <= x2) then
00783             xs = x1
00784             xl = x2
00785         else
00786             xs = x2
00787             xl = x1
00788         end if
00789         k = 4d0*exp(-par(1)*xl)/par(1)**3 * &
00790             (par(1)*(xl - xs) + 2d0)
```

### 3.3.3 Variable Documentation

#### 3.3.3.1 integer, parameter kernel_exp_2::m2 = 2

Definition at line 770 of file RKHS.f90.

```
00770     integer, parameter :: m2   = 2
```

#### 3.3.3.2 integer, parameter kernel_exp_2::npar = 1

Definition at line 771 of file RKHS.f90.

```
00771     integer, parameter :: npar = 1
```

#### 3.3.3.3 real(kind(0d0)), parameter kernel_exp_2::p21 = 4d0

Definition at line 772 of file RKHS.f90.

```
00772     real(kind(0d0)), parameter :: p21 = 4d0,&
00773                                   p22 = 4d0
```

#### 3.3.3.4 real(kind(0d0)), parameter kernel_exp_2::p22 = 4d0

Definition at line 772 of file RKHS.f90.

## 3.4 kernel_exp_3 Module Reference

Exponential decay kernel with n = 3.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f24 (x, par)
- pure real(kind(0d0)) function df24 (x, par)
- pure real(kind(0d0)) function d2f24 (x, par)
- pure real(kind(0d0)) function f25 (x, par)
- pure real(kind(0d0)) function df25 (x, par)
- pure real(kind(0d0)) function d2f25 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)
- pure real(kind(0d0)) function f34 (x, par)
- pure real(kind(0d0)) function df34 (x, par)
- pure real(kind(0d0)) function d2f34 (x, par)
- pure real(kind(0d0)) function f35 (x, par)
- pure real(kind(0d0)) function df35 (x, par)
- pure real(kind(0d0)) function d2f35 (x, par)

**Variables**

- integer, parameter m2 = 5
- integer, parameter npar = 1
- real(kind(0d0)), parameter p21 = 18d0
- real(kind(0d0)), parameter p22 = 18d0
- real(kind(0d0)), parameter p23 = 18d0
- real(kind(0d0)), parameter p24 = 18d0
- real(kind(0d0)), parameter p25 = 18d0

### 3.4.1 Detailed Description

Exponential decay kernel with n = 3.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the exponential decay kernel with n = 3. For values larger than the greatest point in the grid, the kernel decays exponentially with exp(-beta∗x). Note that beta can be set to any positive value, but is initialized automatically with the value 1. This type of kernel is recommended for short-range intermolecular interactions, which often decay exponentially. The kernel is only defined for values in the interval [0,infinity).

### 3.4.2 Function/Subroutine Documentation

#### 3.4.2.1 pure real(kind(0d0)) function kernel_exp_3::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 991 of file RKHS.f90.

```
00991         implicit none
00992         real(kind(0d0)), intent(in) :: x
00993         real(kind(0d0)), dimension(:), intent(in) :: par
00994         d2f21 = 0d0
```

#### 3.4.2.2 pure real(kind(0d0)) function kernel_exp_3::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1012 of file RKHS.f90.

```
01012         implicit none
01013         real(kind(0d0)), intent(in) :: x
01014         real(kind(0d0)), dimension(:), intent(in) :: par
01015         d2f22 = 0d0
```

#### 3.4.2.3 pure real(kind(0d0)) function kernel_exp_3::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1033 of file RKHS.f90.

```
01033         implicit none
01034         real(kind(0d0)), intent(in) :: x
01035         real(kind(0d0)), dimension(:), intent(in) :: par
01036         d2f23 = 2d0*par(1)**2
```

#### 3.4.2.4 pure real(kind(0d0)) function kernel_exp_3::d2f24 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1054 of file RKHS.f90.

```
01054         implicit none
01055         real(kind(0d0)), intent(in) :: x
01056         real(kind(0d0)), dimension(:), intent(in) :: par
01057         d2f24 = 0d0
```

#### 3.4.2.5 pure real(kind(0d0)) function kernel_exp_3::d2f25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1075 of file RKHS.f90.

```
01075         implicit none
01076         real(kind(0d0)), intent(in) :: x
01077         real(kind(0d0)), dimension(:), intent(in) :: par
01078         d2f25 = 0d0
```

#### 3.4.2.6 pure real(kind(0d0)) function kernel_exp_3::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1096 of file RKHS.f90.

```
01096         implicit none
01097         real(kind(0d0)), intent(in) :: x
01098         real(kind(0d0)), dimension(:), intent(in) :: par
01099         d2f31 = exp(-par(1)*x)/par(1)**3
```

#### 3.4.2.7 pure real(kind(0d0)) function kernel_exp_3::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1117 of file RKHS.f90.

```
01117         implicit none
01118         real(kind(0d0)), intent(in) :: x
01119         real(kind(0d0)), dimension(:), intent(in) :: par
01120         d2f32 = exp(-par(1)*x)/par(1)**3 * (x - 2d0/par(1))
```

#### 3.4.2.8 pure real(kind(0d0)) function kernel_exp_3::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1138 of file RKHS.f90.

```
01138         implicit none
01139         real(kind(0d0)), intent(in) :: x
01140         real(kind(0d0)), dimension(:), intent(in) :: par
01141         d2f33 = exp(-par(1)*x)/par(1)**3
```

#### 3.4.2.9 pure real(kind(0d0)) function kernel_exp_3::d2f34 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1159 of file RKHS.f90.

```
01159         implicit none
01160         real(kind(0d0)), intent(in) :: x
01161         real(kind(0d0)), dimension(:), intent(in) :: par
01162         d2f34 = exp(-par(1)*x)/par(1)**3 * (x - 2d0/par(1))
```

#### 3.4.2.10 pure real(kind(0d0)) function kernel_exp_3::d2f35 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1180 of file RKHS.f90.

```
01180         implicit none
01181         real(kind(0d0)), intent(in) :: x
01182         real(kind(0d0)), dimension(:), intent(in) :: par
01183         d2f35 = exp(-par(1)*x)/par(1)**3 * (x**2 - 4d0*x/par(1) + 2d0/par(1)**2)
```

#### 3.4.2.11 pure real(kind(0d0)) function kernel_exp_3::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 964 of file RKHS.f90.

```
00964         implicit none
00965         real(kind(0d0)), intent(in) :: x1,x2
00966         real(kind(0d0)), dimension(:), intent(in) :: par
00967         !find larger/smaller of x1 and x2
00968         if(x1 <= x2) then
00969             d2k = 36d0*exp(-par(1)*x2)/par(1)**3
00970         else
00971             d2k = 18d0*exp(-par(1)*x1)/par(1)**3 * ((par(1)*(x1-x2))**2 + 2d0*par(1)*(x1-x2) + 2d0)
00972         end if
```

#### 3.4.2.12 pure real(kind(0d0)) function kernel_exp_3::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 984 of file RKHS.f90.

```
00984         implicit none
00985         real(kind(0d0)), intent(in) :: x
00986         real(kind(0d0)), dimension(:), intent(in) :: par
00987         df21 = -6d0*par(1)
```

#### 3.4.2.13 pure real(kind(0d0)) function kernel_exp_3::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1005 of file RKHS.f90.

```
01005         implicit none
01006         real(kind(0d0)), intent(in) :: x
01007         real(kind(0d0)), dimension(:), intent(in) :: par
01008         df22 = 0d0
```

#### 3.4.2.14 pure real(kind(0d0)) function kernel_exp_3::df23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1026 of file RKHS.f90.

```
01026         implicit none
01027         real(kind(0d0)), intent(in) :: x
01028         real(kind(0d0)), dimension(:), intent(in) :: par
01029         df23 = 2d0*par(1)**2*x
```

#### 3.4.2.15 pure real(kind(0d0)) function kernel_exp_3::df24 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1047 of file RKHS.f90.

```
01047         implicit none
01048         real(kind(0d0)), intent(in) :: x
01049         real(kind(0d0)), dimension(:), intent(in) :: par
01050         df24 = -2d0*par(1)**2
```

#### 3.4.2.16 pure real(kind(0d0)) function kernel_exp_3::df25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1068 of file RKHS.f90.

```
01068         implicit none
01069         real(kind(0d0)), intent(in) :: x
01070         real(kind(0d0)), dimension(:), intent(in) :: par
01071         df25 = 0d0
```

#### 3.4.2.17 pure real(kind(0d0)) function kernel_exp_3::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1089 of file RKHS.f90.

```
01089         implicit none
01090         real(kind(0d0)), intent(in) :: x
01091         real(kind(0d0)), dimension(:), intent(in) :: par
01092         df31 = -exp(-par(1)*x)/par(1)**4
```

#### 3.4.2.18 pure real(kind(0d0)) function kernel_exp_3::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1110 of file RKHS.f90.

```
01110         implicit none
01111         real(kind(0d0)), intent(in) :: x
01112         real(kind(0d0)), dimension(:), intent(in) :: par
01113         df32 = exp(-par(1)*x)/par(1)**4 * (1d0/par(1) - x)
```

#### 3.4.2.19 pure real(kind(0d0)) function kernel_exp_3::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1131 of file RKHS.f90.

```
01131         implicit none
01132         real(kind(0d0)), intent(in) :: x
01133         real(kind(0d0)), dimension(:), intent(in) :: par
01134         df33 = -exp(-par(1)*x)/par(1)**4
```

#### 3.4.2.20 pure real(kind(0d0)) function kernel_exp_3::df34 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1152 of file RKHS.f90.

```
01152         implicit none
01153         real(kind(0d0)), intent(in) :: x
01154         real(kind(0d0)), dimension(:), intent(in) :: par
01155         df34 = exp(-par(1)*x)/par(1)**4 * (1d0/par(1) - x)
```

#### 3.4.2.21 pure real(kind(0d0)) function kernel_exp_3::df35 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1173 of file RKHS.f90.

```
01173         implicit none
01174         real(kind(0d0)), intent(in) :: x
01175         real(kind(0d0)), dimension(:), intent(in) :: par
01176         df35 = exp(-par(1)*x)/par(1)**4 * (2d0*x/par(1) - x**2)
```

#### 3.4.2.22 pure real(kind(0d0)) function kernel_exp_3::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 951 of file RKHS.f90.

```
00951         implicit none
00952         real(kind(0d0)), intent(in) :: x1,x2
00953         real(kind(0d0)), dimension(:), intent(in) :: par
00954         !find larger/smaller of x1 and x2
00955         if(x1 <= x2) then
00956             dk = -36d0*exp(-par(1)*x2)/par(1)**4 * (par(1)*(x2-x1)+3d0)
00957         else
00958             dk = -18d0*exp(-par(1)*x1)/par(1)**4 * (par(1)**2 * (x1**2 - 2d0*x1*x2 + x2**2) &
00959                                                     + 4d0*par(1)*(x1 - x2) + 6d0)
00960         end if
```

#### 3.4.2.23 pure real(kind(0d0)) function kernel_exp_3::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 977 of file RKHS.f90.

```
00977         implicit none
00978         real(kind(0d0)), intent(in) :: x
00979         real(kind(0d0)), dimension(:), intent(in) :: par
00980         f21 = 12d0 - 6d0*par(1)*x
```

#### 3.4.2.24 pure real(kind(0d0)) function kernel_exp_3::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 998 of file RKHS.f90.

```
00998         implicit none
00999         real(kind(0d0)), intent(in) :: x
01000         real(kind(0d0)), dimension(:), intent(in) :: par
01001         f22 = 6d0*par(1)
```

#### 3.4.2.25 pure real(kind(0d0)) function kernel_exp_3::f23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1019 of file RKHS.f90.

```
01019         implicit none
01020         real(kind(0d0)), intent(in) :: x
01021         real(kind(0d0)), dimension(:), intent(in) :: par
01022         f23 = (par(1)*x)**2
```

#### 3.4.2.26 pure real(kind(0d0)) function kernel_exp_3::f24 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1040 of file RKHS.f90.

```
01040         implicit none
01041         real(kind(0d0)), intent(in) :: x
01042         real(kind(0d0)), dimension(:), intent(in) :: par
01043         f24 = -2d0*par(1)**2*x
```

#### 3.4.2.27 pure real(kind(0d0)) function kernel_exp_3::f25 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1061 of file RKHS.f90.

```
01061         implicit none
01062         real(kind(0d0)), intent(in) :: x
01063         real(kind(0d0)), dimension(:), intent(in) :: par
01064         f25 = par(1)**2
```

#### 3.4.2.28 pure real(kind(0d0)) function kernel_exp_3::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1082 of file RKHS.f90.

```
01082         implicit none
01083         real(kind(0d0)), intent(in) :: x
01084         real(kind(0d0)), dimension(:), intent(in) :: par
01085         f31 = exp(-par(1)*x)/par(1)**5
```

#### 3.4.2.29 pure real(kind(0d0)) function kernel_exp_3::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1103 of file RKHS.f90.

```
01103         implicit none
01104         real(kind(0d0)), intent(in) :: x
01105         real(kind(0d0)), dimension(:), intent(in) :: par
01106         f32 = x*exp(-par(1)*x)/par(1)**5
```

#### 3.4.2.30 pure real(kind(0d0)) function kernel_exp_3::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1124 of file RKHS.f90.

```
01124         implicit none
01125         real(kind(0d0)), intent(in) :: x
01126         real(kind(0d0)), dimension(:), intent(in) :: par
01127         f33 = exp(-par(1)*x)/par(1)**5
```

#### 3.4.2.31 pure real(kind(0d0)) function kernel_exp_3::f34 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1145 of file RKHS.f90.

```
01145         implicit none
01146         real(kind(0d0)), intent(in) :: x
01147         real(kind(0d0)), dimension(:), intent(in) :: par
01148         f34 = x*exp(-par(1)*x)/par(1)**5
```

#### 3.4.2.32 pure real(kind(0d0)) function kernel_exp_3::f35 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1166 of file RKHS.f90.

```
01166         implicit none
01167         real(kind(0d0)), intent(in) :: x
01168         real(kind(0d0)), dimension(:), intent(in) :: par
01169         f35 = x**2*exp(-par(1)*x)/par(1)**5
```

#### 3.4.2.33 pure real(kind(0d0)) function kernel_exp_3::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 933 of file RKHS.f90.

```
00933         implicit none
00934         real(kind(0d0)), intent(in) :: x1,x2
00935         real(kind(0d0))             :: xs,xl
00936         real(kind(0d0)), dimension(:), intent(in) :: par
00937         !find larger/smaller of x1 and x2
00938         if(x1 <= x2) then
00939             xs = x1
00940             xl = x2
00941         else
00942             xs = x2
00943             xl = x1
00944         end if
00945         k = 18d0*exp(-par(1)*xl)/par(1)**5 * (par(1)**2*(xl**2 - 2d0*xl*xs + xs**2) &
00946                                             + 6d0*par(1)*(xl - xs) + 12d0)
```

#### 3.4.3 Variable Documentation

##### 3.4.3.1 integer, parameter kernel_exp_3::m2 = 5

Definition at line 923 of file RKHS.f90.

```
00923     integer, parameter :: m2   = 5
```

##### 3.4.3.2 integer, parameter kernel_exp_3::npar = 1

Definition at line 924 of file RKHS.f90.

```
00924     integer, parameter :: npar = 1
```

##### 3.4.3.3 real(kind(0d0)), parameter kernel_exp_3::p21 = 18d0

Definition at line 925 of file RKHS.f90.

```
00925     real(kind(0d0)), parameter :: p21 = 18d0,&
00926                                   p22 = 18d0,&
00927                                   p23 = 18d0,&
00928                                   p24 = 18d0,&
00929                                   p25 = 18d0
```

##### 3.4.3.4 real(kind(0d0)), parameter kernel_exp_3::p22 = 18d0

Definition at line 925 of file RKHS.f90.

##### 3.4.3.5 real(kind(0d0)), parameter kernel_exp_3::p23 = 18d0

Definition at line 925 of file RKHS.f90.

##### 3.4.3.6 real(kind(0d0)), parameter kernel_exp_3::p24 = 18d0

Definition at line 925 of file RKHS.f90.

##### 3.4.3.7 real(kind(0d0)), parameter kernel_exp_3::p25 = 18d0

Definition at line 925 of file RKHS.f90.

### 3.5 kernel_laplacian Module Reference

Laplacian kernel.

#### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)

#### Variables

- integer, parameter m2 = 1
- integer, parameter npar = 1
- real(kind(0d0)), parameter p21 = 1d0

### 3.5.1 Detailed Description

Laplacian kernel.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the Laplacian kernel given by exp(||x-x'||/sigma). Note that sigma can be set to any positive value, but is initialized automatically with the value 1. It is not recommended for interpolating potential energy surfaces, but it can be useful for machine learning purposes. Note however, that in order for the decomposition into f2k and f3k functions to work, it is necessary that all input variables are positive. Therefore, the kernel is only defined in the interval [0,infinity), whereas normally a Laplacian kernel is valid in the interval (-infinity,infinity).

### 3.5.2 Function/Subroutine Documentation

#### 3.5.2.1 pure real(kind(0d0)) function kernel_laplacian::d2f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 134 of file RKHS.f90.

```
00134         implicit none
00135         real(kind(0d0)), intent(in) :: x
00136         real(kind(0d0)), dimension(:), intent(in) :: par
00137         d2f21 = exp(x/par(1))/par(1)**2
```

#### 3.5.2.2 pure real(kind(0d0)) function kernel_laplacian::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 155 of file RKHS.f90.

```
00155         implicit none
00156         real(kind(0d0)), intent(in) :: x
00157         real(kind(0d0)), dimension(:), intent(in) :: par
00158         d2f31 = exp(-x/par(1))/par(1)**2
```

#### 3.5.2.3 pure real(kind(0d0)) function kernel_laplacian::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 107 of file RKHS.f90.

```
00107         implicit none
00108         real(kind(0d0)), intent(in) :: x1,x2
00109         real(kind(0d0)), dimension(:), intent(in) :: par
00110         !find larger/smaller of x1 and x2
00111         if(x1 <= x2) then
00112             d2k = exp(-(x2-x1)/par(1))/par(1)**2
00113         else
00114             d2k = exp(-(x1-x2)/par(1))/par(1)**2
00115         end if
```

#### 3.5.2.4 pure real(kind(0d0)) function kernel_laplacian::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 127 of file RKHS.f90.

```
00127         implicit none
00128         real(kind(0d0)), intent(in) :: x
00129         real(kind(0d0)), dimension(:), intent(in) :: par
00130         df21 = exp(x/par(1))/par(1)
```

#### 3.5.2.5 pure real(kind(0d0)) function kernel_laplacian::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 148 of file RKHS.f90.

```
00148         implicit none
00149         real(kind(0d0)), intent(in) :: x
00150         real(kind(0d0)), dimension(:), intent(in) :: par
00151         df31 = -exp(-x/par(1))/par(1)
```

#### 3.5.2.6 pure real(kind(0d0)) function kernel_laplacian::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 95 of file RKHS.f90.

```
00095         implicit none
00096         real(kind(0d0)), intent(in) :: x1,x2
00097         real(kind(0d0)), dimension(:), intent(in) :: par
00098         !find larger/smaller of x1 and x2
00099         if(x1 <= x2) then
00100             dk = exp(-(x2-x1)/par(1))/par(1)
00101         else
00102             dk = -exp(-(x1-x2)/par(1))/par(1)
00103         end if
```

#### 3.5.2.7 pure real(kind(0d0)) function kernel_laplacian::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 120 of file RKHS.f90.

```
00120         implicit none
00121         real(kind(0d0)), intent(in) :: x
00122         real(kind(0d0)), dimension(:), intent(in) :: par
00123         f21 = exp(x/par(1))
```

#### 3.5.2.8 pure real(kind(0d0)) function kernel_laplacian::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 141 of file RKHS.f90.

```
00141         implicit none
00142         real(kind(0d0)), intent(in) :: x
00143         real(kind(0d0)), dimension(:), intent(in) :: par
00144         f31 = exp(-x/par(1))
```

#### 3.5.2.9 pure real(kind(0d0)) function kernel_laplacian::k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 79 of file RKHS.f90.

```
00079         implicit none
00080         real(kind(0d0)), intent(in) :: x1,x2
00081         real(kind(0d0))             :: xs,xl
00082         real(kind(0d0)), dimension(:), intent(in) :: par
```

```
00083         !find larger/smaller of x1 and x2
00084         if(x1 <= x2) then
00085             xs = x1
00086             xl = x2
00087         else
00088             xs = x2
00089             xl = x1
00090         end if
00091         k = exp(-(xl-xs)/par(1))
```

#### 3.5.3 Variable Documentation

##### 3.5.3.1 integer parameter kernel_ts_3::m2 = 1

Definition at line 72 of file RKHS.f90.

```
00072     integer, parameter :: m2   = 1
```

##### 3.5.3.2 integer parameter kernel_ts_3::npar = 1

Definition at line 73 of file RKHS.f90.

```
00073     integer, parameter :: npar = 1
```

##### 3.5.3.3 real(kind(0d0)), parameter kernel_laplacian::p21 = 1d0

Definition at line 74 of file RKHS.f90.

```
00074     real(kind(0d0)), parameter :: p21 = 1d0
```

### 3.6 kernel_rp_2_0 Module Reference

Reciprocal power decay kernel with n = 2 and m = 0.

#### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

#### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 2d0
- real(kind(0d0)), parameter p22 = -2d0/3d0

### 3.6.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 0.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 0. This corresponds to a 1/r decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-charge interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.6.2 Function/Subroutine Documentation

#### 3.6.2.1 pure real(kind(0d0)) function kernel_rp_2_0::d2f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1269 of file RKHS.f90.

```
01269         implicit none
01270         real(kind(0d0)), intent(in) :: x
01271         real(kind(0d0)), dimension(:), intent(in) :: par
01272         d2f21 = 0d0
```

#### 3.6.2.2 pure real(kind(0d0)) function kernel_rp_2_0::d2f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1290 of file RKHS.f90.

```
01290         implicit none
01291         real(kind(0d0)), intent(in) :: x
01292         real(kind(0d0)), dimension(:), intent(in) :: par
01293         d2f22 = 0d0
```

#### 3.6.2.3 pure real(kind(0d0)) function kernel_rp_2_0::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1311 of file RKHS.f90.

```
01311         implicit none
01312         real(kind(0d0)), intent(in) :: x
01313         real(kind(0d0)), dimension(:), intent(in) :: par
01314         d2f31 = 2d0/x**3
```

#### 3.6.2.4 pure real(kind(0d0)) function kernel_rp_2_0::d2f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1332 of file RKHS.f90.

```
01332         implicit none
01333         real(kind(0d0)), intent(in) :: x
01334         real(kind(0d0)), dimension(:), intent(in) :: par
01335         d2f32 = 6d0/x**4
```

#### 3.6.2.5 pure real(kind(0d0)) function kernel_rp_2_0::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1242 of file RKHS.f90.

```
01242         implicit none
01243         real(kind(0d0)), intent(in) :: x1,x2
01244         real(kind(0d0)), dimension(:), intent(in) :: par
01245         !find larger/smaller of x1 and x2
01246         if(x1 <= x2) then
01247             d2k = 0d0
01248         else
01249             d2k = 4d0/x1**3 - 4d0*x2/x1**4
01250         end if
```

#### 3.6.2.6 pure real(kind(0d0)) function kernel_rp_2_0::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1262 of file RKHS.f90.

```
01262         implicit none
01263         real(kind(0d0)), intent(in) :: x
01264         real(kind(0d0)), dimension(:), intent(in) :: par
01265         df21 = 0d0
```

#### 3.6.2.7 pure real(kind(0d0)) function kernel_rp_2_0::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1283 of file RKHS.f90.

```
01283         implicit none
01284         real(kind(0d0)), intent(in) :: x
01285         real(kind(0d0)), dimension(:), intent(in) :: par
01286         df22 = 1d0
```

#### 3.6.2.8 pure real(kind(0d0)) function kernel_rp_2_0::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1304 of file RKHS.f90.

```
01304         implicit none
01305         real(kind(0d0)), intent(in) :: x
01306         real(kind(0d0)), dimension(:), intent(in) :: par
01307         df31 = -1d0/x**2
```

#### 3.6.2.9 pure real(kind(0d0)) function kernel_rp_2_0::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1325 of file RKHS.f90.

```
01325         implicit none
01326         real(kind(0d0)), intent(in) :: x
01327         real(kind(0d0)), dimension(:), intent(in) :: par
01328         df32 = -2d0/x**3
```

#### 3.6.2.10 pure real(kind(0d0)) function kernel_rp_2_0::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1229 of file RKHS.f90.

```
01229         implicit none
01230         real(kind(0d0)), intent(in) :: x1,x2
01231         real(kind(0d0)), dimension(:), intent(in) :: par
01232         !find larger/smaller of x1 and x2
01233         if(x1 <= x2) then
01234             dk = -2d0/(3d0*x2**2)
01235         else
01236             dk = 4d0/3d0 * x2/x1**3 - 2d0/x1**2
01237         end if
```

#### 3.6.2.11 pure real(kind(0d0)) function kernel_rp_2_0::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1255 of file RKHS.f90.

```
01255         implicit none
01256         real(kind(0d0)), intent(in) :: x
01257         real(kind(0d0)), dimension(:), intent(in) :: par
01258         f21 = 1d0
```

#### 3.6.2.12 pure real(kind(0d0)) function kernel_rp_2_0::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1276 of file RKHS.f90.

```
01276         implicit none
01277         real(kind(0d0)), intent(in) :: x
01278         real(kind(0d0)), dimension(:), intent(in) :: par
01279         f22 = x
```

#### 3.6.2.13 pure real(kind(0d0)) function kernel_rp_2_0::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1297 of file RKHS.f90.

```
01297         implicit none
01298         real(kind(0d0)), intent(in) :: x
01299         real(kind(0d0)), dimension(:), intent(in) :: par
01300         f31 = 1d0/x
```

#### 3.6.2.14 pure real(kind(0d0)) function kernel_rp_2_0::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1318 of file RKHS.f90.

```
01318         implicit none
01319         real(kind(0d0)), intent(in) :: x
01320         real(kind(0d0)), dimension(:), intent(in) :: par
01321         f32 = 1d0/x**2
```

#### 3.6.2.15 pure real(kind(0d0)) function kernel_rp_2_0::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1212 of file RKHS.f90.

```
01212         implicit none
01213         real(kind(0d0)), intent(in) :: x1,x2
01214         real(kind(0d0))             :: xs,xl
01215         real(kind(0d0)), dimension(:), intent(in) :: par
01216         !find larger/smaller of x1 and x2
01217         if(x1 <= x2) then
01218             xs = x1
01219             xl = x2
01220         else
01221             xs = x2
01222             xl = x1
01223         end if
01224         k = 2d0/xl - 2d0/3d0 * xs/xl**2
```

### 3.6.3 Variable Documentation

#### 3.6.3.1 integer, parameter kernel_rp_2_0::m2 = 2

Definition at line 1205 of file RKHS.f90.

```
01205     integer, parameter :: m2   = 2
```

#### 3.6.3.2 integer, parameter kernel_rp_2_0::npar = 0

Definition at line 1206 of file RKHS.f90.

```
01206     integer, parameter :: npar = 0
```

#### 3.6.3.3 real(kind(0d0)), parameter kernel_rp_2_0::p21 = 2d0

Definition at line 1207 of file RKHS.f90.

```
01207     real(kind(0d0)), parameter :: p21 = 2d0  ,&
01208                                   p22 = -2d0/3d0
```

#### 3.6.3.4 real(kind(0d0)), parameter kernel_rp_2_0::p22 = -2d0/3d0

Definition at line 1207 of file RKHS.f90.

## 3.7 kernel_rp_2_1 Module Reference

Reciprocal power decay kernel with n = 2 and m = 1.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 2d0/3d0
- real(kind(0d0)), parameter p22 = -1d0/3d0

### 3.7.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 1.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 1. This corresponds to a 1/r^2 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.7.2 Function/Subroutine Documentation

#### 3.7.2.1 pure real(kind(0d0)) function kernel_rp_2_1::d2f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1616 of file RKHS.f90.

```
01616         implicit none
01617         real(kind(0d0)), intent(in) :: x
01618         real(kind(0d0)), dimension(:), intent(in) :: par
01619         d2f21 = 0d0
```

#### 3.7.2.2 pure real(kind(0d0)) function kernel_rp_2_1::d2f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1637 of file RKHS.f90.

```
01637         implicit none
01638         real(kind(0d0)), intent(in) :: x
01639         real(kind(0d0)), dimension(:), intent(in) :: par
01640         d2f22 = 0d0
```

#### 3.7.2.3 pure real(kind(0d0)) function kernel_rp_2_1::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1658 of file RKHS.f90.

```
01658         implicit none
01659         real(kind(0d0)), intent(in) :: x
01660         real(kind(0d0)), dimension(:), intent(in) :: par
01661         d2f31 = 6d0/x**4
```

#### 3.7.2.4 pure real(kind(0d0)) function kernel_rp_2_1::d2f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1679 of file RKHS.f90.

```
01679         implicit none
01680         real(kind(0d0)), intent(in) :: x
01681         real(kind(0d0)), dimension(:), intent(in) :: par
01682         d2f32 = 12d0/x**5
```

#### 3.7.2.5 pure real(kind(0d0)) function kernel_rp_2_1::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1589 of file RKHS.f90.

```
01589         implicit none
01590         real(kind(0d0)), intent(in) :: x1,x2
01591         real(kind(0d0)), dimension(:), intent(in) :: par
01592         !find larger/smaller of x1 and x2
01593         if(x1 <= x2) then
01594             d2k = 0d0
01595         else
01596             d2k = 4d0/x1**4 - 4d0*x2/x1**5
01597         end if
```

#### 3.7.2.6 pure real(kind(0d0)) function kernel_rp_2_1::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1609 of file RKHS.f90.

```
01609         implicit none
01610         real(kind(0d0)), intent(in) :: x
01611         real(kind(0d0)), dimension(:), intent(in) :: par
01612         df21 = 0d0
```

#### 3.7.2.7 pure real(kind(0d0)) function kernel_rp_2_1::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1630 of file RKHS.f90.

```
01630         implicit none
01631         real(kind(0d0)), intent(in) :: x
01632         real(kind(0d0)), dimension(:), intent(in) :: par
01633         df22 = 1d0
```

#### 3.7.2.8 pure real(kind(0d0)) function kernel_rp_2_1::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1651 of file RKHS.f90.

```
01651         implicit none
01652         real(kind(0d0)), intent(in) :: x
01653         real(kind(0d0)), dimension(:), intent(in) :: par
01654         df31 = -2d0/x**3
```

#### 3.7.2.9 pure real(kind(0d0)) function kernel_rp_2_1::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1672 of file RKHS.f90.

```
01672         implicit none
01673         real(kind(0d0)), intent(in) :: x
01674         real(kind(0d0)), dimension(:), intent(in) :: par
01675         df32 = -3d0/x**4
```

#### 3.7.2.10 pure real(kind(0d0)) function kernel_rp_2_1::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1576 of file RKHS.f90.

```
01576         implicit none
01577         real(kind(0d0)), intent(in) :: x1,x2
01578         real(kind(0d0)), dimension(:), intent(in) :: par
01579         !find larger/smaller of x1 and x2
01580         if(x1 <= x2) then
01581             dk = -1d0/(3d0*x2**3)
01582         else
01583             dk = x2/x1**4 - 4d0/(3d0*x1**3)
01584         end if
```

#### 3.7.2.11 pure real(kind(0d0)) function kernel_rp_2_1::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1602 of file RKHS.f90.

```
01602         implicit none
01603         real(kind(0d0)), intent(in) :: x
01604         real(kind(0d0)), dimension(:), intent(in) :: par
01605         f21 = 1d0
```

#### 3.7.2.12 pure real(kind(0d0)) function kernel_rp_2_1::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1623 of file RKHS.f90.

```
01623         implicit none
01624         real(kind(0d0)), intent(in) :: x
01625         real(kind(0d0)), dimension(:), intent(in) :: par
01626         f22 = x
```

#### 3.7.2.13 pure real(kind(0d0)) function kernel_rp_2_1::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1644 of file RKHS.f90.

```
01644         implicit none
01645         real(kind(0d0)), intent(in) :: x
01646         real(kind(0d0)), dimension(:), intent(in) :: par
01647         f31 = 1d0/x**2
```

#### 3.7.2.14 pure real(kind(0d0)) function kernel_rp_2_1::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1665 of file RKHS.f90.

```
01665         implicit none
01666         real(kind(0d0)), intent(in) :: x
01667         real(kind(0d0)), dimension(:), intent(in) :: par
01668         f32 = 1d0/x**3
```

#### 3.7.2.15 pure real(kind(0d0)) function kernel_rp_2_1::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1559 of file RKHS.f90.

```
01559         implicit none
01560         real(kind(0d0)), intent(in) :: x1,x2
01561         real(kind(0d0))             :: xs,xl
01562         real(kind(0d0)), dimension(:), intent(in) :: par
01563         !find larger/smaller of x1 and x2
01564         if(x1 <= x2) then
01565             xs = x1
01566             xl = x2
01567         else
01568             xs = x2
01569             xl = x1
01570         end if
01571         k = 2d0/(3d0*xl**2) - 1d0/3d0 * xs/xl**3
```

### 3.7.3 Variable Documentation

#### 3.7.3.1 integer, parameter kernel_rp_2_1::m2 = 2

Definition at line 1552 of file RKHS.f90.

```
01552     integer, parameter :: m2   = 2
```

#### 3.7.3.2 integer, parameter kernel_rp_2_1::npar = 0

Definition at line 1553 of file RKHS.f90.

```
01553     integer, parameter :: npar = 0
```

#### 3.7.3.3 real(kind(0d0)), parameter kernel_rp_2_1::p21 = 2d0/3d0

Definition at line 1554 of file RKHS.f90.

```
01554     real(kind(0d0)), parameter :: p21 = 2d0/3d0,&
01555                                   p22 = -1d0/3d0
```

#### 3.7.3.4 real(kind(0d0)), parameter kernel_rp_2_1::p22 = -1d0/3d0

Definition at line 1554 of file RKHS.f90.

## 3.8 kernel_rp_2_2 Module Reference

Reciprocal power decay kernel with n = 2 and m = 2.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 1d0/3d0
- real(kind(0d0)), parameter p22 = -1d0/5d0

### 3.8.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 2.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 2. This corresponds to a 1/r^3 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling dipole-dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.8.2 Function/Subroutine Documentation

#### 3.8.2.1 pure real(kind(0d0)) function kernel_rp_2_2::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1963 of file RKHS.f90.

```
01963         implicit none
01964         real(kind(0d0)), intent(in) :: x
01965         real(kind(0d0)), dimension(:), intent(in) :: par
01966         d2f21 = 0d0
```

#### 3.8.2.2 pure real(kind(0d0)) function kernel_rp_2_2::d2f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1984 of file RKHS.f90.

```
01984         implicit none
01985         real(kind(0d0)), intent(in) :: x
01986         real(kind(0d0)), dimension(:), intent(in) :: par
01987         d2f22 = 0d0
```

#### 3.8.2.3 pure real(kind(0d0)) function kernel_rp_2_2::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2005 of file RKHS.f90.

```
02005         implicit none
02006         real(kind(0d0)), intent(in) :: x
02007         real(kind(0d0)), dimension(:), intent(in) :: par
02008         d2f31 = 12d0/x**5
```

#### 3.8.2.4 pure real(kind(0d0)) function kernel_rp_2_2::d2f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2026 of file RKHS.f90.

```
02026         implicit none
02027         real(kind(0d0)), intent(in) :: x
02028         real(kind(0d0)), dimension(:), intent(in) :: par
02029         d2f32 = 20d0/x**6
```

#### 3.8.2.5 pure real(kind(0d0)) function kernel_rp_2_2::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1936 of file RKHS.f90.

```
01936         implicit none
01937         real(kind(0d0)), intent(in) :: x1,x2
01938         real(kind(0d0)), dimension(:), intent(in) :: par
01939         !find larger/smaller of x1 and x2
01940         if(x1 <= x2) then
01941             d2k = 0d0
01942         else
01943             d2k = 4d0/x1**5 - 4d0*x2/x1**6
01944         end if
```

#### 3.8.2.6 pure real(kind(0d0)) function kernel_rp_2_2::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1956 of file RKHS.f90.

```
01956         implicit none
01957         real(kind(0d0)), intent(in) :: x
01958         real(kind(0d0)), dimension(:), intent(in) :: par
01959         df21 = 0d0
```

#### 3.8.2.7 pure real(kind(0d0)) function kernel_rp_2_2::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1977 of file RKHS.f90.

```
01977         implicit none
01978         real(kind(0d0)), intent(in) :: x
01979         real(kind(0d0)), dimension(:), intent(in) :: par
01980         df22 = 1d0
```

#### 3.8.2.8 pure real(kind(0d0)) function kernel_rp_2_2::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1998 of file RKHS.f90.

```
01998         implicit none
01999         real(kind(0d0)), intent(in) :: x
02000         real(kind(0d0)), dimension(:), intent(in) :: par
02001         df31 = -3d0/x**4
```

#### 3.8.2.9 pure real(kind(0d0)) function kernel_rp_2_2::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2019 of file RKHS.f90.

```
02019         implicit none
02020         real(kind(0d0)), intent(in) :: x
02021         real(kind(0d0)), dimension(:), intent(in) :: par
02022         df32 = -4d0/x**5
```

#### 3.8.2.10 pure real(kind(0d0)) function kernel_rp_2_2::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1923 of file RKHS.f90.

```
01923         implicit none
01924         real(kind(0d0)), intent(in) :: x1,x2
01925         real(kind(0d0)), dimension(:), intent(in) :: par
01926         !find larger/smaller of x1 and x2
01927         if(x1 <= x2) then
01928             dk = -1d0/(5d0*x2**4)
01929         else
01930             dk = 4d0/5d0*(x2/x1**5) - 1d0/x1**4
01931         end if
```

#### 3.8.2.11 pure real(kind(0d0)) function kernel_rp_2_2::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1949 of file RKHS.f90.

```
01949         implicit none
01950         real(kind(0d0)), intent(in) :: x
01951         real(kind(0d0)), dimension(:), intent(in) :: par
01952         f21 = 1d0
```

#### 3.8.2.12 pure real(kind(0d0)) function kernel_rp_2_2::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1970 of file RKHS.f90.

```
01970         implicit none
01971         real(kind(0d0)), intent(in) :: x
01972         real(kind(0d0)), dimension(:), intent(in) :: par
01973         f22 = x
```

#### 3.8.2.13 pure real(kind(0d0)) function kernel_rp_2_2::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1991 of file RKHS.f90.

```
01991         implicit none
01992         real(kind(0d0)), intent(in) :: x
01993         real(kind(0d0)), dimension(:), intent(in) :: par
01994         f31 = 1d0/x**3
```

#### 3.8.2.14 pure real(kind(0d0)) function kernel_rp_2_2::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2012 of file RKHS.f90.

```
02012         implicit none
02013         real(kind(0d0)), intent(in) :: x
02014         real(kind(0d0)), dimension(:), intent(in) :: par
02015         f32 = 1d0/x**4
```

#### 3.8.2.15 pure real(kind(0d0)) function kernel_rp_2_2::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1906 of file RKHS.f90.

```
01906         implicit none
01907         real(kind(0d0)), intent(in) :: x1,x2
01908         real(kind(0d0))             :: xs,xl
01909         real(kind(0d0)), dimension(:), intent(in) :: par
01910         !find larger/smaller of x1 and x2
01911         if(x1 <= x2) then
01912             xs = x1
01913             xl = x2
01914         else
01915             xs = x2
01916             xl = x1
01917         end if
01918         k = 1d0/(3d0*xl**3) - 1d0/5d0 * xs/xl**4
```

### 3.8.3 Variable Documentation

#### 3.8.3.1 integer, parameter kernel_rp_2_2::m2 = 2

Definition at line 1899 of file RKHS.f90.

```
01899     integer, parameter :: m2   = 2
```

#### 3.8.3.2 integer, parameter kernel_rp_2_2::npar = 0

Definition at line 1900 of file RKHS.f90.

```
01900     integer, parameter :: npar = 0
```

#### 3.8.3.3 real(kind(0d0)), parameter kernel_rp_2_2::p21 = 1d0/3d0

Definition at line 1901 of file RKHS.f90.

```
01901     real(kind(0d0)), parameter :: p21 = 1d0/3d0,&
01902                                   p22 = -1d0/5d0
```

#### 3.8.3.4 real(kind(0d0)), parameter kernel_rp_2_2::p22 = -1d0/5d0

Definition at line 1901 of file RKHS.f90.

## 3.9 kernel_rp_2_3 Module Reference

Reciprocal power decay kernel with n = 2 and m = 3.

#### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

#### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 1d0/5d0
- real(kind(0d0)), parameter p22 = -2d0/15d0

### 3.9.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 3.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 3. This corresponds to a 1/r^4 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-induced dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.9.2 Function/Subroutine Documentation

#### 3.9.2.1 pure real(kind(0d0)) function kernel_rp_2_3::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2310 of file RKHS.f90.

```
02310         implicit none
02311         real(kind(0d0)), intent(in) :: x
02312         real(kind(0d0)), dimension(:), intent(in) :: par
02313         d2f21 = 0d0
```

#### 3.9.2.2 pure real(kind(0d0)) function kernel_rp_2_3::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2331 of file RKHS.f90.

```
02331         implicit none
02332         real(kind(0d0)), intent(in) :: x
02333         real(kind(0d0)), dimension(:), intent(in) :: par
02334         d2f22 = 0d0
```

#### 3.9.2.3 pure real(kind(0d0)) function kernel_rp_2_3::d2f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2352 of file RKHS.f90.

```
02352         implicit none
02353         real(kind(0d0)), intent(in) :: x
02354         real(kind(0d0)), dimension(:), intent(in) :: par
02355         d2f31 = 20d0/x**6
```

#### 3.9.2.4 pure real(kind(0d0)) function kernel_rp_2_3::d2f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2373 of file RKHS.f90.

```
02373         implicit none
02374         real(kind(0d0)), intent(in) :: x
02375         real(kind(0d0)), dimension(:), intent(in) :: par
02376         d2f32 = 30d0/x**7
```

#### 3.9.2.5 pure real(kind(0d0)) function kernel_rp_2_3::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2283 of file RKHS.f90.

```
02283         implicit none
02284         real(kind(0d0)), intent(in) :: x1,x2
02285         real(kind(0d0)), dimension(:), intent(in) :: par
02286         !find larger/smaller of x1 and x2
02287         if(x1 <= x2) then
02288             d2k = 0d0
02289         else
02290             d2k = 4d0/x1**6 - 4d0*x2/x1**7
02291         end if
```

#### 3.9.2.6 pure real(kind(0d0)) function kernel_rp_2_3::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2303 of file RKHS.f90.

```
02303         implicit none
02304         real(kind(0d0)), intent(in) :: x
02305         real(kind(0d0)), dimension(:), intent(in) :: par
02306         df21 = 0d0
```

#### 3.9.2.7 pure real(kind(0d0)) function kernel_rp_2_3::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2324 of file RKHS.f90.

```
02324         implicit none
02325         real(kind(0d0)), intent(in) :: x
02326         real(kind(0d0)), dimension(:), intent(in) :: par
02327         df22 = 1d0
```

#### 3.9.2.8 pure real(kind(0d0)) function kernel_rp_2_3::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2345 of file RKHS.f90.

```
02345         implicit none
02346         real(kind(0d0)), intent(in) :: x
02347         real(kind(0d0)), dimension(:), intent(in) :: par
02348         df31 = -4d0/x**5
```

#### 3.9.2.9 pure real(kind(0d0)) function kernel_rp_2_3::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2366 of file RKHS.f90.

```
02366         implicit none
02367         real(kind(0d0)), intent(in) :: x
02368         real(kind(0d0)), dimension(:), intent(in) :: par
02369         df32 = -5d0/x**6
```

#### 3.9.2.10 pure real(kind(0d0)) function kernel_rp_2_3::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2270 of file RKHS.f90.

```
02270         implicit none
02271         real(kind(0d0)), intent(in) :: x1,x2
02272         real(kind(0d0)), dimension(:), intent(in) :: par
02273         !find larger/smaller of x1 and x2
02274         if(x1 <= x2) then
02275             dk = -2d0/(15d0*x2**5)
02276         else
02277             dk = 2d0/3d0*x2/x1**6 - 4d0/(5d0*x1**5)
02278         end if
```

#### 3.9.2.11 pure real(kind(0d0)) function kernel_rp_2_3::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2296 of file RKHS.f90.

```
02296         implicit none
02297         real(kind(0d0)), intent(in) :: x
02298         real(kind(0d0)), dimension(:), intent(in) :: par
02299         f21 = 1d0
```

#### 3.9.2.12 pure real(kind(0d0)) function kernel_rp_2_3::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2317 of file RKHS.f90.

```
02317         implicit none
02318         real(kind(0d0)), intent(in) :: x
02319         real(kind(0d0)), dimension(:), intent(in) :: par
02320         f22 = x
```

#### 3.9.2.13 pure real(kind(0d0)) function kernel_rp_2_3::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2338 of file RKHS.f90.

```
02338         implicit none
02339         real(kind(0d0)), intent(in) :: x
02340         real(kind(0d0)), dimension(:), intent(in) :: par
02341         f31 = 1d0/x**4
```

#### 3.9.2.14 pure real(kind(0d0)) function kernel_rp_2_3::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2359 of file RKHS.f90.

```
02359         implicit none
02360         real(kind(0d0)), intent(in) :: x
02361         real(kind(0d0)), dimension(:), intent(in) :: par
02362         f32 = 1d0/x**5
```

#### 3.9.2.15 pure real(kind(0d0)) function kernel_rp_2_3::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2253 of file RKHS.f90.

```
02253         implicit none
02254         real(kind(0d0)), intent(in) :: x1,x2
02255         real(kind(0d0))             :: xs,xl
02256         real(kind(0d0)), dimension(:), intent(in) :: par
02257         !find larger/smaller of x1 and x2
02258         if(x1 <= x2) then
02259             xs = x1
02260             xl = x2
02261         else
02262             xs = x2
02263             xl = x1
02264         end if
02265         k = 1d0/(5d0*xl**4) - 2d0/15d0 * xs/xl**5
```

### 3.9.3 Variable Documentation

#### 3.9.3.1 integer, parameter kernel_rp_2_3::m2 = 2

Definition at line 2246 of file RKHS.f90.

```
02246     integer, parameter :: m2   = 2
```

#### 3.9.3.2 integer, parameter kernel_rp_2_3::npar = 0

Definition at line 2247 of file RKHS.f90.

```
02247     integer, parameter :: npar = 0
```

#### 3.9.3.3 real(kind(0d0)), parameter kernel_rp_2_3::p21 = 1d0/5d0

Definition at line 2248 of file RKHS.f90.

```
02248     real(kind(0d0)), parameter :: p21 = 1d0/5d0  ,&
02249                                   p22 = -2d0/15d0
```

#### 3.9.3.4 real(kind(0d0)), parameter kernel_rp_2_3::p22 = -2d0/15d0

Definition at line 2248 of file RKHS.f90.

## 3.10 kernel_rp_2_4 Module Reference

Reciprocal power decay kernel with n = 2 and m = 4.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)

- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

**Variables**

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 2d0/15d0
- real(kind(0d0)), parameter p22 = -2d0/21d0

### 3.10.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 4.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 4. This corresponds to a 1/r^5 decay for values larger than the greatest point in the grid. The kernel is only defined for values in the interval [0,infinity).

### 3.10.2 Function/Subroutine Documentation

#### 3.10.2.1 pure real(kind(0d0)) function kernel_rp_2_4::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2656 of file RKHS.f90.

```
02656         implicit none
02657         real(kind(0d0)), intent(in) :: x
02658         real(kind(0d0)), dimension(:), intent(in) :: par
02659         d2f21 = 0d0
```

#### 3.10.2.2 pure real(kind(0d0)) function kernel_rp_2_4::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2677 of file RKHS.f90.

```
02677         implicit none
02678         real(kind(0d0)), intent(in) :: x
02679         real(kind(0d0)), dimension(:), intent(in) :: par
02680         d2f22 = 0d0
```

#### 3.10.2.3 pure real(kind(0d0)) function kernel_rp_2_4::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2698 of file RKHS.f90.

```
02698         implicit none
02699         real(kind(0d0)), intent(in) :: x
02700         real(kind(0d0)), dimension(:), intent(in) :: par
02701         d2f31 = 30d0/x**7
```

#### 3.10.2.4 pure real(kind(0d0)) function kernel_rp_2_4::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2719 of file RKHS.f90.

```
02719         implicit none
02720         real(kind(0d0)), intent(in) :: x
02721         real(kind(0d0)), dimension(:), intent(in) :: par
02722         d2f32 = 42d0/x**8
```

#### 3.10.2.5 pure real(kind(0d0)) function kernel_rp_2_4::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2629 of file RKHS.f90.

```
02629         implicit none
02630         real(kind(0d0)), intent(in) :: x1,x2
02631         real(kind(0d0)), dimension(:), intent(in) :: par
02632         !find larger/smaller of x1 and x2
02633         if(x1 <= x2) then
02634             d2k = 0d0
02635         else
02636             d2k = 4d0/x1**7 - 4d0*x2/x1**8
02637         end if
```

#### 3.10.2.6 pure real(kind(0d0)) function kernel_rp_2_4::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2649 of file RKHS.f90.

```
02649         implicit none
02650         real(kind(0d0)), intent(in) :: x
02651         real(kind(0d0)), dimension(:), intent(in) :: par
02652         df21 = 0d0
```

#### 3.10.2.7 pure real(kind(0d0)) function kernel_rp_2_4::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2670 of file RKHS.f90.

```
02670         implicit none
02671         real(kind(0d0)), intent(in) :: x
02672         real(kind(0d0)), dimension(:), intent(in) :: par
02673         df22 = 1d0
```

#### 3.10.2.8 pure real(kind(0d0)) function kernel_rp_2_4::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2691 of file RKHS.f90.

```
02691         implicit none
02692         real(kind(0d0)), intent(in) :: x
02693         real(kind(0d0)), dimension(:), intent(in) :: par
02694         df31 = -5d0/x**6
```

#### 3.10.2.9 pure real(kind(0d0)) function kernel_rp_2_4::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2712 of file RKHS.f90.

```
02712         implicit none
02713         real(kind(0d0)), intent(in) :: x
02714         real(kind(0d0)), dimension(:), intent(in) :: par
02715         df32 = -6d0/x**7
```

#### 3.10.2.10 pure real(kind(0d0)) function kernel_rp_2_4::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2616 of file RKHS.f90.

```
02616         implicit none
02617         real(kind(0d0)), intent(in) :: x1,x2
02618         real(kind(0d0)), dimension(:), intent(in) :: par
02619         !find larger/smaller of x1 and x2
02620         if(x1 <= x2) then
02621             dk = -2d0/(21d0*x2**6)
02622         else
02623             dk = -2d0/(3d0*x1**6) + 4d0/7d0 * x2/x1**7
02624         end if
```

#### 3.10.2.11 pure real(kind(0d0)) function kernel_rp_2_4::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2642 of file RKHS.f90.

```
02642         implicit none
02643         real(kind(0d0)), intent(in) :: x
02644         real(kind(0d0)), dimension(:), intent(in) :: par
02645         f21 = 1d0
```

#### 3.10.2.12 pure real(kind(0d0)) function kernel_rp_2_4::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2663 of file RKHS.f90.

```
02663         implicit none
02664         real(kind(0d0)), intent(in) :: x
02665         real(kind(0d0)), dimension(:), intent(in) :: par
02666         f22 = x
```

#### 3.10.2.13 pure real(kind(0d0)) function kernel_rp_2_4::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2684 of file RKHS.f90.

```
02684         implicit none
02685         real(kind(0d0)), intent(in) :: x
02686         real(kind(0d0)), dimension(:), intent(in) :: par
02687         f31 = 1d0/x**5
```

#### 3.10.2.14 pure real(kind(0d0)) function kernel_rp_2_4::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2705 of file RKHS.f90.

```
02705         implicit none
02706         real(kind(0d0)), intent(in) :: x
02707         real(kind(0d0)), dimension(:), intent(in) :: par
02708         f32 = 1d0/x**6
```

#### 3.10.2.15 pure real(kind(0d0)) function kernel_rp_2_4::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2599 of file RKHS.f90.

```
02599         implicit none
02600         real(kind(0d0)), intent(in) :: x1,x2
02601         real(kind(0d0))             :: xs,xl
02602         real(kind(0d0)), dimension(:), intent(in) :: par
```

```
02603         !find larger/smaller of x1 and x2
02604         if(x1 <= x2) then
02605             xs = x1
02606             xl = x2
02607         else
02608             xs = x2
02609             xl = x1
02610         end if
02611         k = 2d0/(15d0*xl**5) - 2d0/21d0*xs/xl**6
```

#### 3.10.3 Variable Documentation

##### 3.10.3.1 integer, parameter kernel_rp_2_4::m2 = 2

Definition at line 2592 of file RKHS.f90.

```
02592     integer, parameter :: m2   = 2
```

##### 3.10.3.2 integer, parameter kernel_rp_2_4::npar = 0

Definition at line 2593 of file RKHS.f90.

```
02593     integer, parameter :: npar = 0
```

##### 3.10.3.3 real(kind(0d0)), parameter kernel_rp_2_4::p21 = 2d0/15d0

Definition at line 2594 of file RKHS.f90.

```
02594     real(kind(0d0)), parameter :: p21 = 2d0/15d0,&
02595                                   p22 = -2d0/21d0
```

##### 3.10.3.4 real(kind(0d0)), parameter kernel_rp_2_4::p22 = -2d0/21d0

Definition at line 2594 of file RKHS.f90.

### 3.11 kernel_rp_2_5 Module Reference

Reciprocal power decay kernel with n = 2 and m = 5.

#### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

**Variables**

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 2d0/21d0
- real(kind(0d0)), parameter p22 = -1d0/14d0

### 3.11.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 5.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 5. This corresponds to a 1/r^6 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling dipole-induced dipole and dispersion interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.11.2 Function/Subroutine Documentation

#### 3.11.2.1 pure real(kind(0d0)) function kernel_rp_2_5::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3002 of file RKHS.f90.

```
03002         implicit none
03003         real(kind(0d0)), intent(in) :: x
03004         real(kind(0d0)), dimension(:), intent(in) :: par
03005         d2f21 = 0d0
```

#### 3.11.2.2 pure real(kind(0d0)) function kernel_rp_2_5::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3023 of file RKHS.f90.

```
03023         implicit none
03024         real(kind(0d0)), intent(in) :: x
03025         real(kind(0d0)), dimension(:), intent(in) :: par
03026         d2f22 = 0d0
```

#### 3.11.2.3 pure real(kind(0d0)) function kernel_rp_2_5::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3044 of file RKHS.f90.

```
03044         implicit none
03045         real(kind(0d0)), intent(in) :: x
03046         real(kind(0d0)), dimension(:), intent(in) :: par
03047         d2f31 = 42d0/x**8
```

#### 3.11.2.4 pure real(kind(0d0)) function kernel_rp_2_5::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3065 of file RKHS.f90.

```
03065         implicit none
03066         real(kind(0d0)), intent(in) :: x
03067         real(kind(0d0)), dimension(:), intent(in) :: par
03068         d2f32 = 56d0/x**9
```

#### 3.11.2.5 pure real(kind(0d0)) function kernel_rp_2_5::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2975 of file RKHS.f90.

```
02975         implicit none
02976         real(kind(0d0)), intent(in) :: x1,x2
02977         real(kind(0d0)), dimension(:), intent(in) :: par
02978         !find larger/smaller of x1 and x2
02979         if(x1 <= x2) then
02980             d2k = 0d0
02981         else
02982             d2k = 4d0/x1**8 - 4d0*x2/x1**9
02983         end if
```

#### 3.11.2.6 pure real(kind(0d0)) function kernel_rp_2_5::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2995 of file RKHS.f90.

```
02995         implicit none
02996         real(kind(0d0)), intent(in) :: x
02997         real(kind(0d0)), dimension(:), intent(in) :: par
02998         df21 = 0d0
```

#### 3.11.2.7 pure real(kind(0d0)) function kernel_rp_2_5::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3016 of file RKHS.f90.

```
03016         implicit none
03017         real(kind(0d0)), intent(in) :: x
03018         real(kind(0d0)), dimension(:), intent(in) :: par
03019         df22 = 1d0
```

#### 3.11.2.8 pure real(kind(0d0)) function kernel_rp_2_5::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3037 of file RKHS.f90.

```
03037         implicit none
03038         real(kind(0d0)), intent(in) :: x
03039         real(kind(0d0)), dimension(:), intent(in) :: par
03040         df31 = -6d0/x**7
```

#### 3.11.2.9 pure real(kind(0d0)) function kernel_rp_2_5::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3058 of file RKHS.f90.

```
03058         implicit none
03059         real(kind(0d0)), intent(in) :: x
03060         real(kind(0d0)), dimension(:), intent(in) :: par
03061         df32 = -7d0/x**8
```

#### 3.11.2.10 pure real(kind(0d0)) function kernel_rp_2_5::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2962 of file RKHS.f90.

```
02962         implicit none
02963         real(kind(0d0)), intent(in) :: x1,x2
02964         real(kind(0d0)), dimension(:), intent(in) :: par
02965         !find larger/smaller of x1 and x2
02966         if(x1 <= x2) then
02967             dk = -1d0/(14d0*x2**7)
02968         else
02969             dk = 1d0*x2/(2d0*x1**8) - 4d0/(7d0*x1**7)
02970         end if
```

#### 3.11.2.11 pure real(kind(0d0)) function kernel_rp_2_5::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2988 of file RKHS.f90.

```
02988         implicit none
02989         real(kind(0d0)), intent(in) :: x
02990         real(kind(0d0)), dimension(:), intent(in) :: par
02991         f21 = 1d0
```

#### 3.11.2.12 pure real(kind(0d0)) function kernel_rp_2_5::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3009 of file RKHS.f90.

```
03009         implicit none
03010         real(kind(0d0)), intent(in) :: x
03011         real(kind(0d0)), dimension(:), intent(in) :: par
03012         f22 = x
```

#### 3.11.2.13 pure real(kind(0d0)) function kernel_rp_2_5::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3030 of file RKHS.f90.

```
03030         implicit none
03031         real(kind(0d0)), intent(in) :: x
03032         real(kind(0d0)), dimension(:), intent(in) :: par
03033         f31 = 1d0/x**6
```

#### 3.11.2.14 pure real(kind(0d0)) function kernel_rp_2_5::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3051 of file RKHS.f90.

```
03051         implicit none
03052         real(kind(0d0)), intent(in) :: x
03053         real(kind(0d0)), dimension(:), intent(in) :: par
03054         f32 = 1d0/x**7
```

#### 3.11.2.15 pure real(kind(0d0)) function kernel_rp_2_5::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2945 of file RKHS.f90.

```
02945         implicit none
02946         real(kind(0d0)), intent(in) :: x1,x2
02947         real(kind(0d0))             :: xs,xl
02948         real(kind(0d0)), dimension(:), intent(in) :: par
02949         !find larger/smaller of x1 and x2
02950         if(x1 <= x2) then
02951             xs = x1
02952             xl = x2
02953         else
02954             xs = x2
02955             xl = x1
02956         end if
02957         k = 2d0/(21d0 * xl**6) - 1d0/14d0 * xs/xl**7
```

### 3.11.3 Variable Documentation

#### 3.11.3.1 integer, parameter kernel_rp_2_5::m2 = 2

Definition at line 2938 of file RKHS.f90.

```
02938     integer, parameter :: m2   = 2
```

#### 3.11.3.2 integer, parameter kernel_rp_2_5::npar = 0

Definition at line 2939 of file RKHS.f90.

```
02939     integer, parameter :: npar = 0
```

#### 3.11.3.3 real(kind(0d0)), parameter kernel_rp_2_5::p21 = 2d0/21d0

Definition at line 2940 of file RKHS.f90.

```
02940     real(kind(0d0)), parameter :: p21 = 2d0/21d0  ,&
02941                                   p22 = -1d0/14d0
```

#### 3.11.3.4 real(kind(0d0)), parameter kernel_rp_2_5::p22 = -1d0/14d0

Definition at line 2940 of file RKHS.f90.

## 3.12 kernel_rp_2_6 Module Reference

Reciprocal power decay kernel with n = 2 and m = 6.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)

### Variables

- integer, parameter m2 = 2
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 1d0/14d0
- real(kind(0d0)), parameter p22 = -1d0/18d0

### 3.12.1 Detailed Description

Reciprocal power decay kernel with n = 2 and m = 6.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 2 and m = 6. This corresponds to a 1/r^7 decay for values larger than the greatest point in the grid. The kernel is only defined for values in the interval [0,infinity).

### 3.12.2 Function/Subroutine Documentation

#### 3.12.2.1 pure real(kind(0d0)) function kernel_rp_2_6::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3348 of file RKHS.f90.

```
03348         implicit none
03349         real(kind(0d0)), intent(in) :: x
03350         real(kind(0d0)), dimension(:), intent(in) :: par
03351         d2f21 = 0d0
```

#### 3.12.2.2 pure real(kind(0d0)) function kernel_rp_2_6::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3369 of file RKHS.f90.

```
03369         implicit none
03370         real(kind(0d0)), intent(in) :: x
03371         real(kind(0d0)), dimension(:), intent(in) :: par
03372         d2f22 = 0d0
```

#### 3.12.2.3 pure real(kind(0d0)) function kernel_rp_2_6::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3390 of file RKHS.f90.

```
03390         implicit none
03391         real(kind(0d0)), intent(in) :: x
03392         real(kind(0d0)), dimension(:), intent(in) :: par
03393         d2f31 = 56d0/x**9
```

#### 3.12.2.4 pure real(kind(0d0)) function kernel_rp_2_6::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3411 of file RKHS.f90.

```
03411         implicit none
03412         real(kind(0d0)), intent(in) :: x
03413         real(kind(0d0)), dimension(:), intent(in) :: par
03414         d2f32 = 72d0/x**10
```

#### 3.12.2.5 pure real(kind(0d0)) function kernel_rp_2_6::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3321 of file RKHS.f90.

```
03321         implicit none
03322         real(kind(0d0)), intent(in) :: x1,x2
03323         real(kind(0d0)), dimension(:), intent(in) :: par
03324         !find larger/smaller of x1 and x2
03325         if(x1 <= x2) then
03326             d2k = 0d0
03327         else
03328             d2k = 4d0/x1**9 - 4d0*x2/x1**10
03329         end if
```

#### 3.12.2.6 pure real(kind(0d0)) function kernel_rp_2_6::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3341 of file RKHS.f90.

```
03341         implicit none
03342         real(kind(0d0)), intent(in) :: x
03343         real(kind(0d0)), dimension(:), intent(in) :: par
03344         df21 = 0d0
```

#### 3.12.2.7 pure real(kind(0d0)) function kernel_rp_2_6::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3362 of file RKHS.f90.

```
03362         implicit none
03363         real(kind(0d0)), intent(in) :: x
03364         real(kind(0d0)), dimension(:), intent(in) :: par
03365         df22 = 1d0
```

#### 3.12.2.8 pure real(kind(0d0)) function kernel_rp_2_6::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3383 of file RKHS.f90.

```
03383         implicit none
03384         real(kind(0d0)), intent(in) :: x
03385         real(kind(0d0)), dimension(:), intent(in) :: par
03386         df31 = -7d0/x**8
```

#### 3.12.2.9 pure real(kind(0d0)) function kernel_rp_2_6::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3404 of file RKHS.f90.

```
03404         implicit none
03405         real(kind(0d0)), intent(in) :: x
03406         real(kind(0d0)), dimension(:), intent(in) :: par
03407         df32 = -8d0/x**9
```

#### 3.12.2.10 pure real(kind(0d0)) function kernel_rp_2_6::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3308 of file RKHS.f90.

```
03308         implicit none
03309         real(kind(0d0)), intent(in) :: x1,x2
03310         real(kind(0d0)), dimension(:), intent(in) :: par
03311         !find larger/smaller of x1 and x2
03312         if(x1 <= x2) then
03313             dk = -1d0/(18d0*x2**8)
03314         else
03315             dk = 4d0/9d0 * x2/x1**9 - 1d0/(2d0*x1**8)
03316         end if
```

#### 3.12.2.11 pure real(kind(0d0)) function kernel_rp_2_6::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3334 of file RKHS.f90.

```
03334         implicit none
03335         real(kind(0d0)), intent(in) :: x
03336         real(kind(0d0)), dimension(:), intent(in) :: par
03337         f21 = 1d0
```

#### 3.12.2.12 pure real(kind(0d0)) function kernel_rp_2_6::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3355 of file RKHS.f90.

```
03355         implicit none
03356         real(kind(0d0)), intent(in) :: x
03357         real(kind(0d0)), dimension(:), intent(in) :: par
03358         f22 = x
```

#### 3.12.2.13 pure real(kind(0d0)) function kernel_rp_2_6::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3376 of file RKHS.f90.

```
03376         implicit none
03377         real(kind(0d0)), intent(in) :: x
03378         real(kind(0d0)), dimension(:), intent(in) :: par
03379         f31 = 1d0/x**7
```

#### 3.12.2.14 pure real(kind(0d0)) function kernel_rp_2_6::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3397 of file RKHS.f90.

```
03397         implicit none
03398         real(kind(0d0)), intent(in) :: x
03399         real(kind(0d0)), dimension(:), intent(in) :: par
03400         f32 = 1d0/x**8
```

#### 3.12.2.15 pure real(kind(0d0)) function kernel_rp_2_6::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3291 of file RKHS.f90.

```
03291         implicit none
03292         real(kind(0d0)), intent(in) :: x1,x2
03293         real(kind(0d0))             :: xs,xl
03294         real(kind(0d0)), dimension(:), intent(in) :: par
03295         !find larger/smaller of x1 and x2
03296         if(x1 <= x2) then
03297             xs = x1
03298             xl = x2
03299         else
03300             xs = x2
03301             xl = x1
03302         end if
03303         k = 1d0/(14d0*xl**7) - 1d0/18d0 * xs/xl**8
```

### 3.12.3 Variable Documentation

#### 3.12.3.1 integer, parameter kernel_rp_2_6::m2 = 2

Definition at line 3284 of file RKHS.f90.

```
03284     integer, parameter :: m2   = 2
```

#### 3.12.3.2 integer, parameter kernel_rp_2_6::npar = 0

Definition at line 3285 of file RKHS.f90.

```
03285     integer, parameter :: npar = 0
```

#### 3.12.3.3 real(kind(0d0)), parameter kernel_rp_2_6::p21 = 1d0/14d0

Definition at line 3286 of file RKHS.f90.

```
03286     real(kind(0d0)), parameter :: p21 = 1d0/14d0,&
03287                                   p22 = -1d0/18d0
```

#### 3.12.3.4 real(kind(0d0)), parameter kernel_rp_2_6::p22 = -1d0/18d0

Definition at line 3286 of file RKHS.f90.

## 3.13 kernel_rp_3_0 Module Reference

Reciprocal power decay kernel with n = 3 and m = 0.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0
- real(kind(0d0)), parameter p22 = -3d0/2d0
- real(kind(0d0)), parameter p23 = 3d0/10d0

### 3.13.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 0.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 0. This corresponds to a 1/r decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-charge interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.13.2 Function/Subroutine Documentation

#### 3.13.2.1 pure real(kind(0d0)) function kernel_rp_3_0::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1422 of file RKHS.f90.

```
01422         implicit none
01423         real(kind(0d0)), intent(in) :: x
01424         real(kind(0d0)), dimension(:), intent(in) :: par
01425         d2f21 = 0d0
```

#### 3.13.2.2 pure real(kind(0d0)) function kernel_rp_3_0::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1443 of file RKHS.f90.

```
01443         implicit none
01444         real(kind(0d0)), intent(in) :: x
01445         real(kind(0d0)), dimension(:), intent(in) :: par
01446         d2f22 = 0d0
```

#### 3.13.2.3 pure real(kind(0d0)) function kernel_rp_3_0::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1464 of file RKHS.f90.

```
01464         implicit none
01465         real(kind(0d0)), intent(in) :: x
01466         real(kind(0d0)), dimension(:), intent(in) :: par
01467         d2f23 = 2d0
```

#### 3.13.2.4 pure real(kind(0d0)) function kernel_rp_3_0::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1485 of file RKHS.f90.

```
01485         implicit none
01486         real(kind(0d0)), intent(in) :: x
01487         real(kind(0d0)), dimension(:), intent(in) :: par
01488         d2f31 = 2d0/x**3
```

#### 3.13.2.5 pure real(kind(0d0)) function kernel_rp_3_0::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1506 of file RKHS.f90.

```
01506         implicit none
01507         real(kind(0d0)), intent(in) :: x
01508         real(kind(0d0)), dimension(:), intent(in) :: par
01509         d2f32 = 6d0/x**4
```

#### 3.13.2.6 pure real(kind(0d0)) function kernel_rp_3_0::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1527 of file RKHS.f90.

```
01527         implicit none
01528         real(kind(0d0)), intent(in) :: x
01529         real(kind(0d0)), dimension(:), intent(in) :: par
01530         d2f33 = 12d0/x**5
```

#### 3.13.2.7 pure real(kind(0d0)) function kernel_rp_3_0::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1395 of file RKHS.f90.

```
01395         implicit none
01396         real(kind(0d0)), intent(in) :: x1,x2
01397         real(kind(0d0)), dimension(:), intent(in) :: par
01398         !find larger/smaller of x1 and x2
01399         if(x1 <= x2) then
01400             d2k = 3d0/(5d0*x2**3)
01401         else
01402             d2k = 6d0/x1**3 - 9d0*x2/x1**4 + 18d0/5d0*x2**2/x1**5
01403         end if
```

#### 3.13.2.8 pure real(kind(0d0)) function kernel_rp_3_0::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1415 of file RKHS.f90.

```
01415         implicit none
01416         real(kind(0d0)), intent(in) :: x
01417         real(kind(0d0)), dimension(:), intent(in) :: par
01418         df21 = 0d0
```

#### 3.13.2.9 pure real(kind(0d0)) function kernel_rp_3_0::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1436 of file RKHS.f90.

```
01436         implicit none
01437         real(kind(0d0)), intent(in) :: x
01438         real(kind(0d0)), dimension(:), intent(in) :: par
01439         df22 = 1d0
```

#### 3.13.2.10 pure real(kind(0d0)) function kernel_rp_3_0::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1457 of file RKHS.f90.

```
01457         implicit none
01458         real(kind(0d0)), intent(in) :: x
01459         real(kind(0d0)), dimension(:), intent(in) :: par
01460         df23 = 2d0*x
```

#### 3.13.2.11 pure real(kind(0d0)) function kernel_rp_3_0::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1478 of file RKHS.f90.

```
01478         implicit none
01479         real(kind(0d0)), intent(in) :: x
01480         real(kind(0d0)), dimension(:), intent(in) :: par
01481         df31 = -1d0/x**2
```

#### 3.13.2.12 pure real(kind(0d0)) function kernel_rp_3_0::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1499 of file RKHS.f90.

```
01499         implicit none
01500         real(kind(0d0)), intent(in) :: x
01501         real(kind(0d0)), dimension(:), intent(in) :: par
01502         df32 = -2d0/x**3
```

#### 3.13.2.13 pure real(kind(0d0)) function kernel_rp_3_0::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1520 of file RKHS.f90.

```
01520         implicit none
01521         real(kind(0d0)), intent(in) :: x
01522         real(kind(0d0)), dimension(:), intent(in) :: par
01523         df33 = -3d0/x**4
```

#### 3.13.2.14 pure real(kind(0d0)) function kernel_rp_3_0::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1382 of file RKHS.f90.

```
01382         implicit none
01383         real(kind(0d0)), intent(in) :: x1,x2
01384         real(kind(0d0)), dimension(:), intent(in) :: par
01385         !find larger/smaller of x1 and x2
01386         if(x1 <= x2) then
01387             dk = 3d0/5d0 * x1/x2**3 - 3d0/(2d0*x2**2)
01388         else
01389             dk = -3d0/x1**2 + 3d0*x2/x1**3 - 9d0/10d0 * x2**2/x1**4
01390         end if
```

#### 3.13.2.15 pure real(kind(0d0)) function kernel_rp_3_0::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1408 of file RKHS.f90.

```
01408         implicit none
01409         real(kind(0d0)), intent(in) :: x
01410         real(kind(0d0)), dimension(:), intent(in) :: par
01411         f21 = 1d0
```

#### 3.13.2.16 pure real(kind(0d0)) function kernel_rp_3_0::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1429 of file RKHS.f90.

```
01429         implicit none
01430         real(kind(0d0)), intent(in) :: x
01431         real(kind(0d0)), dimension(:), intent(in) :: par
01432         f22 = x
```

#### 3.13.2.17 pure real(kind(0d0)) function kernel_rp_3_0::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1450 of file RKHS.f90.

```
01450         implicit none
01451         real(kind(0d0)), intent(in) :: x
01452         real(kind(0d0)), dimension(:), intent(in) :: par
01453         f23 = x**2
```

#### 3.13.2.18 pure real(kind(0d0)) function kernel_rp_3_0::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1471 of file RKHS.f90.

```
01471         implicit none
01472         real(kind(0d0)), intent(in) :: x
01473         real(kind(0d0)), dimension(:), intent(in) :: par
01474         f31 = 1d0/x
```

#### 3.13.2.19 pure real(kind(0d0)) function kernel_rp_3_0::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1492 of file RKHS.f90.

```
01492         implicit none
01493         real(kind(0d0)), intent(in) :: x
01494         real(kind(0d0)), dimension(:), intent(in) :: par
01495         f32 = 1d0/x**2
```

#### 3.13.2.20 pure real(kind(0d0)) function kernel_rp_3_0::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1513 of file RKHS.f90.

```
01513         implicit none
01514         real(kind(0d0)), intent(in) :: x
01515         real(kind(0d0)), dimension(:), intent(in) :: par
01516         f33 = 1d0/x**3
```

#### 3.13.2.21 pure real(kind(0d0)) function kernel_rp_3_0::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1365 of file RKHS.f90.

```
01365         implicit none
01366         real(kind(0d0)), intent(in) :: x1,x2
01367         real(kind(0d0))             :: xs,xl
01368         real(kind(0d0)), dimension(:), intent(in) :: par
01369         !find larger/smaller of x1 and x2
01370         if(x1 <= x2) then
01371             xs = x1
01372             xl = x2
01373         else
01374             xs = x2
01375             xl = x1
01376         end if
01377         k = 3d0/xl - 3d0/2d0 * xs/xl**2 + 3d0/10d0 * xs**2/xl**3
```

### 3.13.3 Variable Documentation

#### 3.13.3.1 integer, parameter kernel_rp_3_0::m2 = 3

Definition at line 1357 of file RKHS.f90.

```
01357     integer, parameter :: m2   = 3
```

#### 3.13.3.2 integer, parameter kernel_rp_3_0::npar = 0

Definition at line 1358 of file RKHS.f90.

```
01358     integer, parameter :: npar = 0
```

#### 3.13.3.3 real(kind(0d0)), parameter kernel_rp_3_0::p21 = 3d0

Definition at line 1359 of file RKHS.f90.

```
01359     real(kind(0d0)), parameter :: p21 = 3d0,&
01360                                   p22 = -3d0/2d0,&
01361                                   p23 = 3d0/10d0
```

#### 3.13.3.4 real(kind(0d0)), parameter kernel_rp_3_0::p22 = -3d0/2d0

Definition at line 1359 of file RKHS.f90.

#### 3.13.3.5 real(kind(0d0)), parameter kernel_rp_3_0::p23 = 3d0/10d0

Definition at line 1359 of file RKHS.f90.

## 3.14 kernel_rp_3_1 Module Reference

Reciprocal power decay kernel with n = 3 and m = 1.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/4d0
- real(kind(0d0)), parameter p22 = -3d0/5d0
- real(kind(0d0)), parameter p23 = 3d0/20d0

### 3.14.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 1.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 1. This corresponds to a 1/r^2 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.14.2 Function/Subroutine Documentation

#### 3.14.2.1 pure real(kind(0d0)) function kernel_rp_3_1::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1769 of file RKHS.f90.

```
01769         implicit none
01770         real(kind(0d0)), intent(in) :: x
01771         real(kind(0d0)), dimension(:), intent(in) :: par
01772         d2f21 = 0d0
```

#### 3.14.2.2 pure real(kind(0d0)) function kernel_rp_3_1::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1790 of file RKHS.f90.

```
01790         implicit none
01791         real(kind(0d0)), intent(in) :: x
01792         real(kind(0d0)), dimension(:), intent(in) :: par
01793         d2f22 = 0d0
```

#### 3.14.2.3 pure real(kind(0d0)) function kernel_rp_3_1::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1811 of file RKHS.f90.

```
01811         implicit none
01812         real(kind(0d0)), intent(in) :: x
01813         real(kind(0d0)), dimension(:), intent(in) :: par
01814         d2f23 = 2d0
```

#### 3.14.2.4 pure real(kind(0d0)) function kernel_rp_3_1::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1832 of file RKHS.f90.

```
01832         implicit none
01833         real(kind(0d0)), intent(in) :: x
01834         real(kind(0d0)), dimension(:), intent(in) :: par
01835         d2f31 = 6d0/x**4
```

#### 3.14.2.5 pure real(kind(0d0)) function kernel_rp_3_1::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1853 of file RKHS.f90.

```
01853         implicit none
01854         real(kind(0d0)), intent(in) :: x
01855         real(kind(0d0)), dimension(:), intent(in) :: par
01856         d2f32 = 12d0/x**5
```

#### 3.14.2.6 pure real(kind(0d0)) function kernel_rp_3_1::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1874 of file RKHS.f90.

```
01874         implicit none
01875         real(kind(0d0)), intent(in) :: x
01876         real(kind(0d0)), dimension(:), intent(in) :: par
01877         d2f33 = 20d0/x**6
```

#### 3.14.2.7 pure real(kind(0d0)) function kernel_rp_3_1::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1742 of file RKHS.f90.

```
01742         implicit none
01743         real(kind(0d0)), intent(in) :: x1,x2
01744         real(kind(0d0)), dimension(:), intent(in) :: par
01745         !find larger/smaller of x1 and x2
01746         if(x1 <= x2) then
01747             d2k = 3d0/(10d0*x2**4)
01748         else
01749             d2k = 9d0/(2d0*x1**4) - 36d0/5d0*x2/x1**5 + 3d0*x2**2/x1**6
01750         end if
```

#### 3.14.2.8 pure real(kind(0d0)) function kernel_rp_3_1::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1762 of file RKHS.f90.

```
01762         implicit none
01763         real(kind(0d0)), intent(in) :: x
01764         real(kind(0d0)), dimension(:), intent(in) :: par
01765         df21 = 0d0
```

#### 3.14.2.9 pure real(kind(0d0)) function kernel_rp_3_1::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1783 of file RKHS.f90.

```
01783         implicit none
01784         real(kind(0d0)), intent(in) :: x
01785         real(kind(0d0)), dimension(:), intent(in) :: par
01786         df22 = 1d0
```

#### 3.14.2.10 pure real(kind(0d0)) function kernel_rp_3_1::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1804 of file RKHS.f90.

```
01804         implicit none
01805         real(kind(0d0)), intent(in) :: x
01806         real(kind(0d0)), dimension(:), intent(in) :: par
01807         df23 = 2d0*x
```

#### 3.14.2.11 pure real(kind(0d0)) function kernel_rp_3_1::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1825 of file RKHS.f90.

```
01825         implicit none
01826         real(kind(0d0)), intent(in) :: x
01827         real(kind(0d0)), dimension(:), intent(in) :: par
01828         df31 = -2d0/x**3
```

#### 3.14.2.12 pure real(kind(0d0)) function kernel_rp_3_1::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1846 of file RKHS.f90.

```
01846         implicit none
01847         real(kind(0d0)), intent(in) :: x
01848         real(kind(0d0)), dimension(:), intent(in) :: par
01849         df32 = -3d0/x**4
```

#### 3.14.2.13 pure real(kind(0d0)) function kernel_rp_3_1::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1867 of file RKHS.f90.

```
01867         implicit none
01868         real(kind(0d0)), intent(in) :: x
01869         real(kind(0d0)), dimension(:), intent(in) :: par
01870         df33 = -4d0/x**5
```

#### 3.14.2.14 pure real(kind(0d0)) function kernel_rp_3_1::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1729 of file RKHS.f90.

```
01729         implicit none
01730         real(kind(0d0)), intent(in) :: x1,x2
01731         real(kind(0d0)), dimension(:), intent(in) :: par
01732         !find larger/smaller of x1 and x2
01733         if(x1 <= x2) then
01734             dk = 3d0/10d0 * x1/x2**4 - 3d0/(5d0*x2**3)
01735         else
01736             dk = -3d0/(2d0*x1**3) + 9d0/5d0 * x2/x1**4 - 3d0/5d0 * x2**2/x1**5
01737         end if
```

#### 3.14.2.15 pure real(kind(0d0)) function kernel_rp_3_1::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1755 of file RKHS.f90.

```
01755         implicit none
01756         real(kind(0d0)), intent(in) :: x
01757         real(kind(0d0)), dimension(:), intent(in) :: par
01758         f21 = 1d0
```

#### 3.14.2.16 pure real(kind(0d0)) function kernel_rp_3_1::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1776 of file RKHS.f90.

```
01776         implicit none
01777         real(kind(0d0)), intent(in) :: x
01778         real(kind(0d0)), dimension(:), intent(in) :: par
01779         f22 = x
```

#### 3.14.2.17 pure real(kind(0d0)) function kernel_rp_3_1::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1797 of file RKHS.f90.

```
01797         implicit none
01798         real(kind(0d0)), intent(in) :: x
01799         real(kind(0d0)), dimension(:), intent(in) :: par
01800         f23 = x**2
```

#### 3.14.2.18 pure real(kind(0d0)) function kernel_rp_3_1::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1818 of file RKHS.f90.

```
01818         implicit none
01819         real(kind(0d0)), intent(in) :: x
01820         real(kind(0d0)), dimension(:), intent(in) :: par
01821         f31 = 1d0/x**2
```

#### 3.14.2.19 pure real(kind(0d0)) function kernel_rp_3_1::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1839 of file RKHS.f90.

```
01839         implicit none
01840         real(kind(0d0)), intent(in) :: x
01841         real(kind(0d0)), dimension(:), intent(in) :: par
01842         f32 = 1d0/x**3
```

#### 3.14.2.20 pure real(kind(0d0)) function kernel_rp_3_1::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1860 of file RKHS.f90.

```
01860         implicit none
01861         real(kind(0d0)), intent(in) :: x
01862         real(kind(0d0)), dimension(:), intent(in) :: par
01863         f33 = 1d0/x**4
```

#### 3.14.2.21 pure real(kind(0d0)) function kernel_rp_3_1::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 1712 of file RKHS.f90.

```
01712         implicit none
01713         real(kind(0d0)), intent(in) :: x1,x2
01714         real(kind(0d0))             :: xs,xl
01715         real(kind(0d0)), dimension(:), intent(in) :: par
01716         !find larger/smaller of x1 and x2
01717         if(x1 <= x2) then
01718             xs = x1
01719             xl = x2
01720         else
01721             xs = x2
01722             xl = x1
01723         end if
01724         k = 3d0/(4d0*xl**2) - 3d0/5d0 * xs/xl**3 + 3d0/20d0 * xs**2/xl**4
```

### 3.14.3 Variable Documentation

#### 3.14.3.1 integer, parameter kernel_rp_3_1::m2 = 3

Definition at line 1704 of file RKHS.f90.

```
01704     integer, parameter :: m2   = 3
```

#### 3.14.3.2 integer, parameter kernel_rp_3_1::npar = 0

Definition at line 1705 of file RKHS.f90.

```
01705     integer, parameter :: npar = 0
```

#### 3.14.3.3 real(kind(0d0)), parameter kernel_rp_3_1::p21 = 3d0/4d0

Definition at line 1706 of file RKHS.f90.

```
01706     real(kind(0d0)), parameter :: p21 = 3d0/4d0,&
01707                                   p22 = -3d0/5d0,&
01708                                   p23 = 3d0/20d0
```

#### 3.14.3.4 real(kind(0d0)), parameter kernel_rp_3_1::p22 = -3d0/5d0

Definition at line 1706 of file RKHS.f90.

#### 3.14.3.5 real(kind(0d0)), parameter kernel_rp_3_1::p23 = 3d0/20d0

Definition at line 1706 of file RKHS.f90.

## 3.15 kernel_rp_3_2 Module Reference

Reciprocal power decay kernel with n = 3 and m = 2.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/10d0
- real(kind(0d0)), parameter p22 = -3d0/10d0
- real(kind(0d0)), parameter p23 = 3d0/35d0

### 3.15.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 2.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 2. This corresponds to a 1/r^3 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling dipole-dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.15.2 Function/Subroutine Documentation

#### 3.15.2.1 pure real(kind(0d0)) function kernel_rp_3_2::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2116 of file RKHS.f90.

```
02116         implicit none
02117         real(kind(0d0)), intent(in) :: x
02118         real(kind(0d0)), dimension(:), intent(in) :: par
02119         d2f21 = 0d0
```

#### 3.15.2.2 pure real(kind(0d0)) function kernel_rp_3_2::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2137 of file RKHS.f90.

```
02137         implicit none
02138         real(kind(0d0)), intent(in) :: x
02139         real(kind(0d0)), dimension(:), intent(in) :: par
02140         d2f22 = 0d0
```

#### 3.15.2.3 pure real(kind(0d0)) function kernel_rp_3_2::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2158 of file RKHS.f90.

```
02158         implicit none
02159         real(kind(0d0)), intent(in) :: x
02160         real(kind(0d0)), dimension(:), intent(in) :: par
02161         d2f23 = 2d0
```

#### 3.15.2.4 pure real(kind(0d0)) function kernel_rp_3_2::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2179 of file RKHS.f90.

```
02179         implicit none
02180         real(kind(0d0)), intent(in) :: x
02181         real(kind(0d0)), dimension(:), intent(in) :: par
02182         d2f31 = 12d0/x**5
```

#### 3.15.2.5 pure real(kind(0d0)) function kernel_rp_3_2::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2200 of file RKHS.f90.

```
02200         implicit none
02201         real(kind(0d0)), intent(in) :: x
02202         real(kind(0d0)), dimension(:), intent(in) :: par
02203         d2f32 = 20d0/x**6
```

#### 3.15.2.6 pure real(kind(0d0)) function kernel_rp_3_2::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2221 of file RKHS.f90.

```
02221         implicit none
02222         real(kind(0d0)), intent(in) :: x
02223         real(kind(0d0)), dimension(:), intent(in) :: par
02224         d2f33 = 30d0/x**7
```

#### 3.15.2.7 pure real(kind(0d0)) function kernel_rp_3_2::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2089 of file RKHS.f90.

```
02089         implicit none
02090         real(kind(0d0)), intent(in) :: x1,x2
02091         real(kind(0d0)), dimension(:), intent(in) :: par
02092         !find larger/smaller of x1 and x2
02093         if(x1 <= x2) then
02094             d2k = 6d0/(35d0*x2**5)
02095         else
02096             d2k = 18d0/(5d0*x1**5) - 6d0*x2/x1**6 + 18d0/7d0*x2**2/x1**7
02097         end if
```

#### 3.15.2.8 pure real(kind(0d0)) function kernel_rp_3_2::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2109 of file RKHS.f90.

```
02109         implicit none
02110         real(kind(0d0)), intent(in) :: x
02111         real(kind(0d0)), dimension(:), intent(in) :: par
02112         df21 = 0d0
```

#### 3.15.2.9 pure real(kind(0d0)) function kernel_rp_3_2::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2130 of file RKHS.f90.

```
02130         implicit none
02131         real(kind(0d0)), intent(in) :: x
02132         real(kind(0d0)), dimension(:), intent(in) :: par
02133         df22 = 1d0
```

#### 3.15.2.10 pure real(kind(0d0)) function kernel_rp_3_2::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2151 of file RKHS.f90.

```
02151         implicit none
02152         real(kind(0d0)), intent(in) :: x
02153         real(kind(0d0)), dimension(:), intent(in) :: par
02154         df23 = 2d0*x
```

#### 3.15.2.11 pure real(kind(0d0)) function kernel_rp_3_2::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2172 of file RKHS.f90.

```
02172         implicit none
02173         real(kind(0d0)), intent(in) :: x
02174         real(kind(0d0)), dimension(:), intent(in) :: par
02175         df31 = -3d0/x**4
```

#### 3.15.2.12 pure real(kind(0d0)) function kernel_rp_3_2::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2193 of file RKHS.f90.

```
02193         implicit none
02194         real(kind(0d0)), intent(in) :: x
02195         real(kind(0d0)), dimension(:), intent(in) :: par
02196         df32 = -4d0/x**5
```

#### 3.15.2.13 pure real(kind(0d0)) function kernel_rp_3_2::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2214 of file RKHS.f90.

```
02214         implicit none
02215         real(kind(0d0)), intent(in) :: x
02216         real(kind(0d0)), dimension(:), intent(in) :: par
02217         df33 = -5d0/x**6
```

#### 3.15.2.14 pure real(kind(0d0)) function kernel_rp_3_2::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2076 of file RKHS.f90.

```
02076         implicit none
02077         real(kind(0d0)), intent(in) :: x1,x2
02078         real(kind(0d0)), dimension(:), intent(in) :: par
02079         !find larger/smaller of x1 and x2
02080         if(x1 <= x2) then
02081             dk = 6d0/35d0 * x1/x2**5 - 3d0/(10d0*x2**4)
02082         else
02083             dk = -9d0/(10d0*x1**4) + 6d0/5d0 * x2/x1**5 - 3d0/7d0 * x2**2/x1**6
02084         end if
```

#### 3.15.2.15 pure real(kind(0d0)) function kernel_rp_3_2::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2102 of file RKHS.f90.

```
02102         implicit none
02103         real(kind(0d0)), intent(in) :: x
02104         real(kind(0d0)), dimension(:), intent(in) :: par
02105         f21 = 1d0
```

#### 3.15.2.16 pure real(kind(0d0)) function kernel_rp_3_2::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2123 of file RKHS.f90.

```
02123         implicit none
02124         real(kind(0d0)), intent(in) :: x
02125         real(kind(0d0)), dimension(:), intent(in) :: par
02126         f22 = x
```

#### 3.15.2.17 pure real(kind(0d0)) function kernel_rp_3_2::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2144 of file RKHS.f90.

```
02144         implicit none
02145         real(kind(0d0)), intent(in) :: x
02146         real(kind(0d0)), dimension(:), intent(in) :: par
02147         f23 = x**2
```

#### 3.15.2.18 pure real(kind(0d0)) function kernel_rp_3_2::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2165 of file RKHS.f90.

```
02165         implicit none
02166         real(kind(0d0)), intent(in) :: x
02167         real(kind(0d0)), dimension(:), intent(in) :: par
02168         f31 = 1d0/x**3
```

#### 3.15.2.19 pure real(kind(0d0)) function kernel_rp_3_2::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2186 of file RKHS.f90.

```
02186         implicit none
02187         real(kind(0d0)), intent(in) :: x
02188         real(kind(0d0)), dimension(:), intent(in) :: par
02189         f32 = 1d0/x**4
```

#### 3.15.2.20 pure real(kind(0d0)) function kernel_rp_3_2::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2207 of file RKHS.f90.

```
02207         implicit none
02208         real(kind(0d0)), intent(in) :: x
02209         real(kind(0d0)), dimension(:), intent(in) :: par
02210         f33 = 1d0/x**5
```

#### 3.15.2.21 pure real(kind(0d0)) function kernel_rp_3_2::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2059 of file RKHS.f90.

```
02059         implicit none
02060         real(kind(0d0)), intent(in) :: x1,x2
02061         real(kind(0d0))             :: xs,xl
02062         real(kind(0d0)), dimension(:), intent(in) :: par
02063         !find larger/smaller of x1 and x2
02064         if(x1 <= x2) then
02065             xs = x1
02066             xl = x2
02067         else
02068             xs = x2
02069             xl = x1
02070         end if
02071         k = 3d0/(10d0*xl**3) - 3d0/10d0 * xs/xl**4 + 3d0/35d0 * xs**2/xl**5
```

### 3.15.3 Variable Documentation

#### 3.15.3.1 integer, parameter kernel_rp_3_2::m2 = 3

Definition at line 2051 of file RKHS.f90.

```
02051     integer, parameter :: m2   = 3
```

#### 3.15.3.2 integer, parameter kernel_rp_3_2::npar = 0

Definition at line 2052 of file RKHS.f90.

```
02052     integer, parameter :: npar = 0
```

#### 3.15.3.3 real(kind(0d0)), parameter kernel_rp_3_2::p21 = 3d0/10d0

Definition at line 2053 of file RKHS.f90.

```
02053     real(kind(0d0)), parameter :: p21 = 3d0/10d0,&
02054                                   p22 = -3d0/10d0,&
02055                                   p23 = 3d0/35d0
```

#### 3.15.3.4 real(kind(0d0)), parameter kernel_rp_3_2::p22 = -3d0/10d0

Definition at line 2053 of file RKHS.f90.

#### 3.15.3.5 real(kind(0d0)), parameter kernel_rp_3_2::p23 = 3d0/35d0

Definition at line 2053 of file RKHS.f90.

## 3.16 kernel_rp_3_3 Module Reference

Reciprocal power decay kernel with n = 3 and m = 3.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/20
- real(kind(0d0)), parameter p22 = -6d0/35d0
- real(kind(0d0)), parameter p23 = 3d0/56d0

### 3.16.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 3.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 3. This corresponds to a 1/r^4 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling charge-induced dipole interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.16.2 Function/Subroutine Documentation

#### 3.16.2.1 pure real(kind(0d0)) function kernel_rp_3_3::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2463 of file RKHS.f90.

```
02463         implicit none
02464         real(kind(0d0)), intent(in) :: x
02465         real(kind(0d0)), dimension(:), intent(in) :: par
02466         d2f21 = 0d0
```

#### 3.16.2.2 pure real(kind(0d0)) function kernel_rp_3_3::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2484 of file RKHS.f90.

```
02484         implicit none
02485         real(kind(0d0)), intent(in) :: x
02486         real(kind(0d0)), dimension(:), intent(in) :: par
02487         d2f22 = 0d0
```

#### 3.16.2.3 pure real(kind(0d0)) function kernel_rp_3_3::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2505 of file RKHS.f90.

```
02505         implicit none
02506         real(kind(0d0)), intent(in) :: x
02507         real(kind(0d0)), dimension(:), intent(in) :: par
02508         d2f23 = 2d0
```

#### 3.16.2.4 pure real(kind(0d0)) function kernel_rp_3_3::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2526 of file RKHS.f90.

```
02526         implicit none
02527         real(kind(0d0)), intent(in) :: x
02528         real(kind(0d0)), dimension(:), intent(in) :: par
02529         d2f31 = 20d0/x**6
```

#### 3.16.2.5 pure real(kind(0d0)) function kernel_rp_3_3::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2547 of file RKHS.f90.

```
02547         implicit none
02548         real(kind(0d0)), intent(in) :: x
02549         real(kind(0d0)), dimension(:), intent(in) :: par
02550         d2f32 = 30d0/x**7
```

#### 3.16.2.6 pure real(kind(0d0)) function kernel_rp_3_3::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2568 of file RKHS.f90.

```
02568         implicit none
02569         real(kind(0d0)), intent(in) :: x
02570         real(kind(0d0)), dimension(:), intent(in) :: par
02571         d2f33 = 42d0/x**8
```

#### 3.16.2.7 pure real(kind(0d0)) function kernel_rp_3_3::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2436 of file RKHS.f90.

```
02436         implicit none
02437         real(kind(0d0)), intent(in) :: x1,x2
02438         real(kind(0d0)), dimension(:), intent(in) :: par
02439         !find larger/smaller of x1 and x2
02440         if(x1 <= x2) then
02441             d2k = 3d0/(28d0*x2**6)
02442         else
02443             d2k = 3d0/x1**6 - 36d0/7d0*x2/x1**7 + 9d0/4d0*x2**2/x1**8
02444         end if
```

#### 3.16.2.8 pure real(kind(0d0)) function kernel_rp_3_3::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2456 of file RKHS.f90.

```
02456         implicit none
02457         real(kind(0d0)), intent(in) :: x
02458         real(kind(0d0)), dimension(:), intent(in) :: par
02459         df21 = 0d0
```

#### 3.16.2.9 pure real(kind(0d0)) function kernel_rp_3_3::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2477 of file RKHS.f90.

```
02477         implicit none
02478         real(kind(0d0)), intent(in) :: x
02479         real(kind(0d0)), dimension(:), intent(in) :: par
02480         df22 = 1d0
```

#### 3.16.2.10 pure real(kind(0d0)) function kernel_rp_3_3::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2498 of file RKHS.f90.

```
02498         implicit none
02499         real(kind(0d0)), intent(in) :: x
02500         real(kind(0d0)), dimension(:), intent(in) :: par
02501         df23 = 2d0*x
```

#### 3.16.2.11 pure real(kind(0d0)) function kernel_rp_3_3::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2519 of file RKHS.f90.

```
02519         implicit none
02520         real(kind(0d0)), intent(in) :: x
02521         real(kind(0d0)), dimension(:), intent(in) :: par
02522         df31 = -4d0/x**5
```

#### 3.16.2.12 pure real(kind(0d0)) function kernel_rp_3_3::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2540 of file RKHS.f90.

```
02540         implicit none
02541         real(kind(0d0)), intent(in) :: x
02542         real(kind(0d0)), dimension(:), intent(in) :: par
02543         df32 = -5d0/x**6
```

#### 3.16.2.13 pure real(kind(0d0)) function kernel_rp_3_3::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2561 of file RKHS.f90.

```
02561         implicit none
02562         real(kind(0d0)), intent(in) :: x
02563         real(kind(0d0)), dimension(:), intent(in) :: par
02564         df33 = -6d0/x**7
```

#### 3.16.2.14 pure real(kind(0d0)) function kernel_rp_3_3::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2423 of file RKHS.f90.

```
02423         implicit none
02424         real(kind(0d0)), intent(in) :: x1,x2
02425         real(kind(0d0)), dimension(:), intent(in) :: par
02426         !find larger/smaller of x1 and x2
02427         if(x1 <= x2) then
02428             dk = 3d0/28d0 * x1/x2**6 - 6d0/(35d0*x2**5)
02429         else
02430             dk = -3d0/(5d0*x1**5) + 6d0/7d0 * x2/x1**6 - 9d0/28d0 * x2**2/x1**7
02431         end if
```

#### 3.16.2.15 pure real(kind(0d0)) function kernel_rp_3_3::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2449 of file RKHS.f90.

```
02449         implicit none
02450         real(kind(0d0)), intent(in) :: x
02451         real(kind(0d0)), dimension(:), intent(in) :: par
02452         f21 = 1d0
```

#### 3.16.2.16 pure real(kind(0d0)) function kernel_rp_3_3::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2470 of file RKHS.f90.

```
02470         implicit none
02471         real(kind(0d0)), intent(in) :: x
02472         real(kind(0d0)), dimension(:), intent(in) :: par
02473         f22 = x
```

#### 3.16.2.17 pure real(kind(0d0)) function kernel_rp_3_3::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2491 of file RKHS.f90.

```
02491         implicit none
02492         real(kind(0d0)), intent(in) :: x
02493         real(kind(0d0)), dimension(:), intent(in) :: par
02494         f23 = x**2
```

#### 3.16.2.18 pure real(kind(0d0)) function kernel_rp_3_3::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2512 of file RKHS.f90.

```
02512         implicit none
02513         real(kind(0d0)), intent(in) :: x
02514         real(kind(0d0)), dimension(:), intent(in) :: par
02515         f31 = 1d0/x**4
```

#### 3.16.2.19 pure real(kind(0d0)) function kernel_rp_3_3::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2533 of file RKHS.f90.

```
02533         implicit none
02534         real(kind(0d0)), intent(in) :: x
02535         real(kind(0d0)), dimension(:), intent(in) :: par
02536         f32 = 1d0/x**5
```

#### 3.16.2.20 pure real(kind(0d0)) function kernel_rp_3_3::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2554 of file RKHS.f90.

```
02554         implicit none
02555         real(kind(0d0)), intent(in) :: x
02556         real(kind(0d0)), dimension(:), intent(in) :: par
02557         f33 = 1d0/x**6
```

#### 3.16.2.21 pure real(kind(0d0)) function kernel_rp_3_3::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2406 of file RKHS.f90.

```
02406         implicit none
02407         real(kind(0d0)), intent(in) :: x1,x2
02408         real(kind(0d0))             :: xs,xl
02409         real(kind(0d0)), dimension(:), intent(in) :: par
02410         !find larger/smaller of x1 and x2
02411         if(x1 <= x2) then
02412             xs = x1
02413             xl = x2
02414         else
02415             xs = x2
02416             xl = x1
02417         end if
02418         k = 3d0/(20d0*xl**4) - 6d0/35d0 * xs/xl**5 + 3d0/56d0 * xs**2/xl**6
```

### 3.16.3 Variable Documentation

#### 3.16.3.1 integer, parameter kernel_rp_3_3::m2 = 3

Definition at line 2398 of file RKHS.f90.

```
02398     integer, parameter :: m2   = 3
```

#### 3.16.3.2 integer, parameter kernel_rp_3_3::npar = 0

Definition at line 2399 of file RKHS.f90.

```
02399     integer, parameter :: npar = 0
```

#### 3.16.3.3 real(kind(0d0)), parameter kernel_rp_3_3::p21 = 3d0/20

Definition at line 2400 of file RKHS.f90.

```
02400     real(kind(0d0)), parameter :: p21 = 3d0/20,&
02401                                   p22 = -6d0/35d0,&
02402                                   p23 = 3d0/56d0
```

#### 3.16.3.4 real(kind(0d0)), parameter kernel_rp_3_3::p22 = -6d0/35d0

Definition at line 2400 of file RKHS.f90.

#### 3.16.3.5 real(kind(0d0)), parameter kernel_rp_3_3::p23 = 3d0/56d0

Definition at line 2400 of file RKHS.f90.

## 3.17 kernel_rp_3_4 Module Reference

Reciprocal power decay kernel with n = 3 and m = 4.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/35d0
- real(kind(0d0)), parameter p22 = -3d0/28d0
- real(kind(0d0)), parameter p23 = 1d0/28d0

### 3.17.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 4.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 4. This corresponds to a 1/r^5 decay for values larger than the greatest point in the grid. The kernel is only defined for values in the interval [0,infinity).

### 3.17.2 Function/Subroutine Documentation

#### 3.17.2.1 pure real(kind(0d0)) function kernel_rp_3_4::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2808 of file RKHS.f90.

```
02808         implicit none
02809         real(kind(0d0)), intent(in) :: x
02810         real(kind(0d0)), dimension(:), intent(in) :: par
02811         d2f21 = 0d0
```

#### 3.17.2.2 pure real(kind(0d0)) function kernel_rp_3_4::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2829 of file RKHS.f90.

```
02829         implicit none
02830         real(kind(0d0)), intent(in) :: x
02831         real(kind(0d0)), dimension(:), intent(in) :: par
02832         d2f22 = 0d0
```

#### 3.17.2.3 pure real(kind(0d0)) function kernel_rp_3_4::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2850 of file RKHS.f90.

```
02850         implicit none
02851         real(kind(0d0)), intent(in) :: x
02852         real(kind(0d0)), dimension(:), intent(in) :: par
02853         d2f23 = 2d0
```

#### 3.17.2.4 pure real(kind(0d0)) function kernel_rp_3_4::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2871 of file RKHS.f90.

```
02871         implicit none
02872         real(kind(0d0)), intent(in) :: x
02873         real(kind(0d0)), dimension(:), intent(in) :: par
02874         d2f31 = 30d0/x**7
```

#### 3.17.2.5 pure real(kind(0d0)) function kernel_rp_3_4::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2892 of file RKHS.f90.

```
02892         implicit none
02893         real(kind(0d0)), intent(in) :: x
02894         real(kind(0d0)), dimension(:), intent(in) :: par
02895         d2f32 = 42d0/x**8
```

#### 3.17.2.6 pure real(kind(0d0)) function kernel_rp_3_4::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2913 of file RKHS.f90.

```
02913         implicit none
02914         real(kind(0d0)), intent(in) :: x
02915         real(kind(0d0)), dimension(:), intent(in) :: par
02916         d2f33 = 56d0/x**9
```

#### 3.17.2.7 pure real(kind(0d0)) function kernel_rp_3_4::d2k ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2781 of file RKHS.f90.

```
02781         implicit none
02782         real(kind(0d0)), intent(in) :: x1,x2
02783         real(kind(0d0)), dimension(:), intent(in) :: par
02784         !find larger/smaller of x1 and x2
02785         if(x1 <= x2) then
02786             d2k = 1d0/(14d0*x2**7)
02787         else
02788             d2k = 18d0/(7d0*x1**7) - 9d0/2d0*x2/x1**8 + 2d0*x2**2/x1**9
02789         end if
```

#### 3.17.2.8 pure real(kind(0d0)) function kernel_rp_3_4::df21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2801 of file RKHS.f90.

```
02801         implicit none
02802         real(kind(0d0)), intent(in) :: x
02803         real(kind(0d0)), dimension(:), intent(in) :: par
02804         df21 = 0d0
```

#### 3.17.2.9 pure real(kind(0d0)) function kernel_rp_3_4::df22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2822 of file RKHS.f90.

```
02822         implicit none
02823         real(kind(0d0)), intent(in) :: x
02824         real(kind(0d0)), dimension(:), intent(in) :: par
02825         df22 = 1d0
```

#### 3.17.2.10 pure real(kind(0d0)) function kernel_rp_3_4::df23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2843 of file RKHS.f90.

```
02843         implicit none
02844         real(kind(0d0)), intent(in) :: x
02845         real(kind(0d0)), dimension(:), intent(in) :: par
02846         df23 = 2d0*x
```

#### 3.17.2.11 pure real(kind(0d0)) function kernel_rp_3_4::df31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2864 of file RKHS.f90.

```
02864         implicit none
02865         real(kind(0d0)), intent(in) :: x
02866         real(kind(0d0)), dimension(:), intent(in) :: par
02867         df31 = -5d0/x**6
```

#### 3.17.2.12 pure real(kind(0d0)) function kernel_rp_3_4::df32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2885 of file RKHS.f90.

```
02885         implicit none
02886         real(kind(0d0)), intent(in) :: x
02887         real(kind(0d0)), dimension(:), intent(in) :: par
02888         df32 = -6d0/x**7
```

#### 3.17.2.13 pure real(kind(0d0)) function kernel_rp_3_4::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2906 of file RKHS.f90.

```
02906         implicit none
02907         real(kind(0d0)), intent(in) :: x
02908         real(kind(0d0)), dimension(:), intent(in) :: par
02909         df33 = -7d0/x**8
```

#### 3.17.2.14 pure real(kind(0d0)) function kernel_rp_3_4::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2768 of file RKHS.f90.

```
02768         implicit none
02769         real(kind(0d0)), intent(in) :: x1,x2
02770         real(kind(0d0)), dimension(:), intent(in) :: par
02771         !find larger/smaller of x1 and x2
02772         if(x1 <= x2) then
02773             dk = 1d0/14d0 * x1/x2**7 - 3d0/(28d0*x2**6)
02774         else
02775             dk = -3d0/(7d0*x1**6) + 9d0/14d0 * x2/x1**7 - 1d0/4d0 * x2**2/x1**8
02776         end if
```

#### 3.17.2.15 pure real(kind(0d0)) function kernel_rp_3_4::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2794 of file RKHS.f90.

```
02794         implicit none
02795         real(kind(0d0)), intent(in) :: x
02796         real(kind(0d0)), dimension(:), intent(in) :: par
02797         f21 = 1d0
```

#### 3.17.2.16 pure real(kind(0d0)) function kernel_rp_3_4::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2815 of file RKHS.f90.

```
02815         implicit none
02816         real(kind(0d0)), intent(in) :: x
02817         real(kind(0d0)), dimension(:), intent(in) :: par
02818         f22 = x
```

#### 3.17.2.17 pure real(kind(0d0)) function kernel_rp_3_4::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2836 of file RKHS.f90.

```
02836         implicit none
02837         real(kind(0d0)), intent(in) :: x
02838         real(kind(0d0)), dimension(:), intent(in) :: par
02839         f23 = x**2
```

#### 3.17.2.18 pure real(kind(0d0)) function kernel_rp_3_4::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2857 of file RKHS.f90.

```
02857         implicit none
02858         real(kind(0d0)), intent(in) :: x
02859         real(kind(0d0)), dimension(:), intent(in) :: par
02860         f31 = 1d0/x**5
```

#### 3.17.2.19 pure real(kind(0d0)) function kernel_rp_3_4::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2878 of file RKHS.f90.

```
02878         implicit none
02879         real(kind(0d0)), intent(in) :: x
02880         real(kind(0d0)), dimension(:), intent(in) :: par
02881         f32 = 1d0/x**6
```

#### 3.17.2.20 pure real(kind(0d0)) function kernel_rp_3_4::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2899 of file RKHS.f90.

```
02899         implicit none
02900         real(kind(0d0)), intent(in) :: x
02901         real(kind(0d0)), dimension(:), intent(in) :: par
02902         f33 = 1d0/x**7
```

#### 3.17.2.21 pure real(kind(0d0)) function kernel_rp_3_4::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 2751 of file RKHS.f90.

```
02751         implicit none
02752         real(kind(0d0)), intent(in) :: x1,x2
02753         real(kind(0d0))             :: xs,xl
02754         real(kind(0d0)), dimension(:), intent(in) :: par
02755         !find larger/smaller of x1 and x2
02756         if(x1 <= x2) then
02757             xs = x1
02758             xl = x2
02759         else
02760             xs = x2
02761             xl = x1
02762         end if
02763         k = 3d0/(35d0*xl**5) - 3d0/28d0 * xs/xl**6 + 1d0/28d0 * xs**2/xl**7
```

### 3.17.3 Variable Documentation

#### 3.17.3.1 integer, parameter kernel_rp_3_4::m2 = 3

Definition at line 2743 of file RKHS.f90.

```
02743     integer, parameter :: m2   = 3
```

#### 3.17.3.2 integer, parameter kernel_rp_3_4::npar = 0

Definition at line 2744 of file RKHS.f90.

```
02744     integer, parameter :: npar = 0
```

#### 3.17.3.3 real(kind(0d0)), parameter kernel_rp_3_4::p21 = 3d0/35d0

Definition at line 2745 of file RKHS.f90.

```
02745     real(kind(0d0)), parameter :: p21 = 3d0/35d0,&
02746                                   p22 = -3d0/28d0,&
02747                                   p23 = 1d0/28d0
```

#### 3.17.3.4 real(kind(0d0)), parameter kernel_rp_3_4::p22 = -3d0/28d0

Definition at line 2745 of file RKHS.f90.

#### 3.17.3.5 real(kind(0d0)), parameter kernel_rp_3_4::p23 = 1d0/28d0

Definition at line 2745 of file RKHS.f90.

## 3.18 kernel_rp_3_5 Module Reference

Reciprocal power decay kernel with n = 3 and m = 5.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/56d0
- real(kind(0d0)), parameter p22 = -1d0/14d0
- real(kind(0d0)), parameter p23 = 1d0/40d0

### 3.18.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 5.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 5. This corresponds to a 1/r^6 decay for values larger than the greatest point in the grid. This kernel is recommended for modelling dipole-induced dipole and dispersion interactions. The kernel is only defined for values in the interval [0,infinity).

### 3.18.2 Function/Subroutine Documentation

#### 3.18.2.1 pure real(kind(0d0)) function kernel_rp_3_5::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3155 of file RKHS.f90.

```
03155         implicit none
03156         real(kind(0d0)), intent(in) :: x
03157         real(kind(0d0)), dimension(:), intent(in) :: par
03158         d2f21 = 0d0
```

#### 3.18.2.2 pure real(kind(0d0)) function kernel_rp_3_5::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3176 of file RKHS.f90.

```
03176         implicit none
03177         real(kind(0d0)), intent(in) :: x
03178         real(kind(0d0)), dimension(:), intent(in) :: par
03179         d2f22 = 0d0
```

#### 3.18.2.3 pure real(kind(0d0)) function kernel_rp_3_5::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3197 of file RKHS.f90.

```
03197         implicit none
03198         real(kind(0d0)), intent(in) :: x
03199         real(kind(0d0)), dimension(:), intent(in) :: par
03200         d2f23 = 2d0
```

#### 3.18.2.4 pure real(kind(0d0)) function kernel_rp_3_5::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3218 of file RKHS.f90.

```
03218         implicit none
03219         real(kind(0d0)), intent(in) :: x
03220         real(kind(0d0)), dimension(:), intent(in) :: par
03221         d2f31 = 42d0/x**8
```

#### 3.18.2.5 pure real(kind(0d0)) function kernel_rp_3_5::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3239 of file RKHS.f90.

```
03239         implicit none
03240         real(kind(0d0)), intent(in) :: x
03241         real(kind(0d0)), dimension(:), intent(in) :: par
03242         d2f32 = 56d0/x**9
```

#### 3.18.2.6 pure real(kind(0d0)) function kernel_rp_3_5::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3260 of file RKHS.f90.

```
03260         implicit none
03261         real(kind(0d0)), intent(in) :: x
03262         real(kind(0d0)), dimension(:), intent(in) :: par
03263         d2f33 = 72d0/x**10
```

#### 3.18.2.7 pure real(kind(0d0)) function kernel_rp_3_5::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3128 of file RKHS.f90.

```
03128         implicit none
03129         real(kind(0d0)), intent(in) :: x1,x2
03130         real(kind(0d0)), dimension(:), intent(in) :: par
03131         !find larger/smaller of x1 and x2
03132         if(x1 <= x2) then
03133             d2k = 1d0/(20d0*x2**8)
03134         else
03135             d2k = 9d0/(4d0*x1**8) - 4d0*x2/x1**9 + 9d0/5d0*x2**2/x1**10
03136         end if
```

#### 3.18.2.8 pure real(kind(0d0)) function kernel_rp_3_5::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3148 of file RKHS.f90.

```
03148         implicit none
03149         real(kind(0d0)), intent(in) :: x
03150         real(kind(0d0)), dimension(:), intent(in) :: par
03151         df21 = 0d0
```

#### 3.18.2.9 pure real(kind(0d0)) function kernel_rp_3_5::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3169 of file RKHS.f90.

```
03169         implicit none
03170         real(kind(0d0)), intent(in) :: x
03171         real(kind(0d0)), dimension(:), intent(in) :: par
03172         df22 = 1d0
```

#### 3.18.2.10 pure real(kind(0d0)) function kernel_rp_3_5::df23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3190 of file RKHS.f90.

```
03190         implicit none
03191         real(kind(0d0)), intent(in) :: x
03192         real(kind(0d0)), dimension(:), intent(in) :: par
03193         df23 = 2d0*x
```

#### 3.18.2.11 pure real(kind(0d0)) function kernel_rp_3_5::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3211 of file RKHS.f90.

```
03211         implicit none
03212         real(kind(0d0)), intent(in) :: x
03213         real(kind(0d0)), dimension(:), intent(in) :: par
03214         df31 = -6d0/x**7
```

#### 3.18.2.12 pure real(kind(0d0)) function kernel_rp_3_5::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3232 of file RKHS.f90.

```
03232         implicit none
03233         real(kind(0d0)), intent(in) :: x
03234         real(kind(0d0)), dimension(:), intent(in) :: par
03235         df32 = -7d0/x**8
```

#### 3.18.2.13 pure real(kind(0d0)) function kernel_rp_3_5::df33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3253 of file RKHS.f90.

```
03253         implicit none
03254         real(kind(0d0)), intent(in) :: x
03255         real(kind(0d0)), dimension(:), intent(in) :: par
03256         df33 = -8d0/x**9
```

#### 3.18.2.14 pure real(kind(0d0)) function kernel_rp_3_5::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3115 of file RKHS.f90.

```
03115         implicit none
03116         real(kind(0d0)), intent(in) :: x1,x2
03117         real(kind(0d0)), dimension(:), intent(in) :: par
03118         !find larger/smaller of x1 and x2
03119         if(x1 <= x2) then
03120             dk = 1d0/20d0 * x1/x2**8 - 1d0/(14d0*x2**7)
03121         else
03122             dk = -1d0/5d0 * x2**2/x1**9 + 0.5d0 * x2/x1**8 - 9d0/(28d0*x1**7)
03123         end if
```

#### 3.18.2.15 pure real(kind(0d0)) function kernel_rp_3_5::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3141 of file RKHS.f90.

```
03141         implicit none
03142         real(kind(0d0)), intent(in) :: x
03143         real(kind(0d0)), dimension(:), intent(in) :: par
03144         f21 = 1d0
```

#### 3.18.2.16 pure real(kind(0d0)) function kernel_rp_3_5::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3162 of file RKHS.f90.

```
03162         implicit none
03163         real(kind(0d0)), intent(in) :: x
03164         real(kind(0d0)), dimension(:), intent(in) :: par
03165         f22 = x
```

#### 3.18.2.17 pure real(kind(0d0)) function kernel_rp_3_5::f23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3183 of file RKHS.f90.

```
03183         implicit none
03184         real(kind(0d0)), intent(in) :: x
03185         real(kind(0d0)), dimension(:), intent(in) :: par
03186         f23 = x**2
```

#### 3.18.2.18 pure real(kind(0d0)) function kernel_rp_3_5::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3204 of file RKHS.f90.

```
03204         implicit none
03205         real(kind(0d0)), intent(in) :: x
03206         real(kind(0d0)), dimension(:), intent(in) :: par
03207         f31 = 1d0/x**6
```

#### 3.18.2.19 pure real(kind(0d0)) function kernel_rp_3_5::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3225 of file RKHS.f90.

```
03225         implicit none
03226         real(kind(0d0)), intent(in) :: x
03227         real(kind(0d0)), dimension(:), intent(in) :: par
03228         f32 = 1d0/x**7
```

#### 3.18.2.20 pure real(kind(0d0)) function kernel_rp_3_5::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3246 of file RKHS.f90.

```
03246         implicit none
03247         real(kind(0d0)), intent(in) :: x
03248         real(kind(0d0)), dimension(:), intent(in) :: par
03249         f33 = 1d0/x**8
```

#### 3.18.2.21 pure real(kind(0d0)) function kernel_rp_3_5::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3098 of file RKHS.f90.

```
03098         implicit none
03099         real(kind(0d0)), intent(in) :: x1,x2
03100         real(kind(0d0))             :: xs,xl
03101         real(kind(0d0)), dimension(:), intent(in) :: par
03102         !find larger/smaller of x1 and x2
03103         if(x1 <= x2) then
03104             xs = x1
03105             xl = x2
03106         else
03107             xs = x2
03108             xl = x1
03109         end if
03110         k = 3d0/(56d0*xl**6) - 1d0/14d0 * xs/xl**7 + 1d0/40d0 * xs**2/xl**8
```

### 3.18.3 Variable Documentation

#### 3.18.3.1 integer, parameter kernel_rp_3_5::m2 = 3

Definition at line 3090 of file RKHS.f90.

```
03090     integer, parameter :: m2   = 3
```

#### 3.18.3.2 integer, parameter kernel_rp_3_5::npar = 0

Definition at line 3091 of file RKHS.f90.

```
03091     integer, parameter :: npar = 0
```

#### 3.18.3.3 real(kind(0d0)), parameter kernel_rp_3_5::p21 = 3d0/56d0

Definition at line 3092 of file RKHS.f90.

```
03092     real(kind(0d0)), parameter :: p21 = 3d0/56d0,&
03093                                   p22 = -1d0/14d0,&
03094                                   p23 = 1d0/40d0
```

#### 3.18.3.4 real(kind(0d0)), parameter kernel_rp_3_5::p22 = -1d0/14d0

Definition at line 3092 of file RKHS.f90.

#### 3.18.3.5 real(kind(0d0)), parameter kernel_rp_3_5::p23 = 1d0/40d0

Definition at line 3092 of file RKHS.f90.

## 3.19 kernel_rp_3_6 Module Reference

Reciprocal power decay kernel with n = 3 and m = 6.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 1d0/28d0
- real(kind(0d0)), parameter p22 = -1d0/20d0
- real(kind(0d0)), parameter p23 = 1d0/55d0

### 3.19.1 Detailed Description

Reciprocal power decay kernel with n = 3 and m = 6.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the reciprocal power decay kernel with n = 3 and m = 6. This corresponds to a 1/r^7 decay for values larger than the greatest point in the grid. The kernel is only defined for values in the interval [0,infinity).

### 3.19.2 Function/Subroutine Documentation

#### 3.19.2.1 pure real(kind(0d0)) function kernel_rp_3_6::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3500 of file RKHS.f90.

```
03500         implicit none
03501         real(kind(0d0)), intent(in) :: x
03502         real(kind(0d0)), dimension(:), intent(in) :: par
03503         d2f21 = 0d0
```

#### 3.19.2.2 pure real(kind(0d0)) function kernel_rp_3_6::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3521 of file RKHS.f90.

```
03521         implicit none
03522         real(kind(0d0)), intent(in) :: x
03523         real(kind(0d0)), dimension(:), intent(in) :: par
03524         d2f22 = 0d0
```

#### 3.19.2.3 pure real(kind(0d0)) function kernel_rp_3_6::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3542 of file RKHS.f90.

```
03542         implicit none
03543         real(kind(0d0)), intent(in) :: x
03544         real(kind(0d0)), dimension(:), intent(in) :: par
03545         d2f23 = 2d0
```

#### 3.19.2.4 pure real(kind(0d0)) function kernel_rp_3_6::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3563 of file RKHS.f90.

```
03563         implicit none
03564         real(kind(0d0)), intent(in) :: x
03565         real(kind(0d0)), dimension(:), intent(in) :: par
03566         d2f31 = 56d0/x**9
```

#### 3.19.2.5 pure real(kind(0d0)) function kernel_rp_3_6::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3584 of file RKHS.f90.

```
03584         implicit none
03585         real(kind(0d0)), intent(in) :: x
03586         real(kind(0d0)), dimension(:), intent(in) :: par
03587         d2f32 = 72d0/x**10
```

#### 3.19.2.6 pure real(kind(0d0)) function kernel_rp_3_6::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3605 of file RKHS.f90.

```
03605         implicit none
03606         real(kind(0d0)), intent(in) :: x
03607         real(kind(0d0)), dimension(:), intent(in) :: par
03608         d2f33 = 90d0/x**11
```

#### 3.19.2.7 pure real(kind(0d0)) function kernel_rp_3_6::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3473 of file RKHS.f90.

```
03473         implicit none
03474         real(kind(0d0)), intent(in) :: x1,x2
03475         real(kind(0d0)), dimension(:), intent(in) :: par
03476         !find larger/smaller of x1 and x2
03477         if(x1 <= x2) then
03478             d2k = 2d0/(55d0*x2**9)
03479         else
03480             d2k = 2d0/x1**9 - 18d0/5d0*x2/x1**10 + 18d0/11d0*x2**2/x1**11
03481         end if
```

#### 3.19.2.8 pure real(kind(0d0)) function kernel_rp_3_6::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3493 of file RKHS.f90.

```
03493         implicit none
03494         real(kind(0d0)), intent(in) :: x
03495         real(kind(0d0)), dimension(:), intent(in) :: par
03496         df21 = 0d0
```

#### 3.19.2.9 pure real(kind(0d0)) function kernel_rp_3_6::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3514 of file RKHS.f90.

```
03514         implicit none
03515         real(kind(0d0)), intent(in) :: x
03516         real(kind(0d0)), dimension(:), intent(in) :: par
03517         df22 = 1d0
```

#### 3.19.2.10 pure real(kind(0d0)) function kernel_rp_3_6::df23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3535 of file RKHS.f90.

```
03535         implicit none
03536         real(kind(0d0)), intent(in) :: x
03537         real(kind(0d0)), dimension(:), intent(in) :: par
03538         df23 = 2d0*x
```

#### 3.19.2.11 pure real(kind(0d0)) function kernel_rp_3_6::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3556 of file RKHS.f90.

```
03556         implicit none
03557         real(kind(0d0)), intent(in) :: x
03558         real(kind(0d0)), dimension(:), intent(in) :: par
03559         df31 = -7d0/x**8
```

#### 3.19.2.12 pure real(kind(0d0)) function kernel_rp_3_6::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3577 of file RKHS.f90.

```
03577         implicit none
03578         real(kind(0d0)), intent(in) :: x
03579         real(kind(0d0)), dimension(:), intent(in) :: par
03580         df32 = -8d0/x**9
```

#### 3.19.2.13 pure real(kind(0d0)) function kernel_rp_3_6::df33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3598 of file RKHS.f90.

```
03598         implicit none
03599         real(kind(0d0)), intent(in) :: x
03600         real(kind(0d0)), dimension(:), intent(in) :: par
03601         df33 = -9d0/x**10
```

#### 3.19.2.14 pure real(kind(0d0)) function kernel_rp_3_6::dk ( real(kind(0d0)), intent(in) *x1*, real(kind(0d0)), intent(in) *x2*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3460 of file RKHS.f90.

```
03460         implicit none
03461         real(kind(0d0)), intent(in) :: x1,x2
03462         real(kind(0d0)), dimension(:), intent(in) :: par
03463         !find larger/smaller of x1 and x2
03464         if(x1 <= x2) then
03465             dk = 2d0/55d0 * x1/x2**9 - 1d0/(20d0*x2**8)
03466         else
03467             dk = -1d0/(4d0*x1**8) + 2d0/5d0 * x2/x1**9 - 9d0/55d0 * x2**2/x1**10
03468         end if
```

#### 3.19.2.15 pure real(kind(0d0)) function kernel_rp_3_6::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3486 of file RKHS.f90.

```
03486         implicit none
03487         real(kind(0d0)), intent(in) :: x
03488         real(kind(0d0)), dimension(:), intent(in) :: par
03489         f21 = 1d0
```

#### 3.19.2.16 pure real(kind(0d0)) function kernel_rp_3_6::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3507 of file RKHS.f90.

```
03507         implicit none
03508         real(kind(0d0)), intent(in) :: x
03509         real(kind(0d0)), dimension(:), intent(in) :: par
03510         f22 = x
```

#### 3.19.2.17 pure real(kind(0d0)) function kernel_rp_3_6::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3528 of file RKHS.f90.

```
03528         implicit none
03529         real(kind(0d0)), intent(in) :: x
03530         real(kind(0d0)), dimension(:), intent(in) :: par
03531         f23 = x**2
```

#### 3.19.2.18 pure real(kind(0d0)) function kernel_rp_3_6::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3549 of file RKHS.f90.

```
03549         implicit none
03550         real(kind(0d0)), intent(in) :: x
03551         real(kind(0d0)), dimension(:), intent(in) :: par
03552         f31 = 1d0/x**7
```

#### 3.19.2.19 pure real(kind(0d0)) function kernel_rp_3_6::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3570 of file RKHS.f90.

```
03570         implicit none
03571         real(kind(0d0)), intent(in) :: x
03572         real(kind(0d0)), dimension(:), intent(in) :: par
03573         f32 = 1d0/x**8
```

#### 3.19.2.20 pure real(kind(0d0)) function kernel_rp_3_6::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3591 of file RKHS.f90.

```
03591         implicit none
03592         real(kind(0d0)), intent(in) :: x
03593         real(kind(0d0)), dimension(:), intent(in) :: par
03594         f33 = 1d0/x**9
```

#### 3.19.2.21 pure real(kind(0d0)) function kernel_rp_3_6::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3443 of file RKHS.f90.

```
03443         implicit none
03444         real(kind(0d0)), intent(in) :: x1,x2
03445         real(kind(0d0))             :: xs,xl
03446         real(kind(0d0)), dimension(:), intent(in) :: par
03447         !find larger/smaller of x1 and x2
03448         if(x1 <= x2) then
03449             xs = x1
03450             xl = x2
03451         else
03452             xs = x2
03453             xl = x1
03454         end if
03455         k = 1d0/(28d0*xl**7) - 1d0/20d0 * xs/xl**8 + 1d0/55d0 * xs**2/xl**9
```

### 3.19.3 Variable Documentation

#### 3.19.3.1 integer, parameter kernel_rp_3_6::m2 = 3

Definition at line 3435 of file RKHS.f90.

```
03435     integer, parameter :: m2   = 3
```

#### 3.19.3.2 integer, parameter kernel_rp_3_6::npar = 0

Definition at line 3436 of file RKHS.f90.

```
03436     integer, parameter :: npar = 0
```

#### 3.19.3.3 real(kind(0d0)), parameter kernel_rp_3_6::p21 = 1d0/28d0

Definition at line 3437 of file RKHS.f90.

```
03437     real(kind(0d0)), parameter :: p21 = 1d0/28d0,&
03438                                   p22 = -1d0/20d0,&
03439                                   p23 = 1d0/55d0
```

#### 3.19.3.4 real(kind(0d0)), parameter kernel_rp_3_6::p22 = -1d0/20d0

Definition at line 3437 of file RKHS.f90.

#### 3.19.3.5 real(kind(0d0)), parameter kernel_rp_3_6::p23 = 1d0/55d0

Definition at line 3437 of file RKHS.f90.

## 3.20 kernel_ts_2 Module Reference

Taylor spline kernel with n = 2.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)

### Variables

- integer, parameter m2 = 3
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = -2d0/3d0
- real(kind(0d0)), parameter p22 = 1d0
- real(kind(0d0)), parameter p23 = 2d0

### 3.20.1 Detailed Description

Taylor spline kernel with n = 2.

#### Author

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the Taylor spline kernel with n = 2. Note that this kernel is only defined for values in the interval [0,1]. It is useful for example in describing angular coordinates, as long as a new coordinate is introduced which scales the angular coordinate to the interval [0,1]. An example for such a coordinate would be y = (1d0-cos(alpha))/2d0. Note that by clever choice of y, it is possible to capture the symmetry inherent in the system. The kernel is also applicable to general machine learning problems to interpolate any arbitrary function defined in a finite interval.

### 3.20.2 Function/Subroutine Documentation

#### 3.20.2.1 pure real(kind(0d0)) function kernel_ts_2::d2f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3698 of file RKHS.f90.

```
03698         implicit none
03699         real(kind(0d0)), intent(in) :: x
03700         real(kind(0d0)), dimension(:), intent(in) :: par
03701         d2f21 = 6d0*x
```

#### 3.20.2.2 pure real(kind(0d0)) function kernel_ts_2::d2f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3719 of file RKHS.f90.

```
03719         implicit none
03720         real(kind(0d0)), intent(in) :: x
03721         real(kind(0d0)), dimension(:), intent(in) :: par
03722         d2f22 = 0d0
```

#### 3.20.2.3 pure real(kind(0d0)) function kernel_ts_2::d2f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3740 of file RKHS.f90.

```
03740         implicit none
03741         real(kind(0d0)), intent(in) :: x
03742         real(kind(0d0)), dimension(:), intent(in) :: par
03743         d2f23 = 2d0
```

#### 3.20.2.4 pure real(kind(0d0)) function kernel_ts_2::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3761 of file RKHS.f90.

```
03761         implicit none
03762         real(kind(0d0)), intent(in) :: x
03763         real(kind(0d0)), dimension(:), intent(in) :: par
03764         d2f31 = 0d0
```

#### 3.20.2.5 pure real(kind(0d0)) function kernel_ts_2::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3782 of file RKHS.f90.

```
03782         implicit none
03783         real(kind(0d0)), intent(in) :: x
03784         real(kind(0d0)), dimension(:), intent(in) :: par
03785         d2f32 = 0d0
```

#### 3.20.2.6 pure real(kind(0d0)) function kernel_ts_2::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3803 of file RKHS.f90.

```
03803         implicit none
03804         real(kind(0d0)), intent(in) :: x
03805         real(kind(0d0)), dimension(:), intent(in) :: par
03806         d2f33 = 0d0
```

#### 3.20.2.7 pure real(kind(0d0)) function kernel_ts_2::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3671 of file RKHS.f90.

```
03671         implicit none
03672         real(kind(0d0)), intent(in) :: x1,x2
03673         real(kind(0d0)), dimension(:), intent(in) :: par
03674         !find larger/smaller of x1 and x2
03675         if(x1 <= x2) then
03676             d2k = 4d0*x2 - 4d0*x1
03677         else
03678             d2k = 0d0
03679         end if
```

#### 3.20.2.8 pure real(kind(0d0)) function kernel_ts_2::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3691 of file RKHS.f90.

```
03691         implicit none
03692         real(kind(0d0)), intent(in) :: x
03693         real(kind(0d0)), dimension(:), intent(in) :: par
03694         df21 = 3d0*x**2
```

#### 3.20.2.9 pure real(kind(0d0)) function kernel_ts_2::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3712 of file RKHS.f90.

```
03712         implicit none
03713         real(kind(0d0)), intent(in) :: x
03714         real(kind(0d0)), dimension(:), intent(in) :: par
03715         df22 = 1d0
```

#### 3.20.2.10 pure real(kind(0d0)) function kernel_ts_2::df23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3733 of file RKHS.f90.

```
03733         implicit none
03734         real(kind(0d0)), intent(in) :: x
03735         real(kind(0d0)), dimension(:), intent(in) :: par
03736         df23 = 2d0*x
```

#### 3.20.2.11 pure real(kind(0d0)) function kernel_ts_2::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3754 of file RKHS.f90.

```
03754         implicit none
03755         real(kind(0d0)), intent(in) :: x
03756         real(kind(0d0)), dimension(:), intent(in) :: par
03757         df31 = 0d0
```

#### 3.20.2.12 pure real(kind(0d0)) function kernel_ts_2::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3775 of file RKHS.f90.

```
03775         implicit none
03776         real(kind(0d0)), intent(in) :: x
03777         real(kind(0d0)), dimension(:), intent(in) :: par
03778         df32 = 1d0
```

#### 3.20.2.13 pure real(kind(0d0)) function kernel_ts_2::df33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3796 of file RKHS.f90.

```
03796         implicit none
03797         real(kind(0d0)), intent(in) :: x
03798         real(kind(0d0)), dimension(:), intent(in) :: par
03799         df33 = 1d0
```

#### 3.20.2.14 pure real(kind(0d0)) function kernel_ts_2::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3658 of file RKHS.f90.

```
03658         implicit none
03659         real(kind(0d0)), intent(in) :: x1,x2
03660         real(kind(0d0)), dimension(:), intent(in) :: par
03661         !find larger/smaller of x1 and x2
03662         if(x1 <= x2) then
03663             dk = x2 + 4d0*x2*x1 - 2d0*x1**2
03664         else
03665             dk = x2*(2d0*x2 + 1d0)
03666         end if
```

#### 3.20.2.15 pure real(kind(0d0)) function kernel_ts_2::f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3684 of file RKHS.f90.

```
03684         implicit none
03685         real(kind(0d0)), intent(in) :: x
03686         real(kind(0d0)), dimension(:), intent(in) :: par
03687         f21 = x**3 - 3d0/2d0
```

#### 3.20.2.16 pure real(kind(0d0)) function kernel_ts_2::f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3705 of file RKHS.f90.

```
03705         implicit none
03706         real(kind(0d0)), intent(in) :: x
03707         real(kind(0d0)), dimension(:), intent(in) :: par
03708         f22 = x
```

#### 3.20.2.17 pure real(kind(0d0)) function kernel_ts_2::f23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3726 of file RKHS.f90.

```
03726         implicit none
03727         real(kind(0d0)), intent(in) :: x
03728         real(kind(0d0)), dimension(:), intent(in) :: par
03729         f23 = x**2
```

#### 3.20.2.18 pure real(kind(0d0)) function kernel_ts_2::f31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3747 of file RKHS.f90.

```
03747         implicit none
03748         real(kind(0d0)), intent(in) :: x
03749         real(kind(0d0)), dimension(:), intent(in) :: par
03750         f31 = 1d0
```

#### 3.20.2.19 pure real(kind(0d0)) function kernel_ts_2::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3768 of file RKHS.f90.

```
03768         implicit none
03769         real(kind(0d0)), intent(in) :: x
03770         real(kind(0d0)), dimension(:), intent(in) :: par
03771         f32 = x
```

#### 3.20.2.20 pure real(kind(0d0)) function kernel_ts_2::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3789 of file RKHS.f90.

```
03789         implicit none
03790         real(kind(0d0)), intent(in) :: x
03791         real(kind(0d0)), dimension(:), intent(in) :: par
03792         f33 = x
```

#### 3.20.2.21 pure real(kind(0d0)) function kernel_ts_2::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3641 of file RKHS.f90.

```
03641         implicit none
03642         real(kind(0d0)), intent(in) :: x1,x2
03643         real(kind(0d0))             :: xs,xl
03644         real(kind(0d0)), dimension(:), intent(in) :: par
03645         !find larger/smaller of x1 and x2
03646         if(x1 <= x2) then
03647             xs = x1
03648             xl = x2
03649         else
03650             xs = x2
03651             xl = x1
03652         end if
03653         k = 1d0 + xs*xl + 2d0*xs**2*xl - 2d0/3d0*xs**3
```

### 3.20.3 Variable Documentation

#### 3.20.3.1 integer, parameter kernel_ts_2::m2 = 3

Definition at line 3633 of file RKHS.f90.

```
03633     integer, parameter :: m2   = 3
```

#### 3.20.3.2 integer, parameter kernel_ts_2::npar = 0

Definition at line 3634 of file RKHS.f90.

```
03634     integer, parameter :: npar = 0
```

#### 3.20.3.3 real(kind(0d0)), parameter kernel_ts_2::p21 = -2d0/3d0

Definition at line 3635 of file RKHS.f90.

```
03635     real(kind(0d0)), parameter :: p21 = -2d0/3d0,&
03636                                   p22 = 1d0,&
03637                                   p23 = 2d0
```

#### 3.20.3.4 real(kind(0d0)), parameter kernel_ts_2::p22 = 1d0

Definition at line 3635 of file RKHS.f90.

#### 3.20.3.5 real(kind(0d0)), parameter kernel_ts_2::p23 = 2d0

Definition at line 3635 of file RKHS.f90.

## 3.21 kernel_ts_3 Module Reference

Taylor spline kernel with n = 3.

### Functions/Subroutines

- pure real(kind(0d0)) function k (x1, x2, par)
- pure real(kind(0d0)) function dk (x1, x2, par)
- pure real(kind(0d0)) function d2k (x1, x2, par)
- pure real(kind(0d0)) function f21 (x, par)
- pure real(kind(0d0)) function df21 (x, par)
- pure real(kind(0d0)) function d2f21 (x, par)
- pure real(kind(0d0)) function f22 (x, par)
- pure real(kind(0d0)) function df22 (x, par)
- pure real(kind(0d0)) function d2f22 (x, par)
- pure real(kind(0d0)) function f23 (x, par)
- pure real(kind(0d0)) function df23 (x, par)
- pure real(kind(0d0)) function d2f23 (x, par)
- pure real(kind(0d0)) function f24 (x, par)
- pure real(kind(0d0)) function df24 (x, par)
- pure real(kind(0d0)) function d2f24 (x, par)
- pure real(kind(0d0)) function f25 (x, par)
- pure real(kind(0d0)) function df25 (x, par)
- pure real(kind(0d0)) function d2f25 (x, par)
- pure real(kind(0d0)) function f31 (x, par)
- pure real(kind(0d0)) function df31 (x, par)
- pure real(kind(0d0)) function d2f31 (x, par)
- pure real(kind(0d0)) function f32 (x, par)
- pure real(kind(0d0)) function df32 (x, par)
- pure real(kind(0d0)) function d2f32 (x, par)
- pure real(kind(0d0)) function f33 (x, par)
- pure real(kind(0d0)) function df33 (x, par)
- pure real(kind(0d0)) function d2f33 (x, par)
- pure real(kind(0d0)) function f34 (x, par)
- pure real(kind(0d0)) function df34 (x, par)
- pure real(kind(0d0)) function d2f34 (x, par)
- pure real(kind(0d0)) function f35 (x, par)
- pure real(kind(0d0)) function df35 (x, par)
- pure real(kind(0d0)) function d2f35 (x, par)

### Variables

- integer, parameter m2 = 5
- integer, parameter npar = 0
- real(kind(0d0)), parameter p21 = 3d0/10d0
- real(kind(0d0)), parameter p22 = -3d0/2d0
- real(kind(0d0)), parameter p23 = 3d0
- real(kind(0d0)), parameter p24 = 1d0
- real(kind(0d0)), parameter p25 = 1d0

### 3.21.1 Detailed Description

Taylor spline kernel with n = 3.

**Author**

Oliver T. Unke, University of Basel

This module contains all functions necessary to evaluate the Taylor spline kernel with n = 3. Note that this kernel is only defined for values in the interval [0,1]. It is useful for example in describing angular coordinates, as long as a new coordinate is introduced which scales the angular coordinate to the interval [0,1]. An example for such a coordinate would be y = (1d0-cos(alpha))/2d0. Note that by clever choice of y, it is possible to capture the symmetry inherent in the system. The kernel is also applicable to general machine learning problems to interpolate any arbitrary function defined in a finite interval.

### 3.21.2 Function/Subroutine Documentation

#### 3.21.2.1 pure real(kind(0d0)) function kernel_ts_3::d2f21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3901 of file RKHS.f90.

```
03901         implicit none
03902         real(kind(0d0)), intent(in) :: x
03903         real(kind(0d0)), dimension(:), intent(in) :: par
03904         d2f21 = 20d0*x**3
```

#### 3.21.2.2 pure real(kind(0d0)) function kernel_ts_3::d2f22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3922 of file RKHS.f90.

```
03922         implicit none
03923         real(kind(0d0)), intent(in) :: x
03924         real(kind(0d0)), dimension(:), intent(in) :: par
03925         d2f22 = 12d0*x**2
```

#### 3.21.2.3 pure real(kind(0d0)) function kernel_ts_3::d2f23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3943 of file RKHS.f90.

```
03943         implicit none
03944         real(kind(0d0)), intent(in) :: x
03945         real(kind(0d0)), dimension(:), intent(in) :: par
03946         d2f23 = 6d0*x
```

#### 3.21.2.4 pure real(kind(0d0)) function kernel_ts_3::d2f24 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3964 of file RKHS.f90.

```
03964         implicit none
03965         real(kind(0d0)), intent(in) :: x
03966         real(kind(0d0)), dimension(:), intent(in) :: par
03967         d2f24 = 0d0
```

#### 3.21.2.5 pure real(kind(0d0)) function kernel_ts_3::d2f25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3985 of file RKHS.f90.

```
03985         implicit none
03986         real(kind(0d0)), intent(in) :: x
03987         real(kind(0d0)), dimension(:), intent(in) :: par
03988         d2f25 = 2d0
```

#### 3.21.2.6 pure real(kind(0d0)) function kernel_ts_3::d2f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4006 of file RKHS.f90.

```
04006         implicit none
04007         real(kind(0d0)), intent(in) :: x
04008         real(kind(0d0)), dimension(:), intent(in) :: par
04009         d2f31 = 0d0
```

#### 3.21.2.7 pure real(kind(0d0)) function kernel_ts_3::d2f32 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4027 of file RKHS.f90.

```
04027         implicit none
04028         real(kind(0d0)), intent(in) :: x
04029         real(kind(0d0)), dimension(:), intent(in) :: par
04030         d2f32 = 0d0
```

#### 3.21.2.8 pure real(kind(0d0)) function kernel_ts_3::d2f33 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4048 of file RKHS.f90.

```
04048         implicit none
04049         real(kind(0d0)), intent(in) :: x
04050         real(kind(0d0)), dimension(:), intent(in) :: par
04051         d2f33 = 2d0
```

#### 3.21.2.9 pure real(kind(0d0)) function kernel_ts_3::d2f34 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4069 of file RKHS.f90.

```
04069         implicit none
04070         real(kind(0d0)), intent(in) :: x
04071         real(kind(0d0)), dimension(:), intent(in) :: par
04072         d2f34 = 0d0
```

#### 3.21.2.10 pure real(kind(0d0)) function kernel_ts_3::d2f35 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4090 of file RKHS.f90.

```
04090         implicit none
04091         real(kind(0d0)), intent(in) :: x
04092         real(kind(0d0)), dimension(:), intent(in) :: par
04093         d2f35 = 2d0
```

#### 3.21.2.11 pure real(kind(0d0)) function kernel_ts_3::d2k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3874 of file RKHS.f90.

```
03874         implicit none
03875         real(kind(0d0)), intent(in) :: x1,x2
03876         real(kind(0d0)), dimension(:), intent(in) :: par
03877         !find larger/smaller of x1 and x2
03878         if(x1 <= x2) then
03879             d2k = 6d0*x1**3 - 18d0*x1**2*x2 + 18d0*x1*x2**2 + 2d0*x2**2
03880         else
03881             d2k = 2d0*x2**2*(3d0*x2+1d0)
03882         end if
```

#### 3.21.2.12 pure real(kind(0d0)) function kernel_ts_3::df21 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3894 of file RKHS.f90.

```
03894         implicit none
03895         real(kind(0d0)), intent(in) :: x
03896         real(kind(0d0)), dimension(:), intent(in) :: par
03897         df21 = 5d0*x**4
```

#### 3.21.2.13 pure real(kind(0d0)) function kernel_ts_3::df22 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3915 of file RKHS.f90.

```
03915         implicit none
03916         real(kind(0d0)), intent(in) :: x
03917         real(kind(0d0)), dimension(:), intent(in) :: par
03918         df22 = 4d0*x**3
```

#### 3.21.2.14 pure real(kind(0d0)) function kernel_ts_3::df23 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3936 of file RKHS.f90.

```
03936         implicit none
03937         real(kind(0d0)), intent(in) :: x
03938         real(kind(0d0)), dimension(:), intent(in) :: par
03939         df23 = 3d0*x**2
```

#### 3.21.2.15 pure real(kind(0d0)) function kernel_ts_3::df24 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3957 of file RKHS.f90.

```
03957         implicit none
03958         real(kind(0d0)), intent(in) :: x
03959         real(kind(0d0)), dimension(:), intent(in) :: par
03960         df24 = 1d0
```

#### 3.21.2.16 pure real(kind(0d0)) function kernel_ts_3::df25 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3978 of file RKHS.f90.

```
03978         implicit none
03979         real(kind(0d0)), intent(in) :: x
03980         real(kind(0d0)), dimension(:), intent(in) :: par
03981         df25 = 2d0*x
```

#### 3.21.2.17 pure real(kind(0d0)) function kernel_ts_3::df31 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3999 of file RKHS.f90.

```
03999         implicit none
04000         real(kind(0d0)), intent(in) :: x
04001         real(kind(0d0)), dimension(:), intent(in) :: par
04002         df31 = 0d0
```

#### 3.21.2.18 pure real(kind(0d0)) function kernel_ts_3::df32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4020 of file RKHS.f90.

```
04020         implicit none
04021         real(kind(0d0)), intent(in) :: x
04022         real(kind(0d0)), dimension(:), intent(in) :: par
04023         df32 = 1d0
```

#### 3.21.2.19 pure real(kind(0d0)) function kernel_ts_3::df33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4041 of file RKHS.f90.

```
04041         implicit none
04042         real(kind(0d0)), intent(in) :: x
04043         real(kind(0d0)), dimension(:), intent(in) :: par
04044         df33 = 2d0*x
```

#### 3.21.2.20 pure real(kind(0d0)) function kernel_ts_3::df34 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4062 of file RKHS.f90.

```
04062         implicit none
04063         real(kind(0d0)), intent(in) :: x
04064         real(kind(0d0)), dimension(:), intent(in) :: par
04065         df34 = 1d0
```

#### 3.21.2.21 pure real(kind(0d0)) function kernel_ts_3::df35 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4083 of file RKHS.f90.

```
04083         implicit none
04084         real(kind(0d0)), intent(in) :: x
04085         real(kind(0d0)), dimension(:), intent(in) :: par
04086         df35 = 2d0*x
```

#### 3.21.2.22 pure real(kind(0d0)) function kernel_ts_3::dk ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3859 of file RKHS.f90.

```
03859         implicit none
03860         real(kind(0d0)), intent(in) :: x1,x2
03861         real(kind(0d0)), dimension(:), intent(in) :: par
03862         !find larger/smaller of x1 and x2
03863         if(x1 <= x2) then
03864             dk = 2d0*x1*x2**2 + x2 + 9d0 * (x1*x2)**2 &
03865                  - 6d0*x1**3*x2 + 3d0/2d0 * x1**4
03866         else
03867             dk = 0.5d0*x2 * (12d0*x1*x2**2 - 3d0*x2**3 &
03868                              + 4d0*x1*x2 + 2d0)
03869         end if
```

#### 3.21.2.23 pure real(kind(0d0)) function kernel_ts_3::f21 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3887 of file RKHS.f90.

```
03887         implicit none
03888         real(kind(0d0)), intent(in) :: x
03889         real(kind(0d0)), dimension(:), intent(in) :: par
03890         f21 = x**5 + 10d0/3d0
```

#### 3.21.2.24 pure real(kind(0d0)) function kernel_ts_3::f22 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3908 of file RKHS.f90.

```
03908         implicit none
03909         real(kind(0d0)), intent(in) :: x
03910         real(kind(0d0)), dimension(:), intent(in) :: par
03911         f22 = x**4
```

#### 3.21.2.25 pure real(kind(0d0)) function kernel_ts_3::f23 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3929 of file RKHS.f90.

```
03929         implicit none
03930         real(kind(0d0)), intent(in) :: x
03931         real(kind(0d0)), dimension(:), intent(in) :: par
03932         f23 = x**3
```

#### 3.21.2.26 pure real(kind(0d0)) function kernel_ts_3::f24 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3950 of file RKHS.f90.

```
03950         implicit none
03951         real(kind(0d0)), intent(in) :: x
03952         real(kind(0d0)), dimension(:), intent(in) :: par
03953         f24 = x
```

#### 3.21.2.27 pure real(kind(0d0)) function kernel_ts_3::f25 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3971 of file RKHS.f90.

```
03971         implicit none
03972         real(kind(0d0)), intent(in) :: x
03973         real(kind(0d0)), dimension(:), intent(in) :: par
03974         f25 = x**2
```

#### 3.21.2.28 pure real(kind(0d0)) function kernel_ts_3::f31 ( real(kind(0d0)), intent(in) *x*, real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3992 of file RKHS.f90.

```
03992         implicit none
03993         real(kind(0d0)), intent(in) :: x
03994         real(kind(0d0)), dimension(:), intent(in) :: par
03995         f31 = 1d0
```

#### 3.21.2.29 pure real(kind(0d0)) function kernel_ts_3::f32 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4013 of file RKHS.f90.

```
04013         implicit none
04014         real(kind(0d0)), intent(in) :: x
04015         real(kind(0d0)), dimension(:), intent(in) :: par
04016         f32 = x
```

#### 3.21.2.30 pure real(kind(0d0)) function kernel_ts_3::f33 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4034 of file RKHS.f90.

```
04034         implicit none
04035         real(kind(0d0)), intent(in) :: x
04036         real(kind(0d0)), dimension(:), intent(in) :: par
04037         f33 = x**2
```

#### 3.21.2.31 pure real(kind(0d0)) function kernel_ts_3::f34 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4055 of file RKHS.f90.

```
04055         implicit none
04056         real(kind(0d0)), intent(in) :: x
04057         real(kind(0d0)), dimension(:), intent(in) :: par
04058         f34 = x
```

#### 3.21.2.32 pure real(kind(0d0)) function kernel_ts_3::f35 ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4076 of file RKHS.f90.

```
04076         implicit none
04077         real(kind(0d0)), intent(in) :: x
04078         real(kind(0d0)), dimension(:), intent(in) :: par
04079         f35 = x**2
```

#### 3.21.2.33 pure real(kind(0d0)) function kernel_ts_3::k ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3841 of file RKHS.f90.

```
03841         implicit none
03842         real(kind(0d0)), intent(in) :: x1,x2
03843         real(kind(0d0))             :: xs,xl
03844         real(kind(0d0)), dimension(:), intent(in) :: par
03845         !find larger/smaller of x1 and x2
03846         if(x1 <= x2) then
03847             xs = x1
03848             xl = x2
03849         else
03850             xs = x2
03851             xl = x1
03852         end if
03853         k = 3d0/10d0*xs**5 - 3d0/2d0*xl*xs**4 + 3d0*xl**2*xs**3 &
03854             + xs*xl + (xs*xl)**2 + 1d0
```

### 3.21.3 Variable Documentation

#### 3.21.3.1 integer, parameter kernel_ts_3::m2 = 5

Definition at line 3831 of file RKHS.f90.

```
03831     integer, parameter :: m2   = 5
```

#### 3.21.3.2 integer, parameter kernel_ts_3::npar = 0

Definition at line 3832 of file RKHS.f90.

```
03832     integer, parameter :: npar = 0
```

#### 3.21.3.3 real(kind(0d0)), parameter kernel_ts_3::p21 = 3d0/10d0

Definition at line 3833 of file RKHS.f90.

```
03833     real(kind(0d0)), parameter :: p21 = 3d0/10d0,&
03834                                   p22 = -3d0/2d0,&
03835                                   p23 = 3d0,&
03836                                   p24 = 1d0,&
03837                                   p25 = 1d0
```

#### 3.21.3.4 real(kind(0d0)), parameter kernel_ts_3::p22 = -3d0/2d0

Definition at line 3833 of file RKHS.f90.

#### 3.21.3.5 real(kind(0d0)), parameter kernel_ts_3::p23 = 3d0

Definition at line 3833 of file RKHS.f90.

#### 3.21.3.6 real(kind(0d0)), parameter kernel_ts_3::p24 = 1d0

Definition at line 3833 of file RKHS.f90.

#### 3.21.3.7 real(kind(0d0)), parameter kernel_ts_3::p25 = 1d0

Definition at line 3833 of file RKHS.f90.

## 3.22 linearalgebra Module Reference

Some routines needed for solving matrix equations.

### Functions/Subroutines

- subroutine throw_error (proc, errormsg)

  *Throws error messages if problems are encountered.*
- subroutine, public forward_substitution (L, y, b)

  *Solves L∗y = b for y by forward substitution.*
- subroutine, public backward_substitution (U, x, y)

  *Solves U∗x = y for x by backward substitution.*
- subroutine, public cholesky_decomposition (M)

  *does the Cholesky decomposition of a positive-definite symmetric square matrix M*
- subroutine, public cholesky_solve (L, x, b)

  *solves the matrix equation M∗x = b for x*
- subroutine, public cholesky_inverse (L, invL)

  *computes the inverse of a matrix*

### 3.22.1 Detailed Description

Some routines needed for solving matrix equations.

**Author**

Oliver T. Unke, University of Basel

This module contains all methods which are necessary to solve a matrix equation by Cholesky decomposition. Namely, it contains a method for performing the decomposition and a method to solve an equation using forward and backward substitution.

### 3.22.2 Function/Subroutine Documentation

#### 3.22.2.1 subroutine public linearalgebra::backward_substitution ( real(kind(0d0)), dimension(:,:), intent(in) *U,* real(kind(0d0)), dimension(:), intent(out) *x,* real(kind(0d0)), dimension(:), intent(in) *y* )

Solves U∗x = y for x by backward substitution.

**Author**

Oliver T. Unke, University of Basel

Here, U is an upper triangular matrix.

Definition at line 4180 of file RKHS.f90.

```
04180     implicit none
04181     real(kind(0d0)), dimension(:,:), intent(in)  :: u
04182     real(kind(0d0)), dimension(:),   intent(out) :: x
04183     real(kind(0d0)), dimension(:),   intent(in)  :: y
04184     real(kind(0d0)) :: s
04185     integer :: i,k,n
04186
04187     n = size(u,dim=1)
04188     if(n /= size(x,dim=1) .or. n /= size(y,dim=1)) then
04189         call throw_error("forward_substitution","Matrix U is not the appropriate size for x and/or y!")
04190     end if
04191
04192     !back substitution
04193     x(n) = y(n)/u(n,n)
04194     do i = n-1,1,-1
04195         s = 0d0
04196         do k = i+1,n,1
04197             s = s + u(i,k)*x(k)
04198         end do
04199         x(i) = (y(i)-s)/u(i,i)
04200     end do
04201     return
```

#### 3.22.2.2 subroutine public linearalgebra::cholesky_decomposition ( real(kind(0d0)), dimension(:,:), intent(out) *M* )

does the Cholesky decomposition of a positive-definite symmetric square matrix M

**Author**

Oliver T. Unke, University of Basel

Note: Only the lower triagonal part of M is needed as input and only that part of the matrix is modified.

Definition at line 4216 of file RKHS.f90.

```
04216     implicit none
04217     real(kind(0d0)), dimension(:,:), intent(out) :: m
04218     integer :: i
04219
04220     if(size(m,dim=1) /= size(m,dim=2)) then
04221         call throw_error("cholesky_decomposition","Matrix M is not a square matrix!")
04222     end if
04223
04224     !do the Cholesky decomposition
04225     do i = 1,size(m,dim=1)
04226         !diagonal component
04227         m(i,i) = m(i,i) - dot_product(m(i,1:i-1),m(i,1:i-1))
```

```
04228         if(m(i,i) > 0d0) then
04229             m(i,i) = sqrt(m(i,i))
04230         else
04231             call throw_error("cholesky_decomposition","Matrix M is not positive definite! "//&
04232                              "If you are sure M is correct, maybe the matrix is ill conditioned. "//&
04233                              "Try to use the Tikhonov regularization procedure.")
04234         end if
04235
04236         !off-diagonal component
04237         if(i < size(m,dim=1)) m(i+1:size(m,dim=1),i) = (m(i+1:size(m,dim=1),i) - &
04238             matmul(m(i+1:size(m,dim=1),1:i-1),m(i,1:i-1))) / m(i,i)
04239     end do
04240     return
```

#### 3.22.2.3 subroutine public linearalgebra::cholesky_inverse ( real(kind(0d0)), dimension(:,:), intent(in) *L*, real(kind(0d0)), dimension(:,:), intent(out) *invL* )

computes the inverse of a matrix

**Author**

Oliver T. Unke, University of Basel

The input matrix needs to be the lower tridiagonal matrix, which is obtained by Cholesky decomposition.

Definition at line 4285 of file RKHS.f90.

```
04285     implicit none
04286     real(kind(0d0)), dimension(:,:), intent(in)  :: l
04287     real(kind(0d0)), dimension(:,:), intent(out) :: invl
04288     real(kind(0d0)), dimension(size(L,dim=1))    :: b
04289     integer :: i,n
04290     n = size(l,dim=1)
04291     if(n /= size(invl,dim=1) .or. n /= size(invl,dim=2) .or. n /= size(l,dim=2)) then
04292         call throw_error("cholesky_inverse","Matrix L/invL is not the appropriate size!")
04293     end if
04294     do i = 1,n
04295         b    = 0d0
04296         b(i) = 1d0
04297         call cholesky_solve(l,invl(i,:),b)
04298     end do
04299     return
```

#### 3.22.2.4 subroutine public linearalgebra::cholesky_solve ( real(kind(0d0)), dimension(:,:), intent(in) *L*, real(kind(0d0)), dimension(:), intent(out) *x*, real(kind(0d0)), dimension(:), intent(in) *b* )

solves the matrix equation M∗x = b for x

**Author**

Oliver T. Unke, University of Basel

The input matrix needs to be the lower tridiagonal matrix, which is obtained by Cholesky decomposition. The solution is the obtained by forward and backward substitution.

Definition at line 4256 of file RKHS.f90.

```
04256     implicit none
04257     real(kind(0d0)), dimension(:,:), intent(in)  :: l
04258     real(kind(0d0)), dimension(:),   intent(out) :: x
04259     real(kind(0d0)), dimension(:),   intent(in)  :: b
04260     real(kind(0d0)), dimension(size(x)) :: y
04261     integer :: n
04262
04263     n = size(l,dim=1)
04264     if(n /= size(x,dim=1) .or. n /= size(b,dim=1)) then
04265         call throw_error("cholesky_solve","Matrix L is not the appropriate size for x and/or b!")
04266     end if
04267
04268     call forward_substitution(l,y,b)
04269     call backward_substitution(transpose(l),x,y)
04270     return
```

#### 3.22.2.5 subroutine public linearalgebra::forward_substitution ( real(kind(0d0)), dimension(:,:), intent(in) *L,* real(kind(0d0)), dimension(:), intent(out) *y,* real(kind(0d0)), dimension(:), intent(in) *b* )

Solves L∗y = b for y by forward substitution.

**Author**

Oliver T. Unke, University of Basel

Here, L is a lower triangular matrix.

Definition at line 4145 of file RKHS.f90.

```
04145     implicit none
04146     real(kind(0d0)), dimension(:,:), intent(in)  :: l
04147     real(kind(0d0)), dimension(:),   intent(out) :: y
04148     real(kind(0d0)), dimension(:),   intent(in)  :: b
04149     real(kind(0d0)) :: s
04150     integer :: i,k,n
04151
04152     n = size(l,dim=1)
04153     if(n /= size(y,dim=1) .or. n /= size(b,dim=1)) then
04154         call throw_error("forward_substitution","Matrix L is not the appropriate size for y and/or b!")
04155     end if
04156
04157     !forward substitution
04158     y(1) = b(1)/l(1,1)
04159     do i = 2,n,1
04160         s = 0d0
04161         do k = 1,i-1,1
04162             s = s + l(i,k)*y(k)
04163         end do
04164         y(i) = (b(i)-s)/l(i,i)
04165     end do
04166     return
```

#### 3.22.2.6 subroutine linearalgebra::throw_error ( character(len=∗), intent(in) *proc,* character(len=∗), intent(in) *errormsg* ) [private]

Throws error messages if problems are encountered.

**Author**

Oliver T. Unke, University of Basel

Definition at line 4128 of file RKHS.f90.

```
04128     implicit none
04129     character(len=*), intent(in) :: proc, errormsg
04130     write(*,*) "ERROR in module LinearAlgebra: "//proc//": "//errormsg
04131     stop
```

## 3.23 reproducing_kernels Module Reference

Contains all the definitions for the 1-dimensional kernel functions.

**Data Types**

- type f_ptr

  *for wrapping function pointers, such that it is possible to have arrays of function pointers*
- interface func

  *interface for the f2k/f3k functions that are stored in function pointer arrays*
- interface kdirect

  *interface for naive implementation of kernels (instead of using the fast evaluation)*
- type kernel_1d

  *defines 1-dimensional kernels*

### Functions/Subroutines

- subroutine debug_kernel (kernel, x1, x2, fval_slow, dval_slow, d2val_slow, fval_fast, dval_fast, d2val_fast)

  *test kernel implementation (fast vs normal) -> dval_slow and dval_fast, d2val_slow and d2val_fast as well as fval_fast and fval_slow should be equal after calling this subroutine.*
- subroutine init_kernel (kernel, kernel_type)

  *needs to be called to initialize the kernel and set it to a certain type*
- subroutine init_rp_2_6 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 6*
- subroutine init_rp_3_6 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 6*
- subroutine init_rp_2_5 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 5*
- subroutine init_rp_3_5 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 5*
- subroutine init_rp_2_4 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 4*
- subroutine init_rp_3_4 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 4*
- subroutine init_rp_2_3 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 3*
- subroutine init_rp_3_3 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 3*
- subroutine init_rp_2_2 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 2*
- subroutine init_rp_3_2 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 2*
- subroutine init_rp_2_1 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 1*
- subroutine init_rp_3_1 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 1*
- subroutine init_rp_2_0 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 2 and m = 0*
- subroutine init_rp_3_0 (kernel)

  *initializes the kernel to reciprocal decay kernel with n = 3 and m = 0*
- subroutine init_exp_2 (kernel)

  *initializes the kernel to exponential decay kernel with n = 2*
- subroutine init_exp_3 (kernel)

  *initializes the kernel to exponential decay kernel with n = 3*
- subroutine init_ts_2 (kernel)

  *initializes the kernel to Taylor spline kernel with n = 2*
- subroutine init_ts_3 (kernel)

  *initializes the kernel to Taylor spline kernel with n = 3*
- subroutine init_laplacian (kernel)

  *initializes the kernel to Laplacian kernel*
- subroutine init_bernoulli_2 (kernel)

  *initializes the kernel to Bernoulli kernel with n = 2*
- subroutine debug_kernel (kernel, x1, x2, fval_slow, dval_slow, fval_fast, dval_fast)

  *test kernel implementation (fast vs normal) -> dval_slow and dval_fast, as well as fval_fast and fval_slow should be equal after calling this subroutine.*

**Variables**

- integer, parameter reciprocal_power_n2_m6_kernel = 0

  *enumerator for kernel types*
- integer, parameter reciprocal_power_n3_m6_kernel = 1
- integer, parameter reciprocal_power_n2_m5_kernel = 2
- integer, parameter reciprocal_power_n3_m5_kernel = 3
- integer, parameter reciprocal_power_n2_m4_kernel = 4
- integer, parameter reciprocal_power_n3_m4_kernel = 5
- integer, parameter reciprocal_power_n2_m3_kernel = 6
- integer, parameter reciprocal_power_n3_m3_kernel = 7
- integer, parameter reciprocal_power_n2_m2_kernel = 8
- integer, parameter reciprocal_power_n3_m2_kernel = 9
- integer, parameter reciprocal_power_n2_m1_kernel = 10
- integer, parameter reciprocal_power_n3_m1_kernel = 11
- integer, parameter reciprocal_power_n2_m0_kernel = 12
- integer, parameter reciprocal_power_n3_m0_kernel = 13
- integer, parameter exponential_decay_n2_kernel = 14
- integer, parameter exponential_decay_n3_kernel = 15
- integer, parameter taylor_spline_n2_kernel = 16
- integer, parameter taylor_spline_n3_kernel = 17
- integer, parameter laplacian_kernel = 18
- integer, parameter bernoulli_n2_kernel = 19

### 3.23.1 Detailed Description

Contains all the definitions for the 1-dimensional kernel functions.

**Author**

Oliver T. Unke, University of Basel

Note that for the naive implementation of the derivative of the kernel function, when calling dk(x1,x2), it is always assumed that x1 is the variable with respect to which the kernel function should be derived. It is extremely easy to add new 1-dimensional kernels simply by adding them in this module.

### 3.23.2 Function/Subroutine Documentation

#### 3.23.2.1 subroutine reproducing_kernels::debug_kernel ( type(kernel_1d), intent(in) *kernel,* real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), intent(out) *fval_slow,* real(kind(0d0)), intent(out) *dval_slow,* real(kind(0d0)), intent(out) *fval_fast,* real(kind(0d0)), intent(out) *dval_fast* )

test kernel implementation (fast vs normal) -> dval_slow and dval_fast, as well as fval_fast and fval_slow should be equal after calling this subroutine.

If not, there must be some bug in the kernel implementation!

Definition at line 3235 of file RKHS_preHessian_backup.f90.

```
03235         implicit none
03236         type(kernel_1d), intent(in)  :: kernel
03237         real(kind(0d0)), intent(in)  :: x1,x2     !input variables
03238         real(kind(0d0)), intent(out) :: fval_slow,dval_slow,fval_fast,dval_fast !function value and
      derivative value of
03239                                                                                 !slow and fast
      implementation
03240         real(kind(0d0)) :: xs,xl
03241         integer :: i
03242
03243         !determine larger and smaller value (needed for fast implementation)
03244         if(x1 <= x2) then
```

```
03245             xs = x1
03246             xl = x2
03247         else
03248             xs = x2
03249             xl = x1
03250         end if
03251
03252         fval_slow = kernel% k(x1,x2,kernel%par)
03253         dval_slow = kernel%dk(x1,x2,kernel%par)
03254         fval_fast = 0d0
03255         do i = 1,kernel%M2
03256             fval_fast = fval_fast + kernel%p2(i) *  kernel% f2(i)%f(xs,kernel%par)*kernel% f3(i)%f(xl,
      kernel%par)
03257         end do
03258         !depending on which value x1 is, we need to take the derivative of a different function
03259         dval_fast = 0d0
03260         if(x1 <= x2) then
03261             do i = 1,kernel%M2
03262                 dval_fast = dval_fast + kernel%p2(i) *  kernel%df2(i)%f(xs,kernel%par)*kernel% f3(i)%f(xl,
      kernel%par)
03263             end do
03264         else
03265             do i = 1,kernel%M2
03266                 dval_fast = dval_fast + kernel%p2(i) *  kernel% f2(i)%f(xs,kernel%par)*kernel%df3(i)%f(xl,
      kernel%par)
03267             end do
03268         end if
03269
03270         return
```

#### 3.23.2.2 subroutine reproducing_kernels::debug_kernel ( type(kernel_1d), intent(in) *kernel,* real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), intent(out) *fval_slow,* real(kind(0d0)), intent(out) *dval_slow,* real(kind(0d0)), intent(out) *d2val_slow,* real(kind(0d0)), intent(out) *fval_fast,* real(kind(0d0)), intent(out) *dval_fast,* real(kind(0d0)), intent(out) *d2val_fast* )

test kernel implementation (fast vs normal) -> dval_slow and dval_fast, d2val_slow and d2val_fast as well as fval↩ _fast and fval_slow should be equal after calling this subroutine.

If not, there must be some bug in the kernel implementation!

Definition at line 4406 of file RKHS.f90.

```
04406         implicit none
04407         type(kernel_1d), intent(in)  :: kernel
04408         real(kind(0d0)), intent(in)  :: x1,x2     !input variables
04409         real(kind(0d0)), intent(out) :: fval_slow,dval_slow,d2val_slow,&
04410                                         fval_fast,dval_fast,d2val_fast    !function value and derivative
      values of
04411                                                                           !slow and fast implementation
04412         real(kind(0d0)) :: xs,xl
04413         integer :: i
04414
04415         !determine larger and smaller value (needed for fast implementation)
04416         if(x1 <= x2) then
04417             xs = x1
04418             xl = x2
04419         else
04420             xs = x2
04421             xl = x1
04422         end if
04423
04424         fval_slow  = kernel%  k(x1,x2,kernel%par)
04425         dval_slow  = kernel% dk(x1,x2,kernel%par)
04426         d2val_slow = kernel%d2k(x1,x2,kernel%par)
04427         fval_fast = 0d0
04428         do i = 1,kernel%M2
04429             fval_fast = fval_fast + kernel%p2(i) *  kernel% f2(i)%f(xs,kernel%par)*kernel% f3(i)%f(xl,
      kernel%par)
04430         end do
04431         !depending on which value x1 is, we need to take the derivative of a different function
04432         dval_fast  = 0d0
04433         d2val_fast = 0d0
04434         if(x1 <= x2) then
04435             do i = 1,kernel%M2
04436                 dval_fast  =  dval_fast + kernel%p2(i) *  kernel% df2(i)%f(xs,kernel%par)*kernel% f3(i)%f(
      xl,kernel%par)
04437                 d2val_fast = d2val_fast + kernel%p2(i) *  kernel%d2f2(i)%f(xs,kernel%par)*kernel% f3(i)%f(
      xl,kernel%par)
04438             end do
04439         else
04440             do i = 1,kernel%M2
```

```
04441                 dval_fast  =  dval_fast + kernel%p2(i) *  kernel% f2(i)%f(xs,kernel%par)*kernel% df3(i)%f(
      xl,kernel%par)
04442                 d2val_fast = d2val_fast + kernel%p2(i) *  kernel% f2(i)%f(xs,kernel%par)*kernel%d2f3(i)%f(
      xl,kernel%par)
04443             end do
04444         end if
04445
04446         return
```

#### 3.23.2.3 subroutine reproducing_kernels::init_bernoulli_2 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to Bernoulli kernel with n = 2

Definition at line 5576 of file RKHS.f90.

```
05576         use kernel_bernoulli_2
05577         implicit none
05578         type(kernel_1d), intent(out)  :: kernel
05579         kernel%kernel_type = bernoulli_n2_kernel
05580         kernel%M2   =  m2  !M2
05581         kernel%Npar =  npar!Npar
05582         kernel%k    => k   !k
05583         kernel%dk   => dk  !dk
05584         kernel%d2k  => d2k !d2k
05585
05586         !allocate memory
05587         allocate(kernel%p2  (m2),&
05588                  kernel%f2  (m2),&
05589                  kernel%f3  (m2),&
05590                  kernel%df2 (m2),&
05591                  kernel%df3 (m2),&
05592                  kernel%d2f2(m2),&
05593                  kernel%d2f3(m2))
05594
05595         !set p2 coefficients
05596         kernel%p2(1)  = p21
05597         kernel%p2(2)  = p22
05598         kernel%p2(3)  = p23
05599         kernel%p2(4)  = p24
05600         kernel%p2(5)  = p25
05601         kernel%p2(6)  = p26
05602         kernel%p2(7)  = p27
05603         kernel%p2(8)  = p28
05604         kernel%p2(9)  = p29
05605         kernel%p2(10) = p210
05606         kernel%p2(11) = p211
05607         kernel%p2(12) = p212
05608
05609         !set f2 functions
05610         kernel%f2(1)%f  => f21
05611         kernel%f2(2)%f  => f22
05612         kernel%f2(3)%f  => f23
05613         kernel%f2(4)%f  => f24
05614         kernel%f2(5)%f  => f25
05615         kernel%f2(6)%f  => f26
05616         kernel%f2(7)%f  => f27
05617         kernel%f2(8)%f  => f28
05618         kernel%f2(9)%f  => f29
05619         kernel%f2(10)%f => f210
05620         kernel%f2(11)%f => f211
05621         kernel%f2(12)%f => f212
05622
05623         !set f3 functions
05624         kernel%f3(1)%f  => f31
05625         kernel%f3(2)%f  => f32
05626         kernel%f3(3)%f  => f33
05627         kernel%f3(4)%f  => f34
05628         kernel%f3(5)%f  => f35
05629         kernel%f3(6)%f  => f36
05630         kernel%f3(7)%f  => f37
05631         kernel%f3(8)%f  => f38
05632         kernel%f3(9)%f  => f39
05633         kernel%f3(10)%f => f310
05634         kernel%f3(11)%f => f311
05635         kernel%f3(12)%f => f312
05636
05637         !set df2 functions
05638         kernel%df2(1)%f  => df21
05639         kernel%df2(2)%f  => df22
05640         kernel%df2(3)%f  => df23
05641         kernel%df2(4)%f  => df24
05642         kernel%df2(5)%f  => df25
05643         kernel%df2(6)%f  => df26
05644         kernel%df2(7)%f  => df27
```

```
05645         kernel%df2(8)%f  => df28
05646         kernel%df2(9)%f  => df29
05647         kernel%df2(10)%f => df210
05648         kernel%df2(11)%f => df211
05649         kernel%df2(12)%f => df212
05650
05651         !set df3 functions
05652         kernel%df3(1)%f  => df31
05653         kernel%df3(2)%f  => df32
05654         kernel%df3(3)%f  => df33
05655         kernel%df3(4)%f  => df34
05656         kernel%df3(5)%f  => df35
05657         kernel%df3(6)%f  => df36
05658         kernel%df3(7)%f  => df37
05659         kernel%df3(8)%f  => df38
05660         kernel%df3(9)%f  => df39
05661         kernel%df3(10)%f => df310
05662         kernel%df3(11)%f => df311
05663         kernel%df3(12)%f => df312
05664
05665         !set d2f2 functions
05666         kernel%d2f2(1)%f  => d2f21
05667         kernel%d2f2(2)%f  => d2f22
05668         kernel%d2f2(3)%f  => d2f23
05669         kernel%d2f2(4)%f  => d2f24
05670         kernel%d2f2(5)%f  => d2f25
05671         kernel%d2f2(6)%f  => d2f26
05672         kernel%d2f2(7)%f  => d2f27
05673         kernel%d2f2(8)%f  => d2f28
05674         kernel%d2f2(9)%f  => d2f29
05675         kernel%d2f2(10)%f => d2f210
05676         kernel%d2f2(11)%f => d2f211
05677         kernel%d2f2(12)%f => d2f212
05678
05679         !set d2f3 functions
05680         kernel%d2f3(1)%f  => d2f31
05681         kernel%d2f3(2)%f  => d2f32
05682         kernel%d2f3(3)%f  => d2f33
05683         kernel%d2f3(4)%f  => d2f34
05684         kernel%d2f3(5)%f  => d2f35
05685         kernel%d2f3(6)%f  => d2f36
05686         kernel%d2f3(7)%f  => d2f37
05687         kernel%d2f3(8)%f  => d2f38
05688         kernel%d2f3(9)%f  => d2f39
05689         kernel%d2f3(10)%f => d2f310
05690         kernel%d2f3(11)%f => d2f311
05691         kernel%d2f3(12)%f => d2f312
05692         return
```

#### 3.23.2.4 subroutine reproducing_kernels::init_exp_2 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to exponential decay kernel with n = 2

Definition at line 5270 of file RKHS.f90.

```
05270         use kernel_exp_2
05271         implicit none
05272         type(kernel_1d), intent(out)  :: kernel
05273         kernel%kernel_type = exponential_decay_n2_kernel
05274         kernel%M2   = m2   !M2
05275         kernel%Npar = npar !Npar
05276         if(allocated(kernel%par)) deallocate(kernel%par)
05277         allocate(kernel%par(npar))
05278         kernel%par = 1d0
05279         kernel%k    => k   !k
05280         kernel%dk   => dk  !dk
05281         kernel%d2k  => d2k !d2k
05282
05283         !allocate memory
05284         allocate(kernel%p2  (m2),&
05285                  kernel%f2  (m2),&
05286                  kernel%f3  (m2),&
05287                  kernel%df2 (m2),&
05288                  kernel%df3 (m2),&
05289                  kernel%d2f2(m2),&
05290                  kernel%d2f3(m2))
05291
05292         !set p2 coefficients
05293         kernel%p2(1) = p21
05294         kernel%p2(2) = p22
05295
05296         !set f2 functions
05297         kernel%f2(1)%f => f21
05298         kernel%f2(2)%f => f22
```

```
05299
05300         !set f3 functions
05301         kernel%f3(1)%f => f31
05302         kernel%f3(2)%f => f32
05303
05304         !set df2 functions
05305         kernel%df2(1)%f => df21
05306         kernel%df2(2)%f => df22
05307
05308         !set df3 functions
05309         kernel%df3(1)%f => df31
05310         kernel%df3(2)%f => df32
05311
05312         !set d2f2 functions
05313         kernel%d2f2(1)%f => d2f21
05314         kernel%d2f2(2)%f => d2f22
05315
05316         !set d2f3 functions
05317         kernel%d2f3(1)%f => d2f31
05318         kernel%d2f3(2)%f => d2f32
05319         return
```

#### 3.23.2.5 subroutine reproducing_kernels::init_exp_3 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to exponential decay kernel with n = 3

Definition at line 5324 of file RKHS.f90.

```
05324         use kernel_exp_3
05325         implicit none
05326         type(kernel_1d), intent(out)  :: kernel
05327         kernel%kernel_type = exponential_decay_n3_kernel
05328         kernel%M2   = m2   !M2
05329         kernel%Npar = npar !Npar
05330         if(allocated(kernel%par)) deallocate(kernel%par)
05331         allocate(kernel%par(npar))
05332         kernel%par = 1d0
05333         kernel%k    => k   !k
05334         kernel%dk   => dk  !dk
05335         kernel%d2k  => d2k !d2k
05336
05337         !allocate memory
05338         allocate(kernel%p2  (m2),&
05339                  kernel%f2  (m2),&
05340                  kernel%f3  (m2),&
05341                  kernel%df2 (m2),&
05342                  kernel%df3 (m2),&
05343                  kernel%d2f2(m2),&
05344                  kernel%d2f3(m2))
05345
05346         !set p2 coefficients
05347         kernel%p2(1) = p21
05348         kernel%p2(2) = p22
05349         kernel%p2(3) = p23
05350         kernel%p2(4) = p24
05351         kernel%p2(5) = p25
05352
05353         !set f2 functions
05354         kernel%f2(1)%f => f21
05355         kernel%f2(2)%f => f22
05356         kernel%f2(3)%f => f23
05357         kernel%f2(4)%f => f24
05358         kernel%f2(5)%f => f25
05359
05360         !set f3 functions
05361         kernel%f3(1)%f => f31
05362         kernel%f3(2)%f => f32
05363         kernel%f3(3)%f => f33
05364         kernel%f3(4)%f => f34
05365         kernel%f3(5)%f => f35
05366
05367         !set df2 functions
05368         kernel%df2(1)%f => df21
05369         kernel%df2(2)%f => df22
05370         kernel%df2(3)%f => df23
05371         kernel%df2(4)%f => df24
05372         kernel%df2(5)%f => df25
05373
05374         !set df3 functions
05375         kernel%df3(1)%f => df31
05376         kernel%df3(2)%f => df32
05377         kernel%df3(3)%f => df33
05378         kernel%df3(4)%f => df34
05379         kernel%df3(5)%f => df35
```

```
05380
05381         !set d2f2 functions
05382         kernel%d2f2(1)%f => d2f21
05383         kernel%d2f2(2)%f => d2f22
05384         kernel%d2f2(3)%f => d2f23
05385         kernel%d2f2(4)%f => d2f24
05386         kernel%d2f2(5)%f => d2f25
05387
05388         !set d2f3 functions
05389         kernel%d2f3(1)%f => d2f31
05390         kernel%d2f3(2)%f => d2f32
05391         kernel%d2f3(3)%f => d2f33
05392         kernel%d2f3(4)%f => d2f34
05393         kernel%d2f3(5)%f => d2f35
05394         return
```

#### 3.23.2.6 subroutine reproducing_kernels::init_kernel ( class(kernel_1d) *kernel,* integer, intent(in) *kernel_type* )

needs to be called to initialize the kernel and set it to a certain type

Definition at line 4451 of file RKHS.f90.

```
04451         use kernel_rp_2_5
04452         implicit none
04453         class(kernel_1d) :: kernel
04454         integer, intent(in) :: kernel_type
04455
04456         select case (kernel_type)
04457             case (reciprocal_power_n2_m6_kernel)
04458                 call init_rp_2_6(kernel)
04459             case (reciprocal_power_n3_m6_kernel)
04460                 call init_rp_3_6(kernel)
04461             case (reciprocal_power_n2_m5_kernel)
04462                 call init_rp_2_5(kernel)
04463             case (reciprocal_power_n3_m5_kernel)
04464                 call init_rp_3_5(kernel)
04465             case (reciprocal_power_n2_m4_kernel)
04466                 call init_rp_2_4(kernel)
04467             case (reciprocal_power_n3_m4_kernel)
04468                 call init_rp_3_4(kernel)
04469             case (reciprocal_power_n2_m3_kernel)
04470                 call init_rp_2_3(kernel)
04471             case (reciprocal_power_n3_m3_kernel)
04472                 call init_rp_3_3(kernel)
04473             case (reciprocal_power_n2_m2_kernel)
04474                 call init_rp_2_2(kernel)
04475             case (reciprocal_power_n3_m2_kernel)
04476                 call init_rp_3_2(kernel)
04477             case (reciprocal_power_n2_m1_kernel)
04478                 call init_rp_2_1(kernel)
04479             case (reciprocal_power_n3_m1_kernel)
04480                 call init_rp_3_1(kernel)
04481             case (reciprocal_power_n2_m0_kernel)
04482                 call init_rp_2_0(kernel)
04483             case (reciprocal_power_n3_m0_kernel)
04484                 call init_rp_3_0(kernel)
04485             case (exponential_decay_n2_kernel)
04486                 call init_exp_2(kernel)
04487             case (exponential_decay_n3_kernel)
04488                 call init_exp_3(kernel)
04489             case (taylor_spline_n2_kernel)
04490                 call init_ts_2(kernel)
04491             case (taylor_spline_n3_kernel)
04492                 call init_ts_3(kernel)
04493             case (laplacian_kernel)
04494                 call init_laplacian(kernel)
04495             case (bernoulli_n2_kernel)
04496                 call init_bernoulli_2(kernel)
04497             case default
04498                 write(*,*) "ERROR in module reproducing_kernels: "//&
04499                            "Unknown kernel_type"
04500                 stop
04501         end select
04502         return
```

#### 3.23.2.7 subroutine reproducing_kernels::init_laplacian ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to Laplacian kernel

Definition at line 5529 of file RKHS.f90.

```
05529         use kernel_laplacian
05530         implicit none
05531         type(kernel_1d), intent(out)  :: kernel
05532         kernel%kernel_type = laplacian_kernel
05533         kernel%M2   = m2   !M2
05534         kernel%Npar = npar !Npar
05535         if(allocated(kernel%par)) deallocate(kernel%par)
05536         allocate(kernel%par(npar))
05537         kernel%par = 1d0
05538         kernel%k    => k   !k
05539         kernel%dk   => dk  !dk
05540         kernel%d2k  => d2k !d2k
05541
05542         !allocate memory
05543         allocate(kernel%p2  (m2),&
05544                  kernel%f2  (m2),&
05545                  kernel%f3  (m2),&
05546                  kernel%df2 (m2),&
05547                  kernel%df3 (m2),&
05548                  kernel%d2f2(m2),&
05549                  kernel%d2f3(m2))
05550
05551         !set p2 coefficients
05552         kernel%p2(1) = p21
05553
05554         !set f2 functions
05555         kernel%f2(1)%f => f21
05556
05557         !set f3 functions
05558         kernel%f3(1)%f => f31
05559
05560         !set df2 functions
05561         kernel%df2(1)%f => df21
05562
05563         !set df3 functions
05564         kernel%df3(1)%f => df31
05565
05566         !set d2f2 functions
05567         kernel%d2f2(1)%f => d2f21
05568
05569         !set d2f3 functions
05570         kernel%d2f3(1)%f => d2f31
05571         return
```

#### 3.23.2.8 subroutine reproducing_kernels::init_rp_2_0 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 0

Definition at line 5161 of file RKHS.f90.

```
05161         use kernel_rp_2_0
05162         implicit none
05163         type(kernel_1d), intent(out)  :: kernel
05164         kernel%kernel_type = reciprocal_power_n2_m0_kernel
05165         kernel%M2   =  m2  !M2
05166         kernel%Npar =  npar!Npar
05167         kernel%k    => k   !k
05168         kernel%dk   => dk  !dk
05169         kernel%d2k  => d2k !d2k
05170
05171         !allocate memory
05172         allocate(kernel%p2  (m2),&
05173                  kernel%f2  (m2),&
05174                  kernel%f3  (m2),&
05175                  kernel%df2 (m2),&
05176                  kernel%df3 (m2),&
05177                  kernel%d2f2(m2),&
05178                  kernel%d2f3(m2))
05179
05180         !set p2 coefficients
05181         kernel%p2(1) = p21
05182         kernel%p2(2) = p22
05183
05184         !set f2 functions
05185         kernel%f2(1)%f => f21
05186         kernel%f2(2)%f => f22
05187
05188         !set f3 functions
05189         kernel%f3(1)%f => f31
05190         kernel%f3(2)%f => f32
05191
05192         !set df2 functions
05193         kernel%df2(1)%f => df21
05194         kernel%df2(2)%f => df22
```

```
05195
05196         !set df3 functions
05197         kernel%df3(1)%f => df31
05198         kernel%df3(2)%f => df32
05199
05200         !set d2f2 functions
05201         kernel%d2f2(1)%f => d2f21
05202         kernel%d2f2(2)%f => d2f22
05203
05204         !set d2f3 functions
05205         kernel%d2f3(1)%f => d2f31
05206         kernel%d2f3(2)%f => d2f32
05207         return
```

#### 3.23.2.9 subroutine reproducing_kernels::init_rp_2_1 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 1

Definition at line 5052 of file RKHS.f90.

```
05052         use kernel_rp_2_1
05053         implicit none
05054         type(kernel_1d), intent(out)  :: kernel
05055         kernel%kernel_type = reciprocal_power_n2_m1_kernel
05056         kernel%M2   =  m2  !M2
05057         kernel%Npar =  npar!Npar
05058         kernel%k    => k   !k
05059         kernel%dk   => dk  !dk
05060         kernel%d2k  => d2k !d2k
05061
05062         !allocate memory
05063         allocate(kernel%p2  (m2),&
05064                  kernel%f2  (m2),&
05065                  kernel%f3  (m2),&
05066                  kernel%df2 (m2),&
05067                  kernel%df3 (m2),&
05068                  kernel%d2f2(m2),&
05069                  kernel%d2f3(m2))
05070
05071         !set p2 coefficients
05072         kernel%p2(1) = p21
05073         kernel%p2(2) = p22
05074
05075         !set f2 functions
05076         kernel%f2(1)%f => f21
05077         kernel%f2(2)%f => f22
05078
05079         !set f3 functions
05080         kernel%f3(1)%f => f31
05081         kernel%f3(2)%f => f32
05082
05083         !set df2 functions
05084         kernel%df2(1)%f => df21
05085         kernel%df2(2)%f => df22
05086
05087         !set df3 functions
05088         kernel%df3(1)%f => df31
05089         kernel%df3(2)%f => df32
05090
05091         !set d2f2 functions
05092         kernel%d2f2(1)%f => d2f21
05093         kernel%d2f2(2)%f => d2f22
05094
05095         !set d2f3 functions
05096         kernel%d2f3(1)%f => d2f31
05097         kernel%d2f3(2)%f => d2f32
05098         return
```

#### 3.23.2.10 subroutine reproducing_kernels::init_rp_2_2 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 2

Definition at line 4943 of file RKHS.f90.

```
04943         use kernel_rp_2_2
04944         implicit none
04945         type(kernel_1d), intent(out)  :: kernel
04946         kernel%kernel_type = reciprocal_power_n2_m2_kernel
04947         kernel%M2   =  m2  !M2
04948         kernel%Npar =  npar!Npar
```

```
04949         kernel%k    => k   !k
04950         kernel%dk   => dk  !dk
04951         kernel%d2k  => d2k !d2k
04952
04953         !allocate memory
04954         allocate(kernel%p2  (m2),&
04955                  kernel%f2  (m2),&
04956                  kernel%f3  (m2),&
04957                  kernel%df2 (m2),&
04958                  kernel%df3 (m2),&
04959                  kernel%d2f2(m2),&
04960                  kernel%d2f3(m2))
04961
04962         !set p2 coefficients
04963         kernel%p2(1) = p21
04964         kernel%p2(2) = p22
04965
04966         !set f2 functions
04967         kernel%f2(1)%f => f21
04968         kernel%f2(2)%f => f22
04969
04970         !set f3 functions
04971         kernel%f3(1)%f => f31
04972         kernel%f3(2)%f => f32
04973
04974         !set df2 functions
04975         kernel%df2(1)%f => df21
04976         kernel%df2(2)%f => df22
04977
04978         !set df3 functions
04979         kernel%df3(1)%f => df31
04980         kernel%df3(2)%f => df32
04981
04982         !set d2f2 functions
04983         kernel%d2f2(1)%f => d2f21
04984         kernel%d2f2(2)%f => d2f22
04985
04986         !set d2f3 functions
04987         kernel%d2f3(1)%f => d2f31
04988         kernel%d2f3(2)%f => d2f32
04989         return
```

#### 3.23.2.11 subroutine reproducing_kernels::init_rp_2_3 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 3

Definition at line 4834 of file RKHS.f90.

```
04834         use kernel_rp_2_3
04835         implicit none
04836         type(kernel_1d), intent(out)  :: kernel
04837         kernel%kernel_type = reciprocal_power_n2_m3_kernel
04838         kernel%M2   =  m2  !M2
04839         kernel%Npar =  npar!Npar
04840         kernel%k    => k   !k
04841         kernel%dk   => dk  !dk
04842         kernel%d2k  => d2k !d2k
04843
04844         !allocate memory
04845         allocate(kernel%p2  (m2),&
04846                  kernel%f2  (m2),&
04847                  kernel%f3  (m2),&
04848                  kernel%df2 (m2),&
04849                  kernel%df3 (m2),&
04850                  kernel%d2f2(m2),&
04851                  kernel%d2f3(m2))
04852
04853         !set p2 coefficients
04854         kernel%p2(1) = p21
04855         kernel%p2(2) = p22
04856
04857         !set f2 functions
04858         kernel%f2(1)%f => f21
04859         kernel%f2(2)%f => f22
04860
04861         !set f3 functions
04862         kernel%f3(1)%f => f31
04863         kernel%f3(2)%f => f32
04864
04865         !set df2 functions
04866         kernel%df2(1)%f => df21
04867         kernel%df2(2)%f => df22
04868
04869         !set df3 functions
```

```
04870         kernel%df3(1)%f => df31
04871         kernel%df3(2)%f => df32
04872
04873         !set d2f2 functions
04874         kernel%d2f2(1)%f => d2f21
04875         kernel%d2f2(2)%f => d2f22
04876
04877         !set d2f3 functions
04878         kernel%d2f3(1)%f => d2f31
04879         kernel%d2f3(2)%f => d2f32
04880         return
```

#### 3.23.2.12 subroutine reproducing_kernels::init_rp_2_4 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 4

Definition at line 4725 of file RKHS.f90.

```
04725         use kernel_rp_2_4
04726         implicit none
04727         type(kernel_1d), intent(out)  :: kernel
04728         kernel%kernel_type = reciprocal_power_n2_m4_kernel
04729         kernel%M2   =  m2  !M2
04730         kernel%Npar =  npar!Npar
04731         kernel%k    => k   !k
04732         kernel%dk   => dk  !dk
04733         kernel%d2k  => d2k !d2k
04734
04735         !allocate memory
04736         allocate(kernel%p2  (m2),&
04737                  kernel%f2  (m2),&
04738                  kernel%f3  (m2),&
04739                  kernel%df2 (m2),&
04740                  kernel%df3 (m2),&
04741                  kernel%d2f2(m2),&
04742                  kernel%d2f3(m2))
04743
04744         !set p2 coefficients
04745         kernel%p2(1) = p21
04746         kernel%p2(2) = p22
04747
04748         !set f2 functions
04749         kernel%f2(1)%f => f21
04750         kernel%f2(2)%f => f22
04751
04752         !set f3 functions
04753         kernel%f3(1)%f => f31
04754         kernel%f3(2)%f => f32
04755
04756         !set df2 functions
04757         kernel%df2(1)%f => df21
04758         kernel%df2(2)%f => df22
04759
04760         !set df3 functions
04761         kernel%df3(1)%f => df31
04762         kernel%df3(2)%f => df32
04763
04764         !set d2f2 functions
04765         kernel%d2f2(1)%f => d2f21
04766         kernel%d2f2(2)%f => d2f22
04767
04768         !set d2f3 functions
04769         kernel%d2f3(1)%f => d2f31
04770         kernel%d2f3(2)%f => d2f32
04771         return
```

#### 3.23.2.13 subroutine reproducing_kernels::init_rp_2_5 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 5

Definition at line 4616 of file RKHS.f90.

```
04616         use kernel_rp_2_5
04617         implicit none
04618         type(kernel_1d), intent(out)  :: kernel
04619         kernel%kernel_type = reciprocal_power_n2_m5_kernel
04620         kernel%M2   =  m2  !M2
04621         kernel%Npar =  npar!Npar
04622         kernel%k    => k   !k
04623         kernel%dk   => dk  !dk
```

```
04624         kernel%d2k  => d2k !d2k
04625
04626         !allocate memory
04627         allocate(kernel%p2  (m2),&
04628                  kernel%f2  (m2),&
04629                  kernel%f3  (m2),&
04630                  kernel%df2 (m2),&
04631                  kernel%df3 (m2),&
04632                  kernel%d2f2(m2),&
04633                  kernel%d2f3(m2))
04634
04635         !set p2 coefficients
04636         kernel%p2(1) = p21
04637         kernel%p2(2) = p22
04638
04639         !set f2 functions
04640         kernel%f2(1)%f => f21
04641         kernel%f2(2)%f => f22
04642
04643         !set f3 functions
04644         kernel%f3(1)%f => f31
04645         kernel%f3(2)%f => f32
04646
04647         !set df2 functions
04648         kernel%df2(1)%f => df21
04649         kernel%df2(2)%f => df22
04650
04651         !set df3 functions
04652         kernel%df3(1)%f => df31
04653         kernel%df3(2)%f => df32
04654
04655         !set d2f2 functions
04656         kernel%d2f2(1)%f => d2f21
04657         kernel%d2f2(2)%f => d2f22
04658
04659         !set d2f3 functions
04660         kernel%d2f3(1)%f => d2f31
04661         kernel%d2f3(2)%f => d2f32
04662         return
```

#### 3.23.2.14 subroutine reproducing_kernels::init_rp_2_6 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 2 and m = 6

Definition at line 4507 of file RKHS.f90.

```
04507         use kernel_rp_2_6
04508         implicit none
04509         type(kernel_1d), intent(out)  :: kernel
04510         kernel%kernel_type = reciprocal_power_n2_m6_kernel
04511         kernel%M2   =  m2  !M2
04512         kernel%Npar =  npar!Npar
04513         kernel%k    => k   !k
04514         kernel%dk   => dk  !dk
04515         kernel%d2k  => d2k !d2k
04516
04517         !allocate memory
04518         allocate(kernel%p2  (m2),&
04519                  kernel%f2  (m2),&
04520                  kernel%f3  (m2),&
04521                  kernel%df2 (m2),&
04522                  kernel%df3 (m2),&
04523                  kernel%d2f2(m2),&
04524                  kernel%d2f3(m2))
04525
04526         !set p2 coefficients
04527         kernel%p2(1) = p21
04528         kernel%p2(2) = p22
04529
04530         !set f2 functions
04531         kernel%f2(1)%f => f21
04532         kernel%f2(2)%f => f22
04533
04534         !set f3 functions
04535         kernel%f3(1)%f => f31
04536         kernel%f3(2)%f => f32
04537
04538         !set df2 functions
04539         kernel%df2(1)%f => df21
04540         kernel%df2(2)%f => df22
04541
04542         !set df3 functions
04543         kernel%df3(1)%f => df31
04544         kernel%df3(2)%f => df32
```

```
04545
04546         !set d2f2 functions
04547         kernel%d2f2(1)%f => d2f21
04548         kernel%d2f2(2)%f => d2f22
04549
04550         !set d2f3 functions
04551         kernel%d2f3(1)%f => d2f31
04552         kernel%d2f3(2)%f => d2f32
04553         return
```

#### 3.23.2.15 subroutine reproducing_kernels::init_rp_3_0 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 0

Definition at line 5212 of file RKHS.f90.

```
05212         use kernel_rp_3_0
05213         implicit none
05214         type(kernel_1d), intent(out)  :: kernel
05215         kernel%kernel_type = reciprocal_power_n3_m0_kernel
05216         kernel%M2   =  m2  !M2
05217         kernel%Npar =  npar!Npar
05218         kernel%k    => k   !k
05219         kernel%dk   => dk  !dk
05220         kernel%d2k  => d2k !d2k
05221
05222         !allocate memory
05223         allocate(kernel%p2  (m2),&
05224                  kernel%f2  (m2),&
05225                  kernel%f3  (m2),&
05226                  kernel%df2 (m2),&
05227                  kernel%df3 (m2),&
05228                  kernel%d2f2(m2),&
05229                  kernel%d2f3(m2))
05230
05231         !set p2 coefficients
05232         kernel%p2(1) = p21
05233         kernel%p2(2) = p22
05234         kernel%p2(3) = p23
05235
05236         !set f2 functions
05237         kernel%f2(1)%f => f21
05238         kernel%f2(2)%f => f22
05239         kernel%f2(3)%f => f23
05240
05241         !set f3 functions
05242         kernel%f3(1)%f => f31
05243         kernel%f3(2)%f => f32
05244         kernel%f3(3)%f => f33
05245
05246         !set df2 functions
05247         kernel%df2(1)%f => df21
05248         kernel%df2(2)%f => df22
05249         kernel%df2(3)%f => df23
05250
05251         !set df3 functions
05252         kernel%df3(1)%f => df31
05253         kernel%df3(2)%f => df32
05254         kernel%df3(3)%f => df33
05255
05256         !set d2f2 functions
05257         kernel%d2f2(1)%f => d2f21
05258         kernel%d2f2(2)%f => d2f22
05259         kernel%d2f2(3)%f => d2f23
05260
05261         !set d2f3 functions
05262         kernel%d2f3(1)%f => d2f31
05263         kernel%d2f3(2)%f => d2f32
05264         kernel%d2f3(3)%f => d2f33
05265         return
```

#### 3.23.2.16 subroutine reproducing_kernels::init_rp_3_1 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 1

Definition at line 5103 of file RKHS.f90.

```
05103         use kernel_rp_3_1
05104         implicit none
05105         type(kernel_1d), intent(out)  :: kernel
```

```
05106         kernel%kernel_type = reciprocal_power_n3_m1_kernel
05107         kernel%M2   =  m2  !M2
05108         kernel%Npar =  npar!Npar
05109         kernel%k    => k   !k
05110         kernel%dk   => dk  !dk
05111         kernel%d2k  => d2k !d2k
05112
05113         !allocate memory
05114         allocate(kernel%p2  (m2),&
05115                  kernel%f2  (m2),&
05116                  kernel%f3  (m2),&
05117                  kernel%df2 (m2),&
05118                  kernel%df3 (m2),&
05119                  kernel%d2f2(m2),&
05120                  kernel%d2f3(m2))
05121
05122         !set p2 coefficients
05123         kernel%p2(1) = p21
05124         kernel%p2(2) = p22
05125         kernel%p2(3) = p23
05126
05127         !set f2 functions
05128         kernel%f2(1)%f => f21
05129         kernel%f2(2)%f => f22
05130         kernel%f2(3)%f => f23
05131
05132         !set f3 functions
05133         kernel%f3(1)%f => f31
05134         kernel%f3(2)%f => f32
05135         kernel%f3(3)%f => f33
05136
05137         !set df2 functions
05138         kernel%df2(1)%f => df21
05139         kernel%df2(2)%f => df22
05140         kernel%df2(3)%f => df23
05141
05142         !set df3 functions
05143         kernel%df3(1)%f => df31
05144         kernel%df3(2)%f => df32
05145         kernel%df3(3)%f => df33
05146
05147         !set d2f2 functions
05148         kernel%d2f2(1)%f => d2f21
05149         kernel%d2f2(2)%f => d2f22
05150         kernel%d2f2(3)%f => d2f23
05151
05152         !set d2f3 functions
05153         kernel%d2f3(1)%f => d2f31
05154         kernel%d2f3(2)%f => d2f32
05155         kernel%d2f3(3)%f => d2f33
05156         return
```

#### 3.23.2.17 subroutine reproducing_kernels::init_rp_3_2 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 2

Definition at line 4994 of file RKHS.f90.

```
04994         use kernel_rp_3_2
04995         implicit none
04996         type(kernel_1d), intent(out)  :: kernel
04997         kernel%kernel_type = reciprocal_power_n3_m2_kernel
04998         kernel%M2   =  m2  !M2
04999         kernel%Npar =  npar!Npar
05000         kernel%k    => k   !k
05001         kernel%dk   => dk  !dk
05002         kernel%d2k  => d2k !d2k
05003
05004         !allocate memory
05005         allocate(kernel%p2  (m2),&
05006                  kernel%f2  (m2),&
05007                  kernel%f3  (m2),&
05008                  kernel%df2 (m2),&
05009                  kernel%df3 (m2),&
05010                  kernel%d2f2(m2),&
05011                  kernel%d2f3(m2))
05012
05013         !set p2 coefficients
05014         kernel%p2(1) = p21
05015         kernel%p2(2) = p22
05016         kernel%p2(3) = p23
05017
05018         !set f2 functions
05019         kernel%f2(1)%f => f21
```

```
05020         kernel%f2(2)%f => f22
05021         kernel%f2(3)%f => f23
05022
05023         !set f3 functions
05024         kernel%f3(1)%f => f31
05025         kernel%f3(2)%f => f32
05026         kernel%f3(3)%f => f33
05027
05028         !set df2 functions
05029         kernel%df2(1)%f => df21
05030         kernel%df2(2)%f => df22
05031         kernel%df2(3)%f => df23
05032
05033         !set df3 functions
05034         kernel%df3(1)%f => df31
05035         kernel%df3(2)%f => df32
05036         kernel%df3(3)%f => df33
05037
05038         !set d2f2 functions
05039         kernel%d2f2(1)%f => d2f21
05040         kernel%d2f2(2)%f => d2f22
05041         kernel%d2f2(3)%f => d2f23
05042
05043         !set d2f3 functions
05044         kernel%d2f3(1)%f => d2f31
05045         kernel%d2f3(2)%f => d2f32
05046         kernel%d2f3(3)%f => d2f33
05047         return
```

#### 3.23.2.18 subroutine reproducing_kernels::init_rp_3_3 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 3

Definition at line 4885 of file RKHS.f90.

```
04885         use kernel_rp_3_3
04886         implicit none
04887         type(kernel_1d), intent(out)  :: kernel
04888         kernel%kernel_type = reciprocal_power_n3_m3_kernel
04889         kernel%M2   =  m2  !M2
04890         kernel%Npar =  npar!Npar
04891         kernel%k    => k   !k
04892         kernel%dk   => dk  !dk
04893         kernel%d2k  => d2k !d2k
04894
04895         !allocate memory
04896         allocate(kernel%p2  (m2),&
04897                  kernel%f2  (m2),&
04898                  kernel%f3  (m2),&
04899                  kernel%df2 (m2),&
04900                  kernel%df3 (m2),&
04901                  kernel%d2f2(m2),&
04902                  kernel%d2f3(m2))
04903
04904         !set p2 coefficients
04905         kernel%p2(1) = p21
04906         kernel%p2(2) = p22
04907         kernel%p2(3) = p23
04908
04909         !set f2 functions
04910         kernel%f2(1)%f => f21
04911         kernel%f2(2)%f => f22
04912         kernel%f2(3)%f => f23
04913
04914         !set f3 functions
04915         kernel%f3(1)%f => f31
04916         kernel%f3(2)%f => f32
04917         kernel%f3(3)%f => f33
04918
04919         !set df2 functions
04920         kernel%df2(1)%f => df21
04921         kernel%df2(2)%f => df22
04922         kernel%df2(3)%f => df23
04923
04924         !set df3 functions
04925         kernel%df3(1)%f => df31
04926         kernel%df3(2)%f => df32
04927         kernel%df3(3)%f => df33
04928
04929         !set d2f2 functions
04930         kernel%d2f2(1)%f => d2f21
04931         kernel%d2f2(2)%f => d2f22
04932         kernel%d2f2(3)%f => d2f23
04933
```

```
04934         !set d2f3 functions
04935         kernel%d2f3(1)%f => d2f31
04936         kernel%d2f3(2)%f => d2f32
04937         kernel%d2f3(3)%f => d2f33
04938         return
```

#### 3.23.2.19 subroutine reproducing_kernels::init_rp_3_4 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 4

Definition at line 4776 of file RKHS.f90.

```
04776         use kernel_rp_3_4
04777         implicit none
04778         type(kernel_1d), intent(out)  :: kernel
04779         kernel%kernel_type = reciprocal_power_n3_m4_kernel
04780         kernel%M2    =  m2  !M2
04781         kernel%Npar =  npar!Npar
04782         kernel%k     => k   !k
04783         kernel%dk    => dk  !dk
04784         kernel%d2k   => d2k !d2k
04785
04786         !allocate memory
04787         allocate(kernel%p2  (m2),&
04788                  kernel%f2  (m2),&
04789                  kernel%f3  (m2),&
04790                  kernel%df2 (m2),&
04791                  kernel%df3 (m2),&
04792                  kernel%d2f2(m2),&
04793                  kernel%d2f3(m2))
04794
04795         !set p2 coefficients
04796         kernel%p2(1) = p21
04797         kernel%p2(2) = p22
04798         kernel%p2(3) = p23
04799
04800         !set f2 functions
04801         kernel%f2(1)%f => f21
04802         kernel%f2(2)%f => f22
04803         kernel%f2(3)%f => f23
04804
04805         !set f3 functions
04806         kernel%f3(1)%f => f31
04807         kernel%f3(2)%f => f32
04808         kernel%f3(3)%f => f33
04809
04810         !set df2 functions
04811         kernel%df2(1)%f => df21
04812         kernel%df2(2)%f => df22
04813         kernel%df2(3)%f => df23
04814
04815         !set df3 functions
04816         kernel%df3(1)%f => df31
04817         kernel%df3(2)%f => df32
04818         kernel%df3(3)%f => df33
04819
04820         !set d2f2 functions
04821         kernel%d2f2(1)%f => d2f21
04822         kernel%d2f2(2)%f => d2f22
04823         kernel%d2f2(3)%f => d2f23
04824
04825         !set d2f3 functions
04826         kernel%d2f3(1)%f => d2f31
04827         kernel%d2f3(2)%f => d2f32
04828         kernel%d2f3(3)%f => d2f33
04829         return
```

#### 3.23.2.20 subroutine reproducing_kernels::init_rp_3_5 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 5

Definition at line 4667 of file RKHS.f90.

```
04667         use kernel_rp_3_5
04668         implicit none
04669         type(kernel_1d), intent(out)  :: kernel
04670         kernel%kernel_type = reciprocal_power_n3_m5_kernel
04671         kernel%M2    =  m2  !M2
04672         kernel%Npar =  npar!Npar
04673         kernel%k     => k   !k
```

```
04674         kernel%dk   => dk  !dk
04675         kernel%d2k  => d2k !d2k
04676
04677         !allocate memory
04678         allocate(kernel%p2  (m2),&
04679                  kernel%f2  (m2),&
04680                  kernel%f3  (m2),&
04681                  kernel%df2 (m2),&
04682                  kernel%df3 (m2),&
04683                  kernel%d2f2(m2),&
04684                  kernel%d2f3(m2))
04685
04686         !set p2 coefficients
04687         kernel%p2(1) = p21
04688         kernel%p2(2) = p22
04689         kernel%p2(3) = p23
04690
04691         !set f2 functions
04692         kernel%f2(1)%f => f21
04693         kernel%f2(2)%f => f22
04694         kernel%f2(3)%f => f23
04695
04696         !set f3 functions
04697         kernel%f3(1)%f => f31
04698         kernel%f3(2)%f => f32
04699         kernel%f3(3)%f => f33
04700
04701         !set df2 functions
04702         kernel%df2(1)%f => df21
04703         kernel%df2(2)%f => df22
04704         kernel%df2(3)%f => df23
04705
04706         !set df3 functions
04707         kernel%df3(1)%f => df31
04708         kernel%df3(2)%f => df32
04709         kernel%df3(3)%f => df33
04710
04711         !set d2f2 functions
04712         kernel%d2f2(1)%f => d2f21
04713         kernel%d2f2(2)%f => d2f22
04714         kernel%d2f2(3)%f => d2f23
04715
04716         !set d2f3 functions
04717         kernel%d2f3(1)%f => d2f31
04718         kernel%d2f3(2)%f => d2f32
04719         kernel%d2f3(3)%f => d2f33
04720         return
```

#### 3.23.2.21 subroutine reproducing_kernels::init_rp_3_6 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to reciprocal decay kernel with n = 3 and m = 6

Definition at line 4558 of file RKHS.f90.

```
04558         use kernel_rp_3_6
04559         implicit none
04560         type(kernel_1d), intent(out)  :: kernel
04561         kernel%kernel_type = reciprocal_power_n3_m6_kernel
04562         kernel%M2   =  m2  !M2
04563         kernel%Npar =  npar!Npar
04564         kernel%k    => k   !k
04565         kernel%dk   => dk  !dk
04566         kernel%d2k  => d2k !d2k
04567
04568         !allocate memory
04569         allocate(kernel%p2  (m2),&
04570                  kernel%f2  (m2),&
04571                  kernel%f3  (m2),&
04572                  kernel%df2 (m2),&
04573                  kernel%df3 (m2),&
04574                  kernel%d2f2(m2),&
04575                  kernel%d2f3(m2))
04576
04577         !set p2 coefficients
04578         kernel%p2(1) = p21
04579         kernel%p2(2) = p22
04580         kernel%p2(3) = p23
04581
04582         !set f2 functions
04583         kernel%f2(1)%f => f21
04584         kernel%f2(2)%f => f22
04585         kernel%f2(3)%f => f23
04586
04587         !set f3 functions
```

```
04588         kernel%f3(1)%f => f31
04589         kernel%f3(2)%f => f32
04590         kernel%f3(3)%f => f33
04591
04592         !set df2 functions
04593         kernel%df2(1)%f => df21
04594         kernel%df2(2)%f => df22
04595         kernel%df2(3)%f => df23
04596
04597         !set df3 functions
04598         kernel%df3(1)%f => df31
04599         kernel%df3(2)%f => df32
04600         kernel%df3(3)%f => df33
04601
04602         !set d2f2 functions
04603         kernel%d2f2(1)%f => d2f21
04604         kernel%d2f2(2)%f => d2f22
04605         kernel%d2f2(3)%f => d2f23
04606
04607         !set d2f3 functions
04608         kernel%d2f3(1)%f => d2f31
04609         kernel%d2f3(2)%f => d2f32
04610         kernel%d2f3(3)%f => d2f33
04611         return
```

#### 3.23.2.22 subroutine reproducing_kernels::init_ts_2 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to Taylor spline kernel with n = 2

Definition at line 5399 of file RKHS.f90.

```
05399         use kernel_ts_2
05400         implicit none
05401         type(kernel_1d), intent(out)  :: kernel
05402         kernel%kernel_type = taylor_spline_n2_kernel
05403         kernel%M2   =  m2  !M2
05404         kernel%Npar =  npar!Npar
05405         kernel%k    => k   !k
05406         kernel%dk   => dk  !dk
05407         kernel%d2k  => d2k !d2k
05408
05409         !allocate memory
05410         allocate(kernel%p2  (m2),&
05411                  kernel%f2  (m2),&
05412                  kernel%f3  (m2),&
05413                  kernel%df2 (m2),&
05414                  kernel%df3 (m2),&
05415                  kernel%d2f2(m2),&
05416                  kernel%d2f3(m2))
05417
05418         !set p2 coefficients
05419         kernel%p2(1) = p21
05420         kernel%p2(2) = p22
05421         kernel%p2(3) = p23
05422
05423         !set f2 functions
05424         kernel%f2(1)%f => f21
05425         kernel%f2(2)%f => f22
05426         kernel%f2(3)%f => f23
05427
05428         !set f3 functions
05429         kernel%f3(1)%f => f31
05430         kernel%f3(2)%f => f32
05431         kernel%f3(3)%f => f33
05432
05433         !set df2 functions
05434         kernel%df2(1)%f => df21
05435         kernel%df2(2)%f => df22
05436         kernel%df2(3)%f => df23
05437
05438         !set df3 functions
05439         kernel%df3(1)%f => df31
05440         kernel%df3(2)%f => df32
05441         kernel%df3(3)%f => df33
05442
05443         !set d2f2 functions
05444         kernel%d2f2(1)%f => d2f21
05445         kernel%d2f2(2)%f => d2f22
05446         kernel%d2f2(3)%f => d2f23
05447
05448         !set d2f3 functions
05449         kernel%d2f3(1)%f => d2f31
05450         kernel%d2f3(2)%f => d2f32
05451         kernel%d2f3(3)%f => d2f33
```

```
05452         return
```

#### 3.23.2.23 subroutine reproducing_kernels::init_ts_3 ( type(kernel_1d), intent(out) *kernel* )

initializes the kernel to Taylor spline kernel with n = 3

Definition at line 5457 of file RKHS.f90.

```
05457         use kernel_ts_3
05458         implicit none
05459         type(kernel_1d), intent(out)  :: kernel
05460         kernel%kernel_type = taylor_spline_n3_kernel
05461         kernel%M2   =  m2  !M2
05462         kernel%Npar =  npar!Npar
05463         kernel%k    => k   !k
05464         kernel%dk   => dk  !dk
05465         kernel%d2k  => d2k !d2k
05466
05467         !allocate memory
05468         allocate(kernel%p2  (m2),&
05469                  kernel%f2  (m2),&
05470                  kernel%f3  (m2),&
05471                  kernel%df2 (m2),&
05472                  kernel%df3 (m2),&
05473                  kernel%d2f2(m2),&
05474                  kernel%d2f3(m2))
05475
05476         !set p2 coefficients
05477         kernel%p2(1) = p21
05478         kernel%p2(2) = p22
05479         kernel%p2(3) = p23
05480         kernel%p2(4) = p24
05481         kernel%p2(5) = p25
05482
05483         !set f2 functions
05484         kernel%f2(1)%f => f21
05485         kernel%f2(2)%f => f22
05486         kernel%f2(3)%f => f23
05487         kernel%f2(4)%f => f24
05488         kernel%f2(5)%f => f25
05489
05490         !set f3 functions
05491         kernel%f3(1)%f => f31
05492         kernel%f3(2)%f => f32
05493         kernel%f3(3)%f => f33
05494         kernel%f3(4)%f => f34
05495         kernel%f3(5)%f => f35
05496
05497         !set df2 functions
05498         kernel%df2(1)%f => df21
05499         kernel%df2(2)%f => df22
05500         kernel%df2(3)%f => df23
05501         kernel%df2(4)%f => df24
05502         kernel%df2(5)%f => df25
05503
05504         !set df3 functions
05505         kernel%df3(1)%f => df31
05506         kernel%df3(2)%f => df32
05507         kernel%df3(3)%f => df33
05508         kernel%df3(4)%f => df34
05509         kernel%df3(5)%f => df35
05510
05511         !set d2f2 functions
05512         kernel%d2f2(1)%f => d2f21
05513         kernel%d2f2(2)%f => d2f22
05514         kernel%d2f2(3)%f => d2f23
05515         kernel%d2f2(4)%f => d2f24
05516         kernel%d2f2(5)%f => d2f25
05517
05518         !set d2f3 functions
05519         kernel%d2f3(1)%f => d2f31
05520         kernel%d2f3(2)%f => d2f32
05521         kernel%d2f3(3)%f => d2f33
05522         kernel%d2f3(4)%f => d2f34
05523         kernel%d2f3(5)%f => d2f35
05524         return
```

### 3.23.3 Variable Documentation

#### 3.23.3.1 integer parameter reproducing_kernels::bernoulli_n2_kernel = 19

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.2 integer parameter reproducing_kernels::exponential_decay_n2_kernel = 14

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.3 integer parameter reproducing_kernels::exponential_decay_n3_kernel = 15

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.4 integer parameter reproducing_kernels::laplacian_kernel = 18

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.5 integer parameter reproducing_kernels::reciprocal_power_n2_m0_kernel = 12

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.6 integer parameter reproducing_kernels::reciprocal_power_n2_m1_kernel = 10

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.7 integer parameter reproducing_kernels::reciprocal_power_n2_m2_kernel = 8

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.8 integer parameter reproducing_kernels::reciprocal_power_n2_m3_kernel = 6

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.9 integer parameter reproducing_kernels::reciprocal_power_n2_m4_kernel = 4

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.10 integer parameter reproducing_kernels::reciprocal_power_n2_m5_kernel = 2

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.11 integer parameter reproducing_kernels::reciprocal_power_n2_m6_kernel = 0

enumerator for kernel types

Definition at line 4323 of file RKHS.f90.

```
04323     integer, parameter :: reciprocal_power_n2_m6_kernel =  0, &
04324                           reciprocal_power_n3_m6_kernel =  1, &
04325                           reciprocal_power_n2_m5_kernel =  2, &
04326                           reciprocal_power_n3_m5_kernel =  3, &
04327                           reciprocal_power_n2_m4_kernel =  4, &
04328                           reciprocal_power_n3_m4_kernel =  5, &
04329                           reciprocal_power_n2_m3_kernel =  6, &
04330                           reciprocal_power_n3_m3_kernel =  7, &
04331                           reciprocal_power_n2_m2_kernel =  8, &
04332                           reciprocal_power_n3_m2_kernel =  9, &
04333                           reciprocal_power_n2_m1_kernel = 10, &
04334                           reciprocal_power_n3_m1_kernel = 11, &
04335                           reciprocal_power_n2_m0_kernel = 12, &
04336                           reciprocal_power_n3_m0_kernel = 13, &
04337                           exponential_decay_n2_kernel   = 14, &
04338                           exponential_decay_n3_kernel   = 15, &
04339                           taylor_spline_n2_kernel       = 16, &
04340                           taylor_spline_n3_kernel       = 17, &
04341                           laplacian_kernel              = 18, &
04342                           bernoulli_n2_kernel           = 19
```

#### 3.23.3.12 integer parameter reproducing_kernels::reciprocal_power_n3_m0_kernel = 13

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.13 integer parameter reproducing_kernels::reciprocal_power_n3_m1_kernel = 11

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.14 integer parameter reproducing_kernels::reciprocal_power_n3_m2_kernel = 9

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.15 integer parameter reproducing_kernels::reciprocal_power_n3_m3_kernel = 7

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.16 integer parameter reproducing_kernels::reciprocal_power_n3_m4_kernel = 5

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.17 integer parameter reproducing_kernels::reciprocal_power_n3_m5_kernel = 3

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.18 integer parameter reproducing_kernels::reciprocal_power_n3_m6_kernel = 1

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.19 integer parameter reproducing_kernels::taylor_spline_n2_kernel = 16

Definition at line 4323 of file RKHS.f90.

#### 3.23.3.20 integer parameter reproducing_kernels::taylor_spline_n3_kernel = 17

Definition at line 4323 of file RKHS.f90.

## 3.24 rkhs Module Reference

main RKHS module (this is the only module that needs to be directly used)

### Data Types

- type grid

  *contained in kernel type, stores grid points for each dimension*
- type kernel

  *multi-dimensional kernel type (contains 1-d kernels and lookup tables, coefficients, etc.)*
- type kernel_matrix

  *wrapper for matrices, used to store a kernel matrix in tensor product form*

### Functions/Subroutines

- subroutine throw_error (proc, errormsg)

  *Throws error messages if problems are encountered.*
- subroutine save_kernel_to_file (Q, datafile)

  *Saves kernel in a binary file, so that it can be reused later.*
- subroutine load_kernel_from_file (Q, datafile)

  *Loads kernel from a binary file.*

- subroutine deallocate_kernel (Q)

  *Frees kernel resources (deallocates memory)*
- integer function get_sig_idx_from_m_and_k (Q, Ndim, m, k)

  *Computes the index in the lookup table of precomputed sums for m and k.*
- integer function get_sig_idx_from_indices (Q, indices)

  *Computes the index for retrieving the correct lookup table for a certain index combination.*
- integer function get_idx_from_indices (Q, indices)

  *Computes the index for retrieving the correct alpha coefficient.*
- subroutine read_grid_data (Q, datafile)

  *Reads grid data from a .csv file and automatically allocates all necessary memory.*
- subroutine write_grid_data (Q, datafile)

  *Writes grid data as .csv file.*
- subroutine calculate_coefficients_slow (Q, alpha, calcInverse)

  *Calculates the kernel coefficients using the naive way (direct inversion of full matrix)*
- subroutine fast_tensor_solve (Q, KM, solution)

  *Solves a system of linear equations in Tensor product form efficiently.*
- subroutine calculate_coefficients_fast (Q)

  *Calculates the kernel coefficients using the fast way.*
- subroutine calculate_single_sum (Q, sigma, m, k)

  *Computes the presummed values needed for the fast kernel evaluation for a single point.*
- subroutine allocate_sigma_gamma (Q)

  *Allocates memory for the lookup table.*
- subroutine calculate_sums (Q)

  *Computes the presummed values needed for the fast kernel evaluation.*
- integer function binary_search (array, value)

  *Binary search algorithm to find correct indices for lookup tables.*
- subroutine evaluate_kernel_fast (Q, x, f, df, d2f, df_mask, d2f_mask)

  *Evaluates the kernel using the fast algorithm and returns the function value (and derivatives).*
- subroutine evaluate_kernel_slow (Q, x, f, df, d2f, df_mask, d2f_mask)

  *Evaluates the kernel using the naive (slow) algorithm and returns the function value (and derivatives).*
- subroutine calculate_error_bound (Q, x, errorbound)

  *Calculates the error bound of the kernel at position x.*
- subroutine evaluate_kernel_fast (Q, x, f, df)

  *Evaluates the kernel using the fast algorithm and returns the function value (and derivatives).*
- subroutine evaluate_kernel_slow (Q, x, f, df)

  *Evaluates the kernel using the naive (slow) algorithm and returns the function value (and derivatives).*

#### 3.24.1 Detailed Description

main RKHS module (this is the only module that needs to be directly used)

**Author**

Oliver T. Unke, University of Basel

This module contains all necessary type definitions and functions in order to setup an arbitrary kernel for interpolating multi-dimensional functions. The method was first described in: T.-S. Ho and H. Rabitz. "A general method for constructing multidimensional molecular potential energy surfaces from abinitio calculations." The Journal of chemical physics 104.7 (1996): 2584-2597. The fast algorithm was first described in: T. Hollebeek, T.-S. Ho, and H. Rabitz. "A fast algorithm for evaluating multidimensional potential energy surfaces." The Journal of chemical physics 106.17 (1997): 7223-7227. And a useful review article can be found in: T. Hollebeek, T.-S. Ho, and H. Rabitz. "Constructing multidimensional molecular potential energy surfaces from ab initio data." Annual review of physical chemistry 50.1 (1999): 537-570.

### 3.24.2 Function/Subroutine Documentation

#### 3.24.2.1 subroutine rkhs::allocate_sigma_gamma ( type(kernel) *Q* ) [private]

Allocates memory for the lookup table.

**Author**

Oliver T. Unke, University of Basel

Since the amount of memory for storing the lookup table can get quite large when an extremely high-dimensional grid with many points is used, this subroutine will print the approximate RAM requirement in case the allocation fails.

Definition at line 6810 of file RKHS.f90.

```
06810     type(kernel)        :: q
06811     integer             :: alloc_status
06812     character(len=1024) :: alloc_err_msg
06813     integer             :: prod,d
06814
06815     if(allocated(q%sigma))     deallocate(q%sigma)
06816     if(allocated(q%gamma_old)) deallocate(q%gamma_old)
06817     if(allocated(q%gamma_new)) deallocate(q%gamma_new)
06818
06819     !determine how much memory to allocate to sigma
06820     prod = 1
06821     do d = 1,q%Ndim
06822         prod = prod * (size(q%grid(d)%x)+1)
06823     end do
06824
06825     !allocate memory for the storage of the sums
06826     allocate(q%sigma(2**q%Ndim*product(q%k1d(:)%M2),prod),&
06827              q%gamma_old(2**q%Ndim*product(q%k1d(:)%M2)),&
06828              q%gamma_new(2**q%Ndim*product(q%k1d(:)%M2)), stat=alloc_status)
06829     if(alloc_status /= 0) then
06830         write(alloc_err_msg,'(I0)') sizeof(0d0)*2**q%Ndim*product(q%k1d(:)%M2)*(prod+2)
06831         call throw_error("allocate_sigma_gamma","Could not allocate enough memory for storing precomputed
       sums. "//&
06832                                          "Estimated RAM requirement: "//trim(alloc_err_msg)//" bytes.")
06833     end if
06834     return
```

#### 3.24.2.2 integer function rkhs::binary_search ( real(kind(0d0)), dimension(:), intent(in) *array,* real(kind(0d0)), intent(in) *value* ) [private]

Binary search algorithm to find correct indices for lookup tables.

**Author**

Oliver T. Unke, University of Basel

Given an ascendingly sorted array of values and a search value, this function returns the index i of the element xi in array which satisfies xi <= value < xi+1. In the case that the smallest element is smaller than value, the function returns 0.

Definition at line 6903 of file RKHS.f90.

```
06903     implicit none
06904     real(kind(0d0)), dimension(:), intent(in) :: array
06905     real(kind(0d0)), intent(in)               :: value
06906     integer                                   :: l,r,m
06907
06908     !if value is smaller or larger than the contents of the array, we can immediately leave
06909     if (array(1) > value) then
06910         binary_search = 0
06911         return
06912     else if (array(size(array)) < value) then
06913         binary_search = size(array)
06914         return
06915     end if
06916
06917     l = 1
06918     r = size(array)
```

```
06919     do while(l <= r)
06920         m = (l+r)/2
06921         if(array(m) > value) then
06922             r = m-1
06923         else
06924             l = m+1
06925         end if
06926     end do
06927     binary_search = (l+r)/2
```

#### 3.24.2.3 subroutine rkhs::calculate_coefficients_fast ( class(kernel) *Q* ) [private]

Calculates the kernel coefficients using the fast way.

**Author**

Oliver T. Unke, University of Basel

This subroutine calculates the kernel coefficients using the fact that we have a linear system in tensor product form, therefore making the solution much easier. Note that this method to calculate the kernel coefficients should always be prefered over calculate_coefficients_slow, unless the matrix is ill conditioned. In that case the Tikhonov regularization procedure needs to be used, which is only possible with the slow method. Note that it is not possible to calculate the error bound if the fast method is used (because a calculation of the full inverse matrix is avoided - hence it's fast). If error bounds are desired, the slow method needs to be invoked. Holes in the grid are handled automatically.

Definition at line 6511 of file RKHS.f90.

```
06511     implicit none
06512     class(kernel)                                       :: q
06513     type(kernel)                                        :: kq !temporary kernel for interpolations

06514     type(kernel_matrix), dimension(Q%Ndim)              :: km
06515     real(kind(0d0)), dimension(&
        count(.not.Q%valueIsPresent),&
        count(.not.Q%valueIsPresent))                 :: cholq
06516     real(kind(0d0)), dimension(&
        Q%Npoint,&
        count(.not.Q%valueIsPresent))                 :: invq !only a subslice of the full inverse matrix
06517     real(kind(0d0)), dimension(&
        count(.not.Q%valueIsPresent))                 :: deltai
06518     integer, dimension(count(.not.Q%valueIsPresent)) :: indxnotpresent !keeps track of which values are
      missing
06519     real(kind(0d0)), dimension(Q%Npoint)                :: invqslice !slice of the full inverse matrix
06520     real(kind(0d0))                                     :: cdiff !for correcting coefficients when holes are
      present
06521     integer                                             :: i,j,k,counter,counteri,counterj,indxi,indxj,sizedim
06522     integer                                             :: alloc_status
06523
06524     !build the kernel matrices for all dimensions and make their cholesky decomposition
06525     do i = 1,q%Ndim
06526         !initialize
06527         if(allocated(km(i)%M)) deallocate(km(i)%M)
06528         allocate(km(i)%M(size(q%grid(i)%x),size(q%grid(i)%x)),stat=alloc_status)
06529         if(alloc_status /= 0) call throw_error("calculate_coefficients_fast","could not allocate enough
      memory.")
06530
06531         !build the kernel matrices (only the lower diagonal part)
06532         do j = 1,size(q%grid(i)%x)
06533             do k = 1,j
06534                 km(i)%M(j,k) = q%k1d(i)%k(q%grid(i)%x(j),q%grid(i)%x(k),q%k1d(i)%par)
06535             end do
06536         end do
06537
06538         !do the Cholesky decomposition of the kernel matrices
06539         call cholesky_decomposition(km(i)%M)
06540     end do
06541
06542     !calculate coefficients
06543     if(.not.any(.not.q%valueIsPresent)) then !straightforward if no holes present
06544         q%alpha = q%values !initialize alpha to solutions
06545         call fast_tensor_solve(q,km,q%alpha) !solve for alpha
06546     else !a bit more sophisticated if holes are present
06547         !set up the temporary 1d kernel for constructing guess interpolations
06548         call kq%free() !make sure nothing is allocated
06549         kq%Ndim   = 1
06550         kq%Npoint = size(q%grid(q%Ndim)%x)
06551         allocate(kq%k1d(kq%Ndim), kq%grid(kq%Ndim), kq%values(kq%Npoint), kq%alpha(kq%Npoint),&
```

```
06552                 kq%valueIsPresent(kq%Npoint), stat=alloc_status)
06553         if(alloc_status /= 0) call throw_error("calculate_coefficients_fast","could not allocate memory.")
06554         kq%k1d(1) = q%k1d(q%Ndim)
06555         allocate(kq%grid(1)%x(kq%Npoint), stat=alloc_status)
06556         if(alloc_status /= 0) call throw_error("calculate_coefficients_fast","could not allocate memory.")
06557         kq%grid(1)%x = q%grid(q%Ndim)%x
06558
06559         !construct guesses for missing function values
06560         do i = 1,q%Npoint,kq%Npoint
06561             !set up 1-d kernel for this cut
06562             kq%valueIsPresent = q%valueIsPresent(i:i+kq%Npoint-1)
06563             if(count(kq%valueIsPresent) > 0) then !if there is at least 1 point in the grid
06564                 kq%values         = q%values(i:i+kq%Npoint-1)
06565                 call kq%calculate_coefficients_slow()
06566                 call kq%calculate_sums()
06567                 do j = 1,kq%Npoint
06568                     if(.not.kq%valueIsPresent(j)) then
06569                         call kq%evaluate_fast((/kq%grid(1)%x(j)/),q%values(i+j-1))
06570                     end if
06571                 end do
06572             else !no point at all present -> no available information!
06573                 q%values(i:i+kq%Npoint-1) = 0d0
06574             end if
06575         end do
06576
06577         !first calculate solution with "arbitrary" function values
06578         q%alpha = q%values !initialize alpha to solutions
06579         call fast_tensor_solve(q,km,q%alpha)
06580
06581         !build small submatrix
06582         counteri = 0
06583         !first find the indices of missing points
06584         do i = 1,q%Npoint
06585             if(.not.q%valueIsPresent(i)) then
06586                 counteri = counteri + 1
06587                 indxnotpresent(counteri) = i
06588             end if
06589         end do
06590
06591         !construct submatrix of inverse (for all missing points)
06592         do i = 1,size(indxnotpresent)
06593             !compute a slice of the full inverse matrix
06594             invq(:,i)                   = 0d0
06595             invq(indxnotpresent(i),i) = 1d0
06596             call fast_tensor_solve(q,km,invq(:,i))
06597             do j = 1,i !only the lower triangular part is needed
06598                 cholq(i,j) = invq(indxnotpresent(j),i) !store part of the full slice
06599             end do
06600         end do
06601
06602         !do cholesky decomposition of submatrix and solve for the unknown deltai
06603         call cholesky_decomposition(cholq)
06604         call cholesky_solve(cholq,deltai,pack(q%alpha,.not.q%valueIsPresent))
06605
06606         !now that deltai is known, we can correct the coefficients
06607         counteri = 0
06608         do i = 1,q%Npoint
06609             if(q%valueIsPresent(i)) then
06610                 counteri = counteri + 1
06611                 cdiff = 0d0
06612                 do j = 1,size(indxnotpresent)
06613                     cdiff = cdiff + deltai(j)*invq(i,j)
06614                 end do
06615                 !correct the coefficients
06616                 q%alpha(i) =  q%alpha(i) - cdiff
06617             else !points that are not present MUST have a coefficient of 0
06618                 q%alpha(i) = 0d0
06619             end if
06620         end do
06621         call kq%free() !free temporary 1d kernel
06622     end if
06623
06624     !deallocate memory again
06625     do i = 1,q%Ndim
06626         if(allocated(km(i)%M))    deallocate(km(i)%M)
06627     end do
06628     return
06629
```

#### 3.24.2.4 subroutine rkhs::calculate_coefficients_slow ( class(kernel) *Q*, real(kind(0d0)), intent(in), optional *alpha*, logical, intent(in), optional *calcInverse* ) [private]

Calculates the kernel coefficients using the naive way (direct inversion of full matrix)

**Author**

Oliver T. Unke, University of Basel

Note that this method to calculate the kernel coefficients should never be used and instead, calculate_coefficients_fast should be called. The only exception is if the matrix is ill-conditioned, in which case the Tikhonov regularization procedure needs to be used, which is only possible with the slow method. Note that this subroutine also allows the computation of the inverse of the kernel matrix by providing the additional argument calcInverse = .true. This is needed for calculating error bounds if so desired (for large matrices, the calculation of the inverse matrix is exceedingly slow).

Definition at line 6295 of file RKHS.f90.

```
06295     implicit none
06296     class(kernel)                          :: q
06297     real(kind(0d0)), dimension(&
        count(Q%valueIsPresent),&
        count(Q%valueIsPresent))           :: km, invkm
06298     real(kind(0d0)), dimension(&
        count(Q%valueIsPresent),&
        Q%Ndim)                            :: fullgrid !matrix that stores the complete grid
06299     real(kind(0d0)), dimension(&
        count(Q%valueIsPresent))           :: coefficients, functionvalues
06300     real(kind(0d0)), optional, intent(in) :: alpha !regularization parameter
06301     integer, dimension(Q%Ndim)             :: dimindx !only needed for the traditional implementation
06302     integer                                :: i,j,k,counter,counter2,npoint, alloc_status
06303     logical, optional, intent(in)          :: calcinverse !should the inverse be computed?
06304
06305     !Npoint here stores the number of existing points
06306     npoint = count(q%valueIsPresent)
06307
06308     !build the full grid
06309     counter  = 0 !keeps track of where we are in the total grid
06310     counter2 = 0 !keeps track of at which existing point we are
06311     dimindx = 1 !set dimindx = 1
06312     do while(dimindx(1) <= size(q%grid(1)%x))
06313         counter = counter + 1
06314         if(q%valueIsPresent(counter)) then
06315             counter2 = counter2 + 1
06316             do i = 1,q%Ndim
06317                 fullgrid(counter2,i) = q%grid(i)%x(dimindx(i))
06318             end do
06319         end if
06320
06321         !increase dimindx
06322         dimindx(q%Ndim) = dimindx(q%Ndim) + 1
06323         do i = q%Ndim,2,-1
06324             if(dimindx(i) > size(q%grid(i)%x)) then
06325                 dimindx(i)   = 1
06326                 dimindx(i-1) = dimindx(i-1) + 1
06327             end if
06328         end do
06329     end do
06330
06331     !build full kernel matrix
06332     do i = 1,npoint
06333         do j = 1,i
06334             km(i,j) = 1d0
06335             !loop over all dimensions
06336             do k = 1,q%Ndim
06337                 km(i,j) = km(i,j)*q%k1d(k)%k(fullgrid(i,k),fullgrid(j,k),q%k1d(k)%par)
06338             end do
06339         end do
06340     end do
06341
06342     !add the regularization parameter
06343     if(present(alpha)) then
06344         do i = 1,npoint
06345             km(i,i) = km(i,i) + alpha
06346         end do
06347     end if
06348
06349     !do cholesky decomposition and solve for coefficients
06350     functionvalues = pack(q%values,q%valueIsPresent) !only existing functionvalues
06351     call cholesky_decomposition(km)
06352     call cholesky_solve(km,coefficients,functionvalues)
06353
06354     !store the coefficients
06355     counter = 0
06356     do i = 1,q%Npoint
06357         if(q%valueIsPresent(i)) then
06358             counter = counter + 1
06359             q%alpha(i) = coefficients(counter)
```

```
06360         else
06361             q%alpha(i) = 0d0
06362         end if
06363     end do
06364
06365     !also calculate the inverse of the kernel matrix
06366     !and store it (for calculating error bounds)
06367     if(.not.present(calcinverse)) return
06368     if(.not.calcinverse) return
06369
06370     if(.not.allocated(q%invQ)) then
06371         allocate(q%invQ(q%Npoint,q%Npoint),stat=alloc_status)
06372         if(alloc_status /= 0) call throw_error("calculate_coefficients_slow","could not allocate memory.")
06373         call cholesky_inverse(km,invkm)
06374         counter = 0
06375         do i = 1,q%Npoint
06376             if(q%valueIsPresent(i)) then
06377                 counter = counter + 1
06378                 counter2 = 0
06379                 do j = 1,i
06380                    if(q%valueIsPresent(j)) then
06381                        counter2 = counter2 + 1
06382                        q%invQ(i,j) = invkm(counter,counter2)
06383                        q%invQ(j,i) = q%invQ(i,j)
06384                    else
06385                        q%invQ(i,j) = 0d0
06386                    end if
06387                 end do
06388             else
06389                 q%invQ(i,:) = 0d0
06390             end if
06391         end do
06392     end if
06393     return
06394
```

#### 3.24.2.5 subroutine rkhs::calculate_error_bound ( class(kernel) *Q*, real(kind(0d0)), dimension(:), intent(in) *x*, real(kind(0d0)), intent(out) *errorbound* ) [private]

Calculates the error bound of the kernel at position x.

**Author**

Oliver T. Unke, University of Basel

The calculation of the error bound is rather expensive, but it can be useful in order to check the quality of the interpolation (or where it would be useful to introduce more points). This is only possible if coefficients were calculated using the slow method, as the full inverse of the kernel matrix is needed for this (the reason the fast method to calculate coefficients is fast is precisely because the inverse matrix is never computed!)

Definition at line 7438 of file RKHS.f90.

```
07438     implicit none
07439     class(kernel)                               :: q
07440     real(kind(0d0)), dimension(:), intent(in) :: x !point at which to evaluate
07441     real(kind(0d0)), intent(out)              :: errorbound
07442     integer, dimension(size(x))               :: dimindx
07443     real(kind(0d0))                           :: prod
07444     real(kind(0d0))                           :: qxx, cardinal, f !kernel function at point xx
07445     real(kind(0d0)), dimension(Q%Npoint)      :: qvec !vector that contains all xix
07446     integer                                   :: i, j, counter
07447
07448
07449     if(allocated(q%invQ)) then
07450         !loop over all possible grid points to construct Qvec
07451         counter = 0
07452         dimindx = 1 !set dimindx = 1
07453         do while(dimindx(1) <= size(q%grid(1)%x))
07454             counter = counter + 1
07455             if(q%valueIsPresent(counter)) then
07456                 qvec(counter) = 1d0
07457                 do i = 1,q%Ndim
07458                     qvec(counter) = qvec(counter)*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07459                 end do
07460             else
07461                 qvec(counter) = 0d0
07462             end if
07463
07464             !increase dimindx
```

```
07465             dimindx(q%Ndim) = dimindx(q%Ndim) + 1
07466             do i = q%Ndim,2,-1
07467                 if(dimindx(i) > size(q%grid(i)%x)) then
07468                     dimindx(i)   = 1
07469                     dimindx(i-1) = dimindx(i-1) + 1
07470                 end if
07471             end do
07472         end do
07473
07474         !calculate Qxx
07475         qxx = 1d0
07476         do i = 1,q%Ndim
07477             qxx = qxx * q%k1d(i)%k(x(i),x(i),q%k1d(i)%par)
07478         end do
07479
07480         !calculate f
07481         f = sqrt(sum(pack(q%alpha,q%valueIsPresent)*pack(q%values,q%valueIsPresent)))
07482
07483         errorbound = 0d0
07484         do i = 1,q%Ndim
07485             if(.not.q%valueIsPresent(i)) cycle
07486             !calculate the regularized cardinal function
07487             !cardinal = sum(Q%invQ(i,:)*Qvec)
07488             cardinal = sum(pack(q%invQ(i,:),q%valueIsPresent)*pack(qvec,q%valueIsPresent))
07489             !calculate the sum of Qvec times cardinal function
07490             errorbound = errorbound + sum(cardinal*qvec)
07491         end do
07492         errorbound = abs(qxx - errorbound)*f
07493         return
07494     else
07495         qxx = -huge(0d0)
07496         errorbound = sqrt(qxx) !return NaN on purpose
07497         return
07498     end if
07499     return
```

#### 3.24.2.6 subroutine rkhs::calculate_single_sum ( type(kernel) *Q,* real(kind(0d0)), dimension(q%npoint), intent(out) *sigma,* integer, dimension(q%ndim), intent(in) *m,* integer, dimension(q%ndim), intent(in) *k* ) [private]

Computes the presummed values needed for the fast kernel evaluation for a single point.

**Author**

Oliver T. Unke, University of Basel

This subroutine calculates all sums for a given combination of m and k needed for the fast evaluation of a kernel. Note that the sums are calculated using a recurrence relation and the complexity of computing all sums therefore is O(Npoint^2) (naive implementaion).

Definition at line 6650 of file RKHS.f90.

```
06650     implicit none
06651     type(kernel)                                        :: q
06652     real(kind(0d0)), dimension(Q%Npoint), intent(out) :: sigma
06653     integer, dimension(Q%Ndim), intent(in)             :: m,k
06654     type(f_ptr), dimension(Q%Ndim)                      :: fmk !array of function pointers
06655     real(kind(0d0)), dimension(Q%Ndim)                  :: p2k !array of p values
06656     real(kind(0d0))                                     :: prod, sjk  !intermediate product and intermediate
      sum
06657     integer, dimension(Q%Ndim)                          :: z, z_start, z_end, z_incr !keeps track of z values
06658     integer, dimension(Q%Ndim)                          :: zvec, ivec !index vectors
06659     integer, dimension(Q%Ndim)                          :: loop_end, loop_idx
06660     integer                                             :: d,idx,j,pos1,pos2,pos3,counter
06661     logical                                             :: sum_exists
06662
06663     sigma = 0d0 !set everything to 0
06664
06665     !build the array of function pointers and p values for this combination of m and k
06666     !and also initialize loop_start and loop_end  for each dimension (depends on m),
06667     !as well as decide how to loop over the z values (also depends on m)
06668     !NOTE: m = 0 encodes f2 functions, m=1 encodes f3 functions
06669     do d = 1,q%Ndim
06670         p2k(d) = q%k1d(d)%p2(k(d)) !p2k is independent of m
06671         if(m(d) == 0) then !this is like m=2 in the paper (f2k functions)
06672             fmk(d)%f => q%k1d(d)%f2(k(d))%f !assign to f2k function
06673             z_start(d) = 0 !loop should start with z(d) = 0
06674             z_end(d)   = size(q%grid(d)%x) !loop should end with z(d) = Nd
06675             z_incr(d)  = 1
06676         else               !this is like m=3 in the paper (f3k functions)
```

```
06677             fmk(d)%f => q%k1d(d)%f3(k(d))%f !assign to f3k function
06678             z_start(d) = size(q%grid(d)%x) !loop should start with z(d) = Nd
06679             z_end(d)   = 0 !loop should end with z(d) = 0
06680             z_incr(d)  = -1
06681         end if
06682         loop_end(d) = size(q%grid(d)%x) !loop end is always N
06683     end do
06684
06685     !initialize z values
06686     z = z_start
06687
06688     !loop over all possible grid points
06689     do while(z(1) /= (z_end(1) + z_incr(1)))
06690         idx        = q%get_lookup_idx(z)
06691         sigma(idx) = 0d0 !initialize sum to 0
06692
06693         !assign appropriate loop_idx, depending on z and m
06694         do d = 1,q%Ndim
06695             if(m(d) == 0) then !this is like m=2 in the paper (f2k functions)
06696                 loop_idx(d) = z(d)
06697             else !this is like m=3 in the paper (f3k functions)
06698                 loop_idx(d) = z(d)+1
06699             end if
06700         end do
06701
06702         !initialize to the value at the current point (if meaningful, else it stays at 0)
06703         if(.not.any(loop_idx > loop_end) .and. .not.any(loop_idx < 1)) then
06704             prod = 1d0 !initialize product
06705             do d = 1,q%Ndim
06706                 prod = prod * p2k(d)*fmk(d)%f(q%grid(d)%x(loop_idx(d)),q%k1d(d)%par)
06707             end do
06708             sigma(idx) = sigma(idx) + q%alpha(q%get_alpha_idx(loop_idx)) * prod !add to sum
06709         end if
06710
06711         !loop over all possible values of j
06712         do j = 1,q%Ndim
06713             !initialize intermediate sum to 0
06714             sjk = 0d0
06715
06716             !loop over the set of vectors containing j ones and Ndim-j zeros
06717             counter = 0
06718             pos1 = 1
06719             pos2 = pos1+j-1
06720             pos3 = pos2+1
06721             ivec = 0
06722             ivec(pos1:pos2) = 1
06723             do
06724                 !build the z vector (depends on what values of m there are)
06725                 sum_exists = .true.
06726                 do d = 1,q%Ndim
06727                     if(m(d) == 0) then !m=2, ivec gets subtracted
06728                         zvec(d) = z(d) - ivec(d)
06729                         if(zvec(d) < 0) then
06730                             sum_exists = .false.
06731                             exit
06732                         end if
06733                     else !m=3, ivec gets added
06734                         zvec(d) = z(d) + ivec(d)
06735                         if(zvec(d) > size(q%grid(d)%x)) then
06736                             sum_exists = .false.
06737                             exit
06738                         end if
06739                     end if
06740                 end do
06741                 !add to Sjk (if that value even exists)
06742                 if(sum_exists) then
06743                     sjk = sjk + sigma(q%get_lookup_idx(zvec))
06744                 end if
06745
06746                 !everything below is just to make sure that every possible
06747                 !ivec is exactly used once!
06748                 if(.not.any(ivec(q%Ndim-j+1:q%Ndim) /= 1)) exit
06749
06750                 if(pos3 <= q%Ndim-counter) then
06751                     ivec(pos3)   = 1
06752                     ivec(pos3-1) = 0
06753                     pos3         = pos3 + 1
06754                 else
06755                     counter = counter + 1
06756                     pos2       = pos2-1
06757                     ivec(pos2) = 0
06758                     pos3       = pos2+1
06759
06760                     if(pos3 <= q%Ndim-counter) then
06761                         ivec(pos3)   = 1
06762                         ivec(pos3-1) = 0
06763                         pos3         = pos3 + 1
```

```
06764                   end if
06765               end if
06766
06767               if(pos2 <= pos1) then
06768                   counter = 0
06769                   pos1 = pos1+1
06770                   pos2 = pos1+j-1
06771                   pos3 = pos2+1
06772                   ivec = 0
06773                   ivec(pos1:pos2) = 1
06774               end if
06775           end do
06776
06777           if(mod(j,2) == 1) then
06778               sigma(idx) = sigma(idx) + sjk
06779           else
06780               sigma(idx) = sigma(idx) - sjk
06781           end if
06782       end do
06783
06784       !increase z successively (loops over all possible grid points this way)
06785       z(q%Ndim) = z(q%Ndim) + z_incr(q%Ndim)
06786       do d = q%Ndim,2,-1
06787           if(z(d) == (z_end(d) + z_incr(d))) then
06788               z(d)   = z_start(d)
06789               z(d-1) = z(d-1) + z_incr(d-1)
06790           end if
06791       end do
06792   end do
06793   return
```

#### 3.24.2.7 subroutine rkhs::calculate_sums ( class(kernel) *Q* ) [private]

Computes the presummed values needed for the fast kernel evaluation.

**Author**

Oliver T. Unke, University of Basel

This subroutine calculates all sums needed for the fast evaluation of a kernel. Note that the sums are calculated using a recurrence relation and the complexity of computing all sums therefore is O(Npoint) instead of O(Npoint^2) (naive implementaion).

Definition at line 6850 of file RKHS.f90.

```
06850   implicit none
06851   class(kernel)              :: q
06852   integer, dimension(Q%Ndim) :: m,k
06853   integer                    :: d, idx
06854
06855   call allocate_sigma_gamma(q)
06856
06857   !loop over all possible combinations of m and k
06858   m = 0
06859   k = 1
06860   do while(k(1) <= q%k1d(1)%M2)
06861       m = 0
06862       do while(m(1) <= 1)
06863           !find appropriate index for this combination of m and k
06864           idx = q%get_sig_idx(q%Ndim,m,k)
06865
06866           !do actual summation for this combination of m and k
06867           call calculate_single_sum(q,q%sigma(idx,:),m,k)
06868
06869           !increase m
06870           m(q%Ndim) = m(q%Ndim) + 1
06871           do d = q%Ndim,2,-1
06872               if(m(d) > 1) then
06873                   m(d)   = 0
06874                   m(d-1) = m(d-1) + 1
06875               end if
06876           end do
06877       end do
06878       !increase k
06879       k(q%Ndim) = k(q%Ndim) + 1
06880       do d = q%Ndim,2,-1
06881           if(k(d) > q%k1d(d)%M2) then
06882               k(d)   = 1
06883               k(d-1) = k(d-1) + 1
06884           end if
```

```
06885         end do
06886     end do
06887     return
```

#### 3.24.2.8 subroutine rkhs::deallocate_kernel ( class(kernel) *Q* ) [private]

Frees kernel resources (deallocates memory)

**Author**

Oliver T. Unke, University of Basel

Definition at line 6007 of file RKHS.f90.

```
06007     implicit none
06008     class(kernel) :: q
06009     integer       :: i
06010
06011     if(allocated(q%k1d)) then
06012         do i = 1,size(q%k1d)
06013             if(allocated(q%k1d(i)%p2))   deallocate(q%k1d(i)%p2)
06014             if(allocated(q%k1d(i)%f2))   deallocate(q%k1d(i)%f2)
06015             if(allocated(q%k1d(i)%f3))   deallocate(q%k1d(i)%f3)
06016             if(allocated(q%k1d(i)%df2))  deallocate(q%k1d(i)%df2)
06017             if(allocated(q%k1d(i)%df3))  deallocate(q%k1d(i)%df3)
06018             if(allocated(q%k1d(i)%d2f2)) deallocate(q%k1d(i)%d2f2)
06019             if(allocated(q%k1d(i)%d2f3)) deallocate(q%k1d(i)%d2f3)
06020             if(allocated(q%k1d(i)%par))  deallocate(q%k1d(i)%par)
06021         end do
06022         deallocate(q%k1d)
06023     end if
06024     if(allocated(q%grid)) then
06025         do i = 1,size(q%grid)
06026             if(allocated(q%grid(i)%x)) deallocate(q%grid(i)%x)
06027         end do
06028         deallocate(q%grid)
06029     end if
06030     if(allocated(q%values))         deallocate(q%values)
06031     if(allocated(q%valueIsPresent)) deallocate(q%valueIsPresent)
06032     if(allocated(q%alpha))          deallocate(q%alpha)
06033     if(allocated(q%invQ))           deallocate(q%invQ)
06034     if(allocated(q%sigma))          deallocate(q%sigma)
06035     if(allocated(q%gamma_old))      deallocate(q%gamma_old)
06036     if(allocated(q%gamma_new))      deallocate(q%gamma_new)
06037     return
```

#### 3.24.2.9 subroutine rkhs::evaluate_kernel_fast ( class(kernel) *Q*, real(kind(0d0)), dimension(:), intent(in) *x*, real(kind(0d0)), intent(out) *f*, real(kind(0d0)), dimension(size(x)), intent(out), optional *df* ) [private]

Evaluates the kernel using the fast algorithm and returns the function value (and derivatives).

**Author**

Oliver T. Unke, University of Basel

This method should always be preferred over evaluate_kernel_slow, unless the memory requirement for the lookup table is larger than the available RAM. In that case, the only way to evaluate the kernel is to use the slow algorithm. For very small grids, the slow evaluation might actually be slightly faster (this should be benchmarked in performance critical applications).

Definition at line 5498 of file RKHS_preHessian_backup.f90.

```
05498     implicit none
05499     class(kernel)                                          :: q
05500     real(kind(0d0)), dimension(:), intent(in)              :: x !point at which to evaluate
05501     real(kind(0d0)), intent(out)                           :: f !function value at the evaluation point
05502     real(kind(0d0)), dimension(size(x)), intent(out), optional :: df !partial derivatives at that point
05503     integer, dimension(Q%Ndim)                             :: m,k
05504     integer, dimension(Q%Ndim)                             :: z !index array
05505     integer                                                :: deriv,d,i,kp1,new_idx,old_idx
05506
05507     !find appropriate indices to initialize gamma from lookup table
```

```
05508     do d = 1,q%Ndim
05509         z(d) = binary_search(q%grid(d)%x,x(d))
05510     end do
05511
05512     !initialize to sigma from lookup table
05513     q%gamma_new = q%sigma(:,q%get_lookup_idx(z))
05514     q%gamma_old = q%gamma_new
05515
05516     !iteratively update gamma values
05517     do d = q%Ndim-1,1,-1 !iteratively update gamma values until arriving at dimension 1
05518         !loop over all possible k and m values
05519         k = 1
05520         do while(k(1) <= q%k1d(1)%M2)
05521             m = 0
05522             do while(m(1) <= 1)
05523                 new_idx = q%get_sig_idx(d,m(1:d),k(1:d)) !get the appropriate index for the gamma_new array
05524                 q%gamma_new(new_idx) = 0d0 !initialize to 0
05525
05526                 !loop over possible k values in dimension d+1
05527                 do kp1 = 1,q%k1d(d+1)%M2
05528                     k(d+1) = kp1
05529
05530                     !first, calculate the f3k part (uses gamma2k)
05531                     m(d+1) = 0
05532                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05533                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05534                                          + q%gamma_old(old_idx) &
05535                                          * q%k1d(d+1)%f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
05536
05537                     !second, calculate the f2k part (uses gamma3k)
05538                     m(d+1) = 1
05539                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05540                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05541                                          + q%gamma_old(old_idx) &
05542                                          * q%k1d(d+1)%f2(kp1)%f(x(d+1),q%k1d(d+1)%par)

05543                 end do
05544
05545                 !increment m
05546                 m(d) = m(d) + 1
05547                 do i = d,2,-1
05548                     if(m(i) > 1) then
05549                         m(i)   = 0
05550                         m(i-1) = m(i-1) + 1
05551                     end if
05552                 end do
05553             end do
05554             !increment k
05555             k(d) = k(d) + 1
05556             do i = d,2,-1
05557                 if(k(i) > q%k1d(i)%M2) then
05558                     k(i)   = 1
05559                     k(i-1) = k(i-1) + 1
05560                 end if
05561             end do
05562         end do
05563
05564         !update old gamma for next iteration, but only the values that are actually used
05565         kp1 = 2**d*product(q%k1d(1:d)%M2)
05566         q%gamma_old(1:kp1) = q%gamma_new(1:kp1)
05567     end do
05568
05569     !Now, finally, the gamma value is updated and the energy can be computed
05570     f = 0d0 !initialize to 0
05571     !loop over possible k values in dimension 1
05572     do kp1 = 1,q%k1d(1)%M2
05573         !first, calculate the f3k part (uses gamma2k)
05574         new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
05575         f = f + q%gamma_new(new_idx) &
05576               * q%k1d(1)%f3(kp1)%f(x(1),q%k1d(1)%par)
05577
05578         !second, calculate the f2k part (uses gamma3k)
05579         new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
05580         f = f + q%gamma_new(new_idx) &
05581               * q%k1d(1)%f2(kp1)%f(x(1),q%k1d(1)%par)
05582     end do
05583
05584     !also compute partial derivatives if needed
05585     if(present(df)) then
05586         !loop over all possible partial derivatives
05587         do deriv = 1,q%Ndim
05588             q%gamma_old = q%sigma(:,q%get_lookup_idx(z)) !initialize to sigma from lookup table
05589             !iteratively update gamma values
05590             do d = q%Ndim-1,1,-1 !iteratively update gamma values until arriving at dimension 1
05591                 !loop over all possible k and m values
05592                 k = 1
05593                 do while(k(1) <= q%k1d(1)%M2)
```

```
05594                     m = 0
05595                     do while(m(1) <= 1)
05596                         new_idx = q%get_sig_idx(d,m(1:d),k(1:d)) !get the appropriate index for the
      gamma_new array
05597                         q%gamma_new(new_idx) = 0d0 !initialize to 0
05598
05599                         !loop over possible k values in dimension d+1
05600                         do kp1 = 1,q%k1d(d+1)%M2
05601                             k(d+1) = kp1
05602
05603                             if(deriv == d+1) then
05604                                 !first, calculate the f3k part (uses gamma2k)
05605                                 m(d+1) = 0
05606                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05607                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05608                                                      + q%gamma_old(old_idx) &
05609                                                      * q%k1d(d+1)%df3(kp1)%f(x(d+1),q%k1d(d+1)%par)
05610
05611                                 !second, calculate the f2k part (uses gamma3k)
05612                                 m(d+1) = 1
05613                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05614                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05615                                                      + q%gamma_old(old_idx) &
05616                                                      * q%k1d(d+1)%df2(kp1)%f(x(d+1),q%k1d(d+1)%par)
05617                             else
05618                                 !first, calculate the f3k part (uses gamma2k)
05619                                 m(d+1) = 0
05620                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05621                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05622                                                      + q%gamma_old(old_idx) &
05623                                                      * q%k1d(d+1)%f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
05624
05625                                 !second, calculate the f2k part (uses gamma3k)
05626                                 m(d+1) = 1
05627                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
05628                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
05629                                                      + q%gamma_old(old_idx) &
05630                                                      * q%k1d(d+1)%f2(kp1)%f(x(d+1),q%k1d(d+1)%par)
05631                             end if
05632                         end do
05633
05634                         !increment m
05635                         m(d) = m(d) + 1
05636                         do i = d,2,-1
05637                             if(m(i) > 1) then
05638                                 m(i)   = 0
05639                                 m(i-1) = m(i-1) + 1
05640                             end if
05641                         end do
05642                     end do
05643                     !increment k
05644                     k(d) = k(d) + 1
05645                     do i = d,2,-1
05646                         if(k(i) > q%k1d(i)%M2) then
05647                             k(i)   = 1
05648                             k(i-1) = k(i-1) + 1
05649                         end if
05650                     end do
05651                 end do
05652
05653                 !update old gamma for next iteration, but only the values that are actually used
05654                 kp1 = 2**d*product(q%k1d(1:d)%M2)
05655                 q%gamma_old(1:kp1) = q%gamma_new(1:kp1)
05656             end do
05657
05658             !Now, finally, the gamma value is updated and the energy can be computed
05659             df(deriv) = 0d0 !initialize to 0
05660             !loop over possible k values in dimension 1
05661             do kp1 = 1,q%k1d(1)%M2
05662                 if(deriv == 1) then
05663                     !first, calculate the f3k part (uses gamma2k)
05664                     new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
05665                     df(deriv) = df(deriv) + q%gamma_new(new_idx) &
05666                                           * q%k1d(1)%df3(kp1)%f(x(1),q%k1d(1)%par)
05667
05668                     !second, calculate the f2k part (uses gamma3k)
05669                     new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
05670                     df(deriv) = df(deriv) + q%gamma_new(new_idx) &
05671                                           * q%k1d(1)%df2(kp1)%f(x(1),q%k1d(1)%par)
05672                 else
05673                     !first, calculate the f3k part (uses gamma2k)
05674                     new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
05675                     df(deriv) = df(deriv) + q%gamma_new(new_idx) &
05676                                           * q%k1d(1)%f3(kp1)%f(x(1),q%k1d(1)%par)
05677
05678                     !second, calculate the f2k part (uses gamma3k)
05679                     new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
```

```
05680                     df(deriv) = df(deriv) + q%gamma_new(new_idx) &
05681                                           * q%k1d(1)%f2(kp1)%f(x(1),q%k1d(1)%par)
05682                 end if
05683             end do
05684         end do
05685     end if
05686     return
```

#### 3.24.2.10 subroutine rkhs::evaluate_kernel_fast ( class(kernel) *Q,* real(kind(0d0)), dimension(:), intent(in) *x,* real(kind(0d0)), intent(out) *f,* real(kind(0d0)), dimension(size(x)), intent(out), optional *df,* real(kind(0d0)), dimension(size(x),size(x)), intent(out), optional *d2f,* logical, dimension(size(x)), intent(in), optional *df_mask,* logical, dimension(size(x),size(x)), intent(in), optional *d2f_mask* ) [private]

Evaluates the kernel using the fast algorithm and returns the function value (and derivatives).

**Author**

Oliver T. Unke, University of Basel

This method should always be preferred over evaluate_kernel_slow, unless the memory requirement for the lookup table is larger than the available RAM. In that case, the only way to evaluate the kernel is to use the slow algorithm. For very small grids, the slow evaluation might actually be slightly faster (this should be benchmarked in performance critical applications).

Definition at line 6944 of file RKHS.f90.

```
06944     implicit none
06945     class(kernel)                                                    :: q
06946     real(kind(0d0)), dimension(:), intent(in)                        :: x !point at which to evaluate
06947     real(kind(0d0)), intent(out)                                     :: f !function value at the
      evaluation point
06948     real(kind(0d0)), dimension(size(x)), intent(out), optional       :: df !partial derivatives at that
      point
06949     real(kind(0d0)), dimension(size(x),size(x)), intent(out), optional :: d2f !Hessian at that point
06950     logical,         dimension(size(x)), intent(in),  optional       :: df_mask !for calculating only
      some derivatives
06951     logical,         dimension(size(x),size(x)), intent(in),  optional :: d2f_mask !for calculating only
      some entries
06952     integer, dimension(Q%Ndim)                                       :: m,k
06953     integer, dimension(Q%Ndim)                                       :: z !index array
06954     integer                                                          :: deriv,deriv2,d,i,kp1,new_idx,
     old_idx
06955
06956     !find appropriate indices to initialize gamma from lookup table
06957     do d = 1,q%Ndim
06958         z(d) = binary_search(q%grid(d)%x,x(d))
06959     end do
06960
06961     !initialize to sigma from lookup table
06962     q%gamma_new = q%sigma(:,q%get_lookup_idx(z))
06963     q%gamma_old = q%gamma_new
06964
06965     !iteratively update gamma values
06966     do d = q%Ndim-1,1,-1 !iteratively update gamma values until arriving at dimension 1
06967         !loop over all possible k and m values
06968         k = 1
06969         do while(k(1) <= q%k1d(1)%M2)
06970             m = 0
06971             do while(m(1) <= 1)
06972                 new_idx = q%get_sig_idx(d,m(1:d),k(1:d)) !get the appropriate index for the gamma_new array
06973                 q%gamma_new(new_idx) = 0d0 !initialize to 0
06974
06975                 !loop over possible k values in dimension d+1
06976                 do kp1 = 1,q%k1d(d+1)%M2
06977                     k(d+1) = kp1
06978
06979                     !first, calculate the f3k part (uses gamma2k)
06980                     m(d+1) = 0
06981                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
06982                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
06983                                          + q%gamma_old(old_idx) &
06984                                          * q%k1d(d+1)%f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
06985
06986                     !second, calculate the f2k part (uses gamma3k)
06987                     m(d+1) = 1
06988                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
06989                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
06990                                          + q%gamma_old(old_idx) &
```

```
06991                                       * q%k1d(d+1)%f2(kp1)%f(x(d+1),q%k1d(d+1)%par)

06992                 end do
06993
06994                 !increment m
06995                 m(d) = m(d) + 1
06996                 do i = d,2,-1
06997                     if(m(i) > 1) then
06998                         m(i)   = 0
06999                         m(i-1) = m(i-1) + 1
07000                     end if
07001                 end do
07002             end do
07003             !increment k
07004             k(d) = k(d) + 1
07005             do i = d,2,-1
07006                 if(k(i) > q%k1d(i)%M2) then
07007                     k(i)   = 1
07008                     k(i-1) = k(i-1) + 1
07009                 end if
07010             end do
07011         end do
07012
07013         !update old gamma for next iteration, but only the values that are actually used
07014         kp1 = 2**d*product(q%k1d(1:d)%M2)
07015         q%gamma_old(1:kp1) = q%gamma_new(1:kp1)
07016     end do
07017
07018     !Now, finally, the gamma value is updated and the energy can be computed
07019     f = 0d0 !initialize to 0
07020     !loop over possible k values in dimension 1
07021     do kp1 = 1,q%k1d(1)%M2
07022         !first, calculate the f3k part (uses gamma2k)
07023         new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07024         f = f + q%gamma_new(new_idx) &
07025               * q%k1d(1)%f3(kp1)%f(x(1),q%k1d(1)%par)
07026
07027         !second, calculate the f2k part (uses gamma3k)
07028         new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07029         f = f + q%gamma_new(new_idx) &
07030               * q%k1d(1)%f2(kp1)%f(x(1),q%k1d(1)%par)
07031     end do
07032
07033     !also compute partial derivatives if needed
07034     if(present(df)) then
07035         !loop over all possible partial derivatives
07036         do deriv = 1,q%Ndim
07037             !skip values if a mask is provided
07038             if(present(df_mask)) then
07039                 if(.not.df_mask(deriv)) cycle
07040             end if
07041             q%gamma_old = q%sigma(:,q%get_lookup_idx(z)) !initialize to sigma from lookup table
07042             !iteratively update gamma values
07043             do d = q%Ndim-1,1,-1 !iteratively update gamma values until arriving at dimension 1
07044                 !loop over all possible k and m values
07045                 k = 1
07046                 do while(k(1) <= q%k1d(1)%M2)
07047                     m = 0
07048                     do while(m(1) <= 1)
07049                         new_idx = q%get_sig_idx(d,m(1:d),k(1:d)) !get the appropriate index for the
    gamma_new array
07050                         q%gamma_new(new_idx) = 0d0 !initialize to 0
07051
07052                         !loop over possible k values in dimension d+1
07053                         do kp1 = 1,q%k1d(d+1)%M2
07054                             k(d+1) = kp1
07055
07056                             if(deriv == d+1) then
07057                                 !first, calculate the f3k part (uses gamma2k)
07058                                 m(d+1) = 0
07059                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07060                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07061                                                      + q%gamma_old(old_idx) &
07062                                                      * q%k1d(d+1)%df3(kp1)%f(x(d+1),q%k1d(d+1)%par)
07063
07064                                 !second, calculate the f2k part (uses gamma3k)
07065                                 m(d+1) = 1
07066                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07067                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07068                                                      + q%gamma_old(old_idx) &
07069                                                      * q%k1d(d+1)%df2(kp1)%f(x(d+1),q%k1d(d+1)%par)
07070                             else
07071                                 !first, calculate the f3k part (uses gamma2k)
07072                                 m(d+1) = 0
07073                                 old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07074                                 q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07075                                                      + q%gamma_old(old_idx) &
```

```
07076                                                   * q%k1d(d+1)%f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
07077
07078                                   !second, calculate the f2k part (uses gamma3k)
07079                                   m(d+1) = 1
07080                                   old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07081                                   q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07082                                                       + q%gamma_old(old_idx) &
07083                                                       * q%k1d(d+1)%f2(kp1)%f(x(d+1),q%k1d(d+1)%par)
07084                               end if
07085                           end do
07086
07087                           !increment m
07088                           m(d) = m(d) + 1
07089                           do i = d,2,-1
07090                               if(m(i) > 1) then
07091                                   m(i)   = 0
07092                                   m(i-1) = m(i-1) + 1
07093                               end if
07094                           end do
07095                       end do
07096                       !increment k
07097                       k(d) = k(d) + 1
07098                       do i = d,2,-1
07099                           if(k(i) > q%k1d(i)%M2) then
07100                               k(i)   = 1
07101                               k(i-1) = k(i-1) + 1
07102                           end if
07103                       end do
07104                   end do
07105
07106                   !update old gamma for next iteration, but only the values that are actually used
07107                   kp1 = 2**d*product(q%k1d(1:d)%M2)
07108                   q%gamma_old(1:kp1) = q%gamma_new(1:kp1)
07109               end do
07110
07111               !Now, finally, the gamma value is updated and the energy can be computed
07112               df(deriv) = 0d0 !initialize to 0
07113               !loop over possible k values in dimension 1
07114               do kp1 = 1,q%k1d(1)%M2
07115                   if(deriv == 1) then
07116                       !first, calculate the f3k part (uses gamma2k)
07117                       new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07118                       df(deriv) = df(deriv) + q%gamma_new(new_idx) &
07119                                             * q%k1d(1)%df3(kp1)%f(x(1),q%k1d(1)%par)
07120
07121                       !second, calculate the f2k part (uses gamma3k)
07122                       new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07123                       df(deriv) = df(deriv) + q%gamma_new(new_idx) &
07124                                             * q%k1d(1)%df2(kp1)%f(x(1),q%k1d(1)%par)
07125                   else
07126                       !first, calculate the f3k part (uses gamma2k)
07127                       new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07128                       df(deriv) = df(deriv) + q%gamma_new(new_idx) &
07129                                             * q%k1d(1)%f3(kp1)%f(x(1),q%k1d(1)%par)
07130
07131                       !second, calculate the f2k part (uses gamma3k)
07132                       new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07133                       df(deriv) = df(deriv) + q%gamma_new(new_idx) &
07134                                             * q%k1d(1)%f2(kp1)%f(x(1),q%k1d(1)%par)
07135                   end if
07136               end do
07137           end do
07138       end if
07139
07140       !also compute Hessian if needed
07141       if(present(d2f)) then
07142           !loop over all possible entries of the Hessian
07143           do deriv = 1,q%Ndim
07144               do deriv2 = 1,q%Ndim
07145                   !skip values if a mask is provided
07146                   if(present(d2f_mask)) then
07147                       if(.not.d2f_mask(deriv,deriv2)) cycle
07148                   end if
07149                   !because the Hessian is symmetric, we don't need to recalculate values
07150                   !that were already calculated
07151                   if(deriv2 > deriv) then
07152                       if(present(d2f_mask)) then
07153                           if(d2f_mask(deriv2,deriv)) then !we can only do this if the value was calculated
      before
07154                               d2f(deriv,deriv2) = d2f(deriv2,deriv)
07155                               cycle
07156                           end if
07157                       else
07158                           d2f(deriv,deriv2) = d2f(deriv2,deriv)
07159                           cycle
07160                       end if
07161                   end if
```

```
07162
07163                 q%gamma_old = q%sigma(:,q%get_lookup_idx(z)) !initialize to sigma from lookup table
07164                 !iteratively update gamma values
07165                 do d = q%Ndim-1,1,-1 !iteratively update gamma values until arriving at dimension 1
07166                     !loop over all possible k and m values
07167                     k = 1
07168                     do while(k(1) <= q%k1d(1)%M2)
07169                         m = 0
07170                         do while(m(1) <= 1)
07171                             new_idx = q%get_sig_idx(d,m(1:d),k(1:d)) !get the appropriate index for the
    gamma_new array
07172                             q%gamma_new(new_idx) = 0d0 !initialize to 0
07173
07174                             !loop over possible k values in dimension d+1
07175                             do kp1 = 1,q%k1d(d+1)%M2
07176                                 k(d+1) = kp1
07177                                 if(deriv == d+1 .and. deriv2 == d+1) then !second derivative needed
07178                                     !first, calculate the f3k part (uses gamma2k)
07179                                     m(d+1) = 0
07180                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07181                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07182                                                          + q%gamma_old(old_idx) &
07183                                                          * q%k1d(d+1)%d2f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
07184
07185                                     !second, calculate the f2k part (uses gamma3k)
07186                                     m(d+1) = 1
07187                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07188                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07189                                                          + q%gamma_old(old_idx) &
07190                                                          * q%k1d(d+1)%d2f2(kp1)%f(x(d+1),q%k1d(d+1)%par)
07191                                 else if(deriv == d+1 .or. deriv2 == d+1) then !first derivative needed
07192                                     !first, calculate the f3k part (uses gamma2k)
07193                                     m(d+1) = 0
07194                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07195                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07196                                                          + q%gamma_old(old_idx) &
07197                                                          * q%k1d(d+1)%df3(kp1)%f(x(d+1),q%k1d(d+1)%par)
07198
07199                                     !second, calculate the f2k part (uses gamma3k)
07200                                     m(d+1) = 1
07201                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07202                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07203                                                          + q%gamma_old(old_idx) &
07204                                                          * q%k1d(d+1)%df2(kp1)%f(x(d+1),q%k1d(d+1)%par)
07205                                 else
07206                                     !first, calculate the f3k part (uses gamma2k)
07207                                     m(d+1) = 0
07208                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07209                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07210                                                          + q%gamma_old(old_idx) &
07211                                                          * q%k1d(d+1)%f3(kp1)%f(x(d+1),q%k1d(d+1)%par)
07212
07213                                     !second, calculate the f2k part (uses gamma3k)
07214                                     m(d+1) = 1
07215                                     old_idx = q%get_sig_idx(d+1,m(1:d+1),k(1:d+1))
07216                                     q%gamma_new(new_idx) = q%gamma_new(new_idx) &
07217                                                          + q%gamma_old(old_idx) &
07218                                                          * q%k1d(d+1)%f2(kp1)%f(x(d+1),q%k1d(d+1)%par)
07219                                 end if
07220                             end do
07221
07222                             !increment m
07223                             m(d) = m(d) + 1
07224                             do i = d,2,-1
07225                                 if(m(i) > 1) then
07226                                     m(i)   = 0
07227                                     m(i-1) = m(i-1) + 1
07228                                 end if
07229                             end do
07230                         end do
07231                         !increment k
07232                         k(d) = k(d) + 1
07233                         do i = d,2,-1
07234                             if(k(i) > q%k1d(i)%M2) then
07235                                 k(i)   = 1
07236                                 k(i-1) = k(i-1) + 1
07237                             end if
07238                         end do
07239                     end do
07240
07241                     !update old gamma for next iteration, but only the values that are actually used
07242                     kp1 = 2**d*product(q%k1d(1:d)%M2)
07243                     q%gamma_old(1:kp1) = q%gamma_new(1:kp1)
07244                 end do
07245
07246                 !Now, finally, the gamma value is updated and the energy can be computed
07247                 d2f(deriv,deriv2) = 0d0 !initialize to 0
```

```
07248                 !loop over possible k values in dimension 1
07249                 do kp1 = 1,q%k1d(1)%M2
07250                     if(deriv == 1 .and. deriv2 == 1) then !second derivative is needed
07251                         !first, calculate the f3k part (uses gamma2k)
07252                         new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07253                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07254                                             * q%k1d(1)%d2f3(kp1)%f(x(1),q%k1d(1)%par)
07255
07256                         !second, calculate the f2k part (uses gamma3k)
07257                         new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07258                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07259                                             * q%k1d(1)%d2f2(kp1)%f(x(1),q%k1d(1)%par)
07260                     else if(deriv == 1 .or. deriv2 == 1) then !first derivative is needed
07261                         !first, calculate the f3k part (uses gamma2k)
07262                         new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07263                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07264                                             * q%k1d(1)%df3(kp1)%f(x(1),q%k1d(1)%par)
07265
07266                         !second, calculate the f2k part (uses gamma3k)
07267                         new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07268                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07269                                             * q%k1d(1)%df2(kp1)%f(x(1),q%k1d(1)%par)
07270                     else
07271                         !first, calculate the f3k part (uses gamma2k)
07272                         new_idx = q%get_sig_idx(1,(/0/),(/kp1/))
07273                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07274                                             * q%k1d(1)%f3(kp1)%f(x(1),q%k1d(1)%par)
07275
07276                         !second, calculate the f2k part (uses gamma3k)
07277                         new_idx = q%get_sig_idx(1,(/1/),(/kp1/))
07278                         d2f(deriv,deriv2) = d2f(deriv,deriv2) + q%gamma_new(new_idx) &
07279                                             * q%k1d(1)%f2(kp1)%f(x(1),q%k1d(1)%par)
07280                     end if
07281                 end do
07282             end do
07283         end do
07284     end if
07285     return
```

#### 3.24.2.11 subroutine rkhs::evaluate_kernel_slow ( class(kernel) *Q*, real(kind(0d0)), dimension(:), intent(in) *x*, real(kind(0d0)), intent(out) *f*, real(kind(0d0)), dimension(size(x)), intent(out), optional *df* ) [private]

Evaluates the kernel using the naive (slow) algorithm and returns the function value (and derivatives).

**Author**

Oliver T. Unke, University of Basel

This method should never be called and instead, evaluate_kernel_fast should be used. However, if not enough RAM is available to store the lookup table of precomputed sums, this may be the only way to evaluate the kernel at all.

Definition at line 5702 of file RKHS_preHessian_backup.f90.

```
05702     implicit none
05703     class(kernel)                                                  :: q
05704     real(kind(0d0)), dimension(:), intent(in)                      :: x !point at which to evaluate
05705     real(kind(0d0)), intent(out)                                   :: f !function value at the evaluation point
05706     real(kind(0d0)), dimension(size(x)), intent(out), optional     :: df !partial derivatives at that point
05707     integer, dimension(size(x))                                    :: dimindx
05708     real(kind(0d0))                                                :: prod
05709     integer                                                        :: i, j, counter
05710
05711     f = 0d0
05712     counter = 0
05713     dimindx = 1 !set dimindx = 1
05714     do while(dimindx(1) <= size(q%grid(1)%x))
05715         counter = counter + 1
05716         prod = 1d0
05717         do i = 1,q%Ndim
05718             prod = prod*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
05719         end do
05720
05721         f = f + q%alpha(counter)*prod
05722
05723         !increase dimindx
05724         dimindx(q%Ndim) = dimindx(q%Ndim) + 1
05725         do i = q%Ndim,2,-1
05726             if(dimindx(i) > size(q%grid(i)%x)) then
05727                 dimindx(i)   = 1
05728                 dimindx(i-1) = dimindx(i-1) + 1
```

```
05729             end if
05730         end do
05731     end do
05732
05733     if(present(df)) then
05734         do j = 1,q%Ndim
05735             df(j) = 0d0
05736             counter = 0
05737             dimindx = 1
05738             do while(dimindx(1) <= size(q%grid(1)%x))
05739                 counter = counter + 1
05740                 prod = 1d0
05741                 do i = 1,q%Ndim
05742                     if(i == j) then
05743                         prod = prod*q%k1d(i)%dk(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
05744                     else
05745                         prod = prod*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
05746                     end if
05747                 end do
05748                 df(j) = df(j) + q%alpha(counter)*prod
05749
05750                 !increase dimindx
05751                 dimindx(q%Ndim) = dimindx(q%Ndim) + 1
05752                 do i = q%Ndim,2,-1
05753                     if(dimindx(i) > size(q%grid(i)%x)) then
05754                         dimindx(i)   = 1
05755                         dimindx(i-1) = dimindx(i-1) + 1
05756                     end if
05757                 end do
05758             end do
05759         end do
05760     end if
05761     return
```

#### 3.24.2.12 subroutine rkhs::evaluate_kernel_slow ( class(kernel) *Q,* real(kind(0d0)), dimension(:), intent(in) *x,* real(kind(0d0)), intent(out) *f,* real(kind(0d0)), dimension(size(x)), intent(out), optional *df,* real(kind(0d0)), dimension(size(x),size(x)), intent(out), optional *d2f,* logical, dimension(size(x)), intent(in), optional *df_mask,* logical, dimension(size(x),size(x)), intent(in), optional *d2f_mask* ) [private]

Evaluates the kernel using the naive (slow) algorithm and returns the function value (and derivatives).

**Author**

Oliver T. Unke, University of Basel

This method should never be called and instead, evaluate_kernel_fast should be used. However, if not enough RAM is available to store the lookup table of precomputed sums, this may be the only way to evaluate the kernel at all.

Definition at line 7301 of file RKHS.f90.

```
07301     implicit none
07302     class(kernel)                                                    :: q
07303     real(kind(0d0)), dimension(:), intent(in)                        :: x !point at which to evaluate
07304     real(kind(0d0)), intent(out)                                     :: f !function value at the
      evaluation point
07305     real(kind(0d0)), dimension(size(x)), intent(out), optional       :: df !partial derivatives at that
      point
07306     real(kind(0d0)), dimension(size(x),size(x)), intent(out), optional :: d2f !Hessian at that point
07307     logical,         dimension(size(x)), intent(in),  optional       :: df_mask !for calculating only
      some derivatives
07308     logical,         dimension(size(x),size(x)), intent(in),  optional :: d2f_mask !for calculating only
      some entries
07309     integer, dimension(size(x))                                      :: dimindx
07310     real(kind(0d0))                                                  :: prod
07311     integer                                                          :: i, d, d2, counter
07312
07313     f = 0d0
07314     counter = 0
07315     dimindx = 1 !set dimindx = 1
07316     do while(dimindx(1) <= size(q%grid(1)%x))
07317         counter = counter + 1
07318         prod = 1d0
07319         do i = 1,q%Ndim
07320             prod = prod*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07321         end do
07322
07323         f = f + q%alpha(counter)*prod
07324
07325         !increase dimindx
```

```
07326         dimindx(q%Ndim) = dimindx(q%Ndim) + 1
07327         do i = q%Ndim,2,-1
07328             if(dimindx(i) > size(q%grid(i)%x)) then
07329                 dimindx(i)   = 1
07330                 dimindx(i-1) = dimindx(i-1) + 1
07331             end if
07332         end do
07333     end do
07334
07335     if(present(df)) then
07336         do d = 1,q%Ndim
07337             !skip values if a mask is provided
07338             if(present(df_mask)) then
07339                 if(.not.df_mask(d)) cycle
07340             end if
07341             df(d) = 0d0
07342             counter = 0
07343             dimindx = 1
07344             do while(dimindx(1) <= size(q%grid(1)%x))
07345                 counter = counter + 1
07346                 prod = 1d0
07347                 do i = 1,q%Ndim
07348                     if(i == d) then
07349                         prod = prod*q%k1d(i)%dk(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07350                     else
07351                         prod = prod*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07352                     end if
07353                 end do
07354                 df(d) = df(d) + q%alpha(counter)*prod
07355
07356                 !increase dimindx
07357                 dimindx(q%Ndim) = dimindx(q%Ndim) + 1
07358                 do i = q%Ndim,2,-1
07359                     if(dimindx(i) > size(q%grid(i)%x)) then
07360                         dimindx(i)   = 1
07361                         dimindx(i-1) = dimindx(i-1) + 1
07362                     end if
07363                 end do
07364             end do
07365         end do
07366     end if
07367
07368     if(present(d2f)) then
07369         do d = 1,q%Ndim
07370             do d2 = 1,q%Ndim
07371                 !skip values if a mask is provided
07372                 if(present(d2f_mask)) then
07373                     if(.not.d2f_mask(d,d2)) cycle
07374                 end if
07375                 !because the Hessian is symmetric, we don't need to recalculate values
07376                 !that were already calculated
07377                 if(d2 > d) then
07378                     if(present(d2f_mask)) then
07379                         if(d2f_mask(d2,d)) then !we can only do this if the value was calculated before
07380                             d2f(d,d2) = d2f(d2,d)
07381                             cycle
07382                         end if
07383                     else
07384                         d2f(d,d2) = d2f(d2,d)
07385                         cycle
07386                     end if
07387                 end if
07388
07389                 d2f(d,d2) = 0d0
07390                 counter = 0
07391                 dimindx = 1
07392                 do while(dimindx(1) <= size(q%grid(1)%x))
07393                     counter = counter + 1
07394                     prod = 1d0
07395                     do i = 1,q%Ndim
07396                         if (i == d .and. i == d2) then !second derivative needed
07397                             prod = prod*q%k1d(i)%d2k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07398                         else if(i == d .or. i == d2) then !first derivative needed
07399                             prod = prod*q%k1d(i)%dk(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07400                         else
07401                             prod = prod*q%k1d(i)%k(x(i),q%grid(i)%x(dimindx(i)),q%k1d(i)%par)
07402                         end if
07403                     end do
07404                     d2f(d,d2) = d2f(d,d2) + q%alpha(counter)*prod
07405
07406                     !increase dimindx
07407                     dimindx(q%Ndim) = dimindx(q%Ndim) + 1
07408                     do i = q%Ndim,2,-1
07409                         if(dimindx(i) > size(q%grid(i)%x)) then
07410                             dimindx(i)   = 1
07411                             dimindx(i-1) = dimindx(i-1) + 1
07412                         end if
```

```
07413                   end do
07414                end do
07415             end do
07416          end do
07417       end if
07418       return
```

#### 3.24.2.13 subroutine rkhs::fast_tensor_solve ( class(kernel) *Q*, type(kernel_matrix), dimension(q%ndim), intent(in) *KM*, real(kind(0d0)), dimension(size(q%values)), intent(out) *solution* ) [private]

Solves a system of linear equations in Tensor product form efficiently.

**Author**

Oliver T. Unke, University of Basel

Solution contains the desired solution to the linear equations on input and the calculated coefficients on output. The algorithm is adapted from: P. Fernandes and B. Plateau. "Triangular solution of linear systems in tensor product format." ACM SIGMETRICS Performance Evaluation Review 28.4 (2001): 30-32.

Definition at line 6416 of file RKHS.f90.

```
06416       implicit none
06417       class(kernel)                                          :: q
06418       type(kernel_matrix), dimension(Q%Ndim), intent(in)     :: km
06419       real(kind(0d0)), dimension(size(Q%values)), intent(out) :: solution
06420       real(kind(0d0)), dimension(size(Q%values))             :: zin, zout
06421       integer                                                :: i,j,k,l,base,indx,nleft,nright
06422
06423       !forward substitution
06424       !initialize nleft and nright (needed for the algorithm)
06425       nleft  = 1
06426       nright = 1
06427       do i = 1,q%Ndim
06428          nright = nright * size(q%grid(i)%x)
06429       end do
06430
06431       !loop over all dimensions
06432       do i = 1,q%Ndim
06433          !calculate new nright
06434          nright = nright/size(q%grid(i)%x)
06435
06436          base = 0
06437          do k = 1,nleft
06438             do j = 1,nright
06439                indx = base + j
06440                do l = 1,size(q%grid(i)%x)
06441                   zin(l) = solution(indx)
06442                   indx = indx + nright
06443                end do
06444                call forward_substitution(km(i)%M,zout(1:size(q%grid(i)%x)),zin(1:size(q%grid(i)%x)))
06445                indx = base + j
06446                do l = 1,size(q%grid(i)%x)
06447                   solution(indx) = zout(l)
06448                   indx = indx + nright
06449                end do
06450             end do
06451             base = base + (nright * size(q%grid(i)%x))
06452          end do
06453          !calculate new nleft
06454          nleft = nleft*size(q%grid(i)%x)
06455       end do
06456
06457       !backward substitution
06458       !initialize nleft and nright (needed for the algorithm)
06459       nleft  = 1
06460       nright = 1
06461       do i = 1,q%Ndim
06462          nright = nright * size(q%grid(i)%x)
06463       end do
06464
06465       !loop over all dimensions
06466       do i = 1,q%Ndim
06467          !calculate new nright
06468          nright = nright/size(q%grid(i)%x)
06469
06470          base = 0
06471          do k = 1,nleft
06472             do j = 1,nright
```

```
06473                 indx = base + j
06474                 do l = 1,size(q%grid(i)%x)
06475                     zin(l) = solution(indx)
06476                     indx = indx + nright
06477                 end do
06478                 call backward_substitution(transpose(km(i)%M),zout(1:size(q%grid(i)%x)),zin(1:size(q%grid(i
    )%x)))
06479                 indx = base + j
06480                 do l = 1,size(q%grid(i)%x)
06481                     solution(indx) = zout(l)
06482                     indx = indx + nright
06483                 end do
06484             end do
06485             base = base + (nright * size(q%grid(i)%x))
06486         end do
06487         !calculate new nleft
06488         nleft = nleft*size(q%grid(i)%x)
06489     end do
06490     return
```

#### 3.24.2.14 integer function rkhs::get_idx_from_indices ( class(kernel), intent(in) *Q*, integer, dimension(q%ndim), intent(in) *indices* ) [private]

Computes the index for retrieving the correct alpha coefficient.

**Author**

Oliver T. Unke, University of Basel

Given a combination of indices, the index for retrieving the correct alpha coefficient is returned. This is necessary because the coefficients are stored in a flattened 1-dimensional array.

Definition at line 6109 of file RKHS.f90.

```
06109     implicit none
06110     class(kernel), intent(in)              :: q
06111     integer, dimension(Q%Ndim), intent(in) :: indices
06112     integer                                :: idx, d, prod
06113
06114     prod = 1
06115     idx = indices(q%Ndim)
06116     do d = q%Ndim-1,1,-1
06117         prod = prod*size(q%grid(d+1)%x)
06118         idx = idx + (indices(d)-1) * prod
06119     end do
```

#### 3.24.2.15 integer function rkhs::get_sig_idx_from_indices ( class(kernel), intent(in) *Q*, integer, dimension(q%ndim), intent(in) *indices* ) [private]

Computes the index for retrieving the correct lookup table for a certain index combination.

**Author**

Oliver T. Unke, University of Basel

Given a combination of indices, the index for retrieving the correct lookup table (containing all possible combinations of m and k) is returned. This is necessary because the lookup table is stored in a flattened 1-dimensional array.

Definition at line 6084 of file RKHS.f90.

```
06084     implicit none
06085     class(kernel), intent(in)              :: q
06086     integer, dimension(Q%Ndim), intent(in) :: indices
06087     integer                                :: idx, d, prod
06088
06089     prod = 1
06090     idx = indices(q%Ndim) + 1
06091     do d = q%Ndim-1,1,-1
06092         prod = prod*(size(q%grid(d+1)%x)+1)
06093         idx = idx + indices(d) * prod
06094     end do
```

#### 3.24.2.16 integer function rkhs::get_sig_idx_from_m_and_k ( class(kernel), intent(in) *Q,* integer, intent(in) *Ndim,* integer, dimension(ndim), intent(in) *m,* integer, dimension(ndim), intent(in) *k* ) [private]

Computes the index in the lookup table of precomputed sums for m and k.

**Author**

Oliver T. Unke, University of Basel

Given a combination of values for m and k, the corresponding index in the lookup table of precomputed sums is returned. This is necessary, because the lookup table is a highly dimensional object (of unknown dimensions) and is stored in a flattened 1-dimensional array

Definition at line 6054 of file RKHS.f90.

```
06054     implicit none
06055     class(kernel), intent(in)            :: q
06056     integer, intent(in)                  :: ndim !idx in how many dimensions
06057     integer, dimension(Ndim), intent(in) :: m,k
06058     integer                              :: idx, d, prod
06059     !NOTE: m here is a vector that contains either 0 or 1, it determines
06060     !whether a f2 or a f3 function is used. A 0 means the f2 function is used
06061     !and a 1 means the f3 function is used
06062
06063     prod = 1
06064     idx  = 2*k(1)-m(1)
06065     do d = 2,ndim
06066         prod = prod*(2*q%k1d(d-1)%M2)
06067         idx = idx + (2*k(d)-m(d)-1)*prod
06068     end do
```

#### 3.24.2.17 subroutine rkhs::load_kernel_from_file ( class(kernel) *Q,* character(len=*), intent(in) *datafile* ) [private]

Loads kernel from a binary file.

**Author**

Oliver T. Unke, University of Basel

Definition at line 5905 of file RKHS.f90.

```
05905     implicit none
05906     character(len=*), intent(in) :: datafile
05907     class(kernel)                :: q
05908     integer                      :: i, ios, grid_size, kernel_type
05909     logical                      :: iswritten
05910
05911     open(io_unit, file=datafile, form="unformatted", access="stream", &
05912                           action="read", status="old", iostat = ios)
05913     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not open "'//datafile//'".')
05914
05915     call q%free() !make sure that Q is a completely fresh kernel
05916
05917     read(30, iostat = ios) q%Ndim
05918     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05919
05920     read(30, iostat = ios) q%Npoint
05921     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05922
05923     read(30, iostat = ios) iswritten
05924     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05925     if(iswritten) then
05926         allocate(q%k1d(q%Ndim))
05927         do i = 1,size(q%k1d)
05928             read(30, iostat = ios) kernel_type
05929             if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05930             call q%k1d(i)%init(kernel_type)
05931             if(q%k1d(i)%Npar > 0) then
05932                 read(30, iostat = ios) q%k1d(i)%par
05933                 if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05934             end if
05935         end do
05936     end if
05937
```

```
05938     read(30, iostat = ios) iswritten
05939     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05940     if(iswritten) then
05941         allocate(q%grid(q%Ndim))
05942         do i = 1,size(q%grid)
05943             read(30, iostat = ios) iswritten
05944             if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05945             if(iswritten) then
05946                 read(30, iostat = ios) grid_size
05947                 if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')

05948                 allocate(q%grid(i)%x(grid_size))
05949                 read(30, iostat = ios) q%grid(i)%x
05950                 if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')

05951             end if
05952         end do
05953     end if
05954
05955     read(30, iostat = ios) iswritten
05956     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05957     if(iswritten) then
05958         allocate(q%values(q%Npoint))
05959         read(30, iostat = ios) q%values
05960         if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05961     end if
05962
05963     read(30, iostat = ios) iswritten
05964     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05965     if(iswritten) then
05966         allocate(q%valueIsPresent(q%Npoint))
05967         read(30, iostat = ios) q%valueIsPresent
05968         if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05969     end if
05970
05971     read(30, iostat = ios) iswritten
05972     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05973     if(iswritten) then
05974         allocate(q%alpha(q%Npoint))
05975         read(30, iostat = ios) q%alpha
05976         if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05977     end if
05978
05979     read(30, iostat = ios) iswritten
05980     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05981     if(iswritten) then
05982         allocate(q%invQ(q%Npoint,q%Npoint))
05983         read(30, iostat = ios) q%invQ
05984         if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05985     end if
05986
05987     read(30, iostat = ios) iswritten
05988     if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05989     if(iswritten) then
05990         call allocate_sigma_gamma(q)
05991         read(30, iostat = ios) q%sigma
05992         if(ios /= 0) call throw_error("load_kernel_from_file",'Could not read "'//datafile//'".')
05993     end if
05994
05995     close(io_unit)
05996     return
```

#### 3.24.2.18 subroutine rkhs::read_grid_data ( class(kernel) *Q*, character(len=∗), intent(in) *datafile* ) [private]

Reads grid data from a .csv file and automatically allocates all necessary memory.

**Author**

Oliver T. Unke, University of Basel

Given a combination of indices, the index for retrieving the correct alpha coefficient is returned. This is necessary because the coefficients are stored in a flattened 1-dimensional array. IMPORTANT: The file must have the format x1,x2,x3,...,xn,f where the xi are the individual coordinates and f the reference values. The file needs to be sorted in this order: x1>x2>x3>...>xn

Definition at line 6146 of file RKHS.f90.

```
06146     implicit none
06147     character(len=*), intent(in)                :: datafile
```

```
06148     class(kernel)                                 :: q
06149     character(len=1)                              :: dummy
06150     integer, dimension(:), allocatable            :: dimcount !counts how many points are in each dimension
06151     real(kind(0d0)), dimension(:),   allocatable :: currvalues
06152     real(kind(0d0)), dimension(:,:), allocatable :: gridvalues
06153     integer                                       :: alloc_status, ios, i, j
06154
06155     !deallocate the kernel first
06156     call q%free()
06157
06158     !open grid datafile
06159     open(io_unit,file=datafile,status="old",action="read",iostat = ios)
06160     if(ios /= 0) call throw_error("read_grid_data", 'File "'//datafile//'" could not be opened.')
06161
06162     !determine the number of dimensions/coordinates by counting the amount of ',' in the grid datafile
06163     i = 0
06164     do while(.true.)
06165         read(30,'(A1)',advance='no',iostat = ios) dummy
06166         if(ios /= 0) exit
06167         if(dummy == ',') i = i + 1
06168     end do
06169     rewind(30) !rewind file
06170     q%Ndim = i
06171
06172     !determine the total number of gridpoints
06173     i = 0
06174     do while(.true.)
06175         read(30,*,iostat = ios) dummy
06176         if(ios /= 0) exit
06177         i = i + 1
06178     end do
06179     rewind(30)
06180     q%Npoint = i
06181
06182     !allocate the required memory to the members of Q
06183     allocate(q%k1d(q%Ndim), q%grid(q%Ndim),q%values(q%Npoint),q%alpha(q%Npoint),&
06184              q%valueIsPresent(q%Npoint),stat=alloc_status)
06185     if(alloc_status /= 0) call throw_error("read_grid_data","could not allocate memory.")
06186
06187     !allocate memory to variables used to find out grid dimensions
06188     allocate(dimcount(q%Ndim), gridvalues(q%Ndim,q%Npoint), currvalues(q%Ndim),stat=alloc_status)
06189     if(alloc_status /= 0) call throw_error("read_grid_data","could not allocate memory.")
06190     dimcount = 1
06191     !read the first line of gridvalues
06192     read(30,*,iostat=ios) gridvalues(:,1), q%values(1)
06193     q%valueIsPresent(1) = .not.isnan(q%values(1)) !detects holes
06194
06195     if(ios /= 0) call throw_error("read_grid_data", 'File "'//datafile//'" could not be read properly.')
06196     !read the remaining grid values
06197     do i = 2,q%Npoint
06198         read(30,*,iostat=ios) currvalues(:), q%values(i)
06199         q%valueIsPresent(i) = .not.isnan(q%values(i)) !detects holes
06200         if(ios /= 0) call throw_error("read_grid_data", 'File "'//datafile//'" could not be read properly.'
      )
06201         do j = 1,q%Ndim
06202             if(currvalues(j) > gridvalues(j,dimcount(j))) then
06203                 dimcount(j) = dimcount(j) + 1
06204                 gridvalues(j,dimcount(j)) = currvalues(j)
06205             end if
06206         end do
06207     end do
06208
06209     !store the individual coordinate grids in the kernel
06210     do i = 1,q%Ndim
06211         allocate(q%grid(i)%x(dimcount(i)))
06212         q%grid(i)%x(:) = gridvalues(i,1:dimcount(i))
06213     end do
06214
06215     !deallocate unused memory again
06216     deallocate(dimcount, gridvalues, currvalues)
06217     return
```

#### 3.24.2.19 subroutine rkhs::save_kernel_to_file ( class(kernel) *Q,* character(len=*), intent(in) *datafile* ) [private]

Saves kernel in a binary file, so that it can be reused later.

**Author**

Oliver T. Unke, University of Basel

Definition at line 5819 of file RKHS.f90.

```
05819     implicit none
05820     character(len=*), intent(in) :: datafile
05821     class(kernel)                :: q
05822     integer                      :: i, ios
05823
05824     open(io_unit, file=datafile, form="unformatted", access="stream", &
05825                             action="write",     status="replace", iostat = ios)
05826     if(ios /= 0) then
05827         call throw_error("save_kernel_to_file",'Could not open "'//datafile//'".')
05828     end if
05829     write(io_unit) q%Ndim
05830     write(io_unit) q%Npoint
05831     if(allocated(q%k1d)) then
05832         write(io_unit) .true.
05833         do i = 1,size(q%k1d)
05834             write(io_unit) q%k1d(i)%kernel_type
05835             if(q%k1d(i)%Npar > 0) then
05836                 write(io_unit) q%k1d(i)%par
05837             end if
05838         end do
05839     else
05840         write(io_unit) .false.
05841     end if
05842
05843     if(allocated(q%grid)) then
05844         write(io_unit) .true.
05845         do i = 1,size(q%grid)
05846             if(allocated(q%grid(i)%x)) then
05847                 write(io_unit) .true.
05848                 write(io_unit) size(q%grid(i)%x)
05849                 write(io_unit) q%grid(i)%x
05850             else
05851                 write(io_unit) .false.
05852             end if
05853         end do
05854     else
05855         write(io_unit) .false.
05856     end if
05857
05858     if(allocated(q%values)) then
05859         write(io_unit) .true.
05860         write(io_unit) q%values
05861     else
05862         write(io_unit) .false.
05863     end if
05864
05865     if(allocated(q%valueIsPresent)) then
05866         write(io_unit) .true.
05867         write(io_unit) q%valueIsPresent
05868     else
05869         write(io_unit) .false.
05870     end if
05871
05872     if(allocated(q%alpha)) then
05873         write(io_unit) .true.
05874         write(io_unit) q%alpha
05875     else
05876         write(io_unit) .false.
05877     end if
05878
05879     if(allocated(q%invQ)) then
05880         write(io_unit) .true.
05881         write(io_unit) q%invQ
05882     else
05883         write(io_unit) .false.
05884     end if
05885
05886     if(allocated(q%sigma)) then
05887         write(io_unit) .true.
05888         write(io_unit) q%sigma
05889     else
05890         write(io_unit) .false.
05891     end if
05892
05893     close(io_unit)
05894     return
```

#### 3.24.2.20 subroutine rkhs::throw_error ( character(len=*), intent(in) *proc,* character(len=*), intent(in) *errormsg* ) [private]

Throws error messages if problems are encountered.

**Author**

Oliver T. Unke, University of Basel

Definition at line 5804 of file RKHS.f90.

```
05804     implicit none
05805     character(len=*), intent(in) :: proc, errormsg
05806
05807     write(*,*) "ERROR in module RKHS: "//proc//": "//errormsg
05808     stop
```

#### 3.24.2.21 subroutine rkhs::write_grid_data ( class(kernel) *Q,* character(len=*), intent(in) *datafile* ) [private]

Writes grid data as .csv file.

**Author**

Oliver T. Unke, University of Basel

Writes the complete information from which the kernel was constructed into a .csv file. This is useful to reconstruct the grid in case only the binary format of the kernel is available. The file has the format x1,x2,x3,...,xn,f where the xi are the individual coordinates and f the reference values. The file is sorted in this order: x1>x2>x3>...>xn

Definition at line 6243 of file RKHS.f90.

```
06243     implicit none
06244     character(len=*), intent(in) :: datafile
06245     class(kernel)                :: q
06246     integer, dimension(Q%Ndim)   :: dimindx !for looping over the dimensions
06247     integer                      :: ios, counter, d
06248
06249     !open grid datafile
06250     open(io_unit,file=datafile,status="replace",action="write",iostat = ios)
06251     if(ios /= 0) call throw_error("write_grid_data", 'File "'//datafile//'" could not be opened.')
06252     !write grid data to file
06253     dimindx = 1
06254     counter = 0
06255     do while(dimindx(1) <= size(q%grid(1)%x))
06256         counter = counter + 1
06257         do d = 1,q%Ndim
06258             write(io_unit,'(ES23.16,A1)',advance='no') q%grid(d)%x(dimindx(d)),','
06259         end do
06260         if(q%valueIsPresent(counter)) then
06261             write(io_unit,'(ES23.16)') q%values(counter)
06262         else
06263             write(io_unit,'(A23)') "NaN"
06264         end if
06265
06266         !increase dimindx
06267         dimindx(q%Ndim) = dimindx(q%Ndim) + 1
06268         do d = q%Ndim,2,-1
06269             if(dimindx(d) > size(q%grid(d)%x)) then
06270                 dimindx(d)   = 1
06271                 dimindx(d-1) = dimindx(d-1) + 1
06272             end if
06273         end do
06274     end do
06275     close(io_unit)
06276     return
```

# 4 Data Type Documentation

## 4.1 reproducing_kernels::f_ptr Type Reference

for wrapping function pointers, such that it is possible to have arrays of function pointers

**Public Attributes**

- procedure(func), pointer, nopass f

#### 4.1.1 Detailed Description

for wrapping function pointers, such that it is possible to have arrays of function pointers

Definition at line 4361 of file RKHS.f90.

#### 4.1.2 Member Data Documentation

##### 4.1.2.1 procedure(func), pointer, nopass reproducing_kernels::f_ptr::f

Definition at line 4362 of file RKHS.f90.

```
04362         procedure(func), pointer, nopass :: f
```

### 4.2 reproducing_kernels::func Interface Reference

interface for the f2k/f3k functions that are stored in function pointer arrays

#### Public Member Functions

- pure real(kind(0d0)) function func (x, par)
- pure real(kind(0d0)) function func (x, par)

#### 4.2.1 Detailed Description

interface for the f2k/f3k functions that are stored in function pointer arrays

Definition at line 4345 of file RKHS.f90.

#### 4.2.2 Constructor & Destructor Documentation

##### 4.2.2.1 pure real(kind(0d0)) function reproducing_kernels::func::func ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4347 of file RKHS.f90.

```
04347             real(kind(0d0)), intent(in)               :: x
04348             real(kind(0d0)), dimension(:), intent(in) :: par !optional parameters, e.g. beta for exp kernel
```

##### 4.2.2.2 pure real(kind(0d0)) function reproducing_kernels::func::func ( real(kind(0d0)), intent(in) *x,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3182 of file RKHS_preHessian_backup.f90.

```
03182             real(kind(0d0)), intent(in)               :: x
03183             real(kind(0d0)), dimension(:), intent(in) :: par !optional parameters, e.g. beta for exp kernel
```

### 4.3 rkhs::grid Type Reference

contained in kernel type, stores grid points for each dimension

#### Private Attributes

- real(kind(0d0)), dimension(:), allocatable x

#### 4.3.1 Detailed Description

contained in kernel type, stores grid points for each dimension

Definition at line 5745 of file RKHS.f90.

#### 4.3.2 Member Data Documentation

##### 4.3.2.1 real(kind(0d0)), dimension(:), allocatable rkhs::grid::x [private]

Definition at line 5746 of file RKHS.f90.

```
05746     real(kind(0d0)), dimension(:),   allocatable :: x
```

## 4.4 reproducing_kernels::kdirect Interface Reference

interface for naive implementation of kernels (instead of using the fast evaluation)

#### Public Member Functions

- pure real(kind(0d0)) function kdirect (x1, x2, par)
- pure real(kind(0d0)) function kdirect (x1, x2, par)

#### 4.4.1 Detailed Description

interface for naive implementation of kernels (instead of using the fast evaluation)

Definition at line 4353 of file RKHS.f90.

#### 4.4.2 Constructor & Destructor Documentation

##### 4.4.2.1 pure real(kind(0d0)) function reproducing_kernels::kdirect::kdirect ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 4355 of file RKHS.f90.

```
04355             real(kind(0d0)), intent(in)               :: x1,x2
04356             real(kind(0d0)), dimension(:), intent(in) :: par
```

##### 4.4.2.2 pure real(kind(0d0)) function reproducing_kernels::kdirect::kdirect ( real(kind(0d0)), intent(in) *x1,* real(kind(0d0)), intent(in) *x2,* real(kind(0d0)), dimension(:), intent(in) *par* )

Definition at line 3190 of file RKHS_preHessian_backup.f90.

```
03190             real(kind(0d0)), intent(in)               :: x1,x2
03191             real(kind(0d0)), dimension(:), intent(in) :: par
```

## 4.5 rkhs::kernel Type Reference

multi-dimensional kernel type (contains 1-d kernels and lookup tables, coefficients, etc.)

#### Private Member Functions

- procedure save_to_file => save_kernel_to_file
- procedure load_from_file => load_kernel_from_file
- procedure free => deallocate_kernel
- procedure read_grid => read_grid_data
- procedure write_grid => write_grid_data
- procedure evaluate_fast => evaluate_kernel_fast
- procedure calculate_error_bound => calculate_error_bound
- procedure evaluate_slow => evaluate_kernel_slow
- procedure calculate_coefficients_fast => calculate_coefficients_fast
- procedure calculate_coefficients_slow => calculate_coefficients_slow
- procedure calculate_sums => calculate_sums
- procedure get_alpha_idx => get_idx_from_indices
- procedure get_sig_idx => get_sig_idx_from_m_and_k
- procedure get_lookup_idx => get_sig_idx_from_indices
- procedure save_to_file => save_kernel_to_file
- procedure load_from_file => load_kernel_from_file
- procedure free => deallocate_kernel
- procedure read_grid => read_grid_data
- procedure write_grid => write_grid_data
- procedure evaluate_fast => evaluate_kernel_fast
- procedure calculate_error_bound => calculate_error_bound
- procedure evaluate_slow => evaluate_kernel_slow
- procedure calculate_coefficients_fast => calculate_coefficients_fast
- procedure calculate_coefficients_slow => calculate_coefficients_slow
- procedure calculate_sums => calculate_sums
- procedure get_alpha_idx => get_idx_from_indices
- procedure get_sig_idx => get_sig_idx_from_m_and_k
- procedure get_lookup_idx => get_sig_idx_from_indices

#### Private Attributes

- integer ndim = 0

  *number of dimensions/1-dimensional kernel functions*
- integer npoint = 0

  *total number of reference points*
- type(kernel_1d), dimension(:), allocatable k1d

  *array of 1-dimensional kernel functions*
- type(grid), dimension(:), allocatable grid

  *stores the values of the grid points of each 1-dimensional grid*
- real(kind(0d0)), dimension(:), allocatable values

  *stores the values of the Npoint reference points*
- real(kind(0d0)), dimension(:,:), allocatable invq

  *stores the inverse of the kernel matrix (needed for calculating error bound)*
- logical, dimension(:), allocatable valueispresent

  *stores which values of the reference points are not missing*
- real(kind(0d0)), dimension(:), allocatable alpha

  *stores the kernel coefficients (obtained by matrix inversion)*
- real(kind(0d0)), dimension(:,:), allocatable sigma

  *stores the lookup table of precomputed sums*
- real(kind(0d0)), dimension(:), allocatable gamma_old

  *stores intermediate results in fast evaluation*
- real(kind(0d0)), dimension(:), allocatable gamma_new

### 4.5.1 Detailed Description

multi-dimensional kernel type (contains 1-d kernels and lookup tables, coefficients, etc.)

Definition at line 5755 of file RKHS.f90.

### 4.5.2 Member Function/Subroutine Documentation

#### 4.5.2.1 procedure rkhs::kernel::calculate_coefficients_fast ( ) [private]

Definition at line 4343 of file RKHS_preHessian_backup.f90.

#### 4.5.2.2 procedure rkhs::kernel::calculate_coefficients_fast ( ) [private]

Definition at line 5787 of file RKHS.f90.

#### 4.5.2.3 procedure rkhs::kernel::calculate_coefficients_slow ( ) [private]

Definition at line 4344 of file RKHS_preHessian_backup.f90.

#### 4.5.2.4 procedure rkhs::kernel::calculate_coefficients_slow ( ) [private]

Definition at line 5788 of file RKHS.f90.

#### 4.5.2.5 procedure rkhs::kernel::calculate_error_bound ( ) [private]

Definition at line 4341 of file RKHS_preHessian_backup.f90.

#### 4.5.2.6 procedure rkhs::kernel::calculate_error_bound ( ) [private]

Definition at line 5785 of file RKHS.f90.

#### 4.5.2.7 procedure rkhs::kernel::calculate_sums ( ) [private]

Definition at line 4345 of file RKHS_preHessian_backup.f90.

#### 4.5.2.8 procedure rkhs::kernel::calculate_sums ( ) [private]

Definition at line 5789 of file RKHS.f90.

#### 4.5.2.9 procedure rkhs::kernel::evaluate_fast ( ) [private]

Definition at line 4340 of file RKHS_preHessian_backup.f90.

#### 4.5.2.10 procedure rkhs::kernel::evaluate_fast ( ) [private]

Definition at line 5784 of file RKHS.f90.

#### 4.5.2.11 procedure rkhs::kernel::evaluate_slow ( ) [private]

Definition at line 4342 of file RKHS_preHessian_backup.f90.

#### 4.5.2.12 procedure rkhs::kernel::evaluate_slow ( ) [private]

Definition at line 5786 of file RKHS.f90.

#### 4.5.2.13 procedure rkhs::kernel::free ( ) [private]

Definition at line 4337 of file RKHS_preHessian_backup.f90.

#### 4.5.2.14 procedure rkhs::kernel::free ( ) [private]

Definition at line 5781 of file RKHS.f90.

#### 4.5.2.15 procedure rkhs::kernel::get_alpha_idx ( ) [private]

Definition at line 4346 of file RKHS_preHessian_backup.f90.

#### 4.5.2.16 procedure rkhs::kernel::get_alpha_idx ( ) [private]

Definition at line 5790 of file RKHS.f90.

#### 4.5.2.17 procedure rkhs::kernel::get_lookup_idx ( ) [private]

Definition at line 4348 of file RKHS_preHessian_backup.f90.

#### 4.5.2.18 procedure rkhs::kernel::get_lookup_idx ( ) [private]

Definition at line 5792 of file RKHS.f90.

#### 4.5.2.19 procedure rkhs::kernel::get_sig_idx ( ) [private]

Definition at line 4347 of file RKHS_preHessian_backup.f90.

#### 4.5.2.20 procedure rkhs::kernel::get_sig_idx ( ) [private]

Definition at line 5791 of file RKHS.f90.

#### 4.5.2.21 procedure rkhs::kernel::load_from_file ( ) [private]

Definition at line 4336 of file RKHS_preHessian_backup.f90.

#### 4.5.2.22 procedure rkhs::kernel::load_from_file ( ) [private]

Definition at line 5780 of file RKHS.f90.

#### 4.5.2.23 procedure rkhs::kernel::read_grid ( ) [private]

Definition at line 4338 of file RKHS_preHessian_backup.f90.

#### 4.5.2.24 procedure rkhs::kernel::read_grid ( ) [private]

Definition at line 5782 of file RKHS.f90.

#### 4.5.2.25 procedure rkhs::kernel::save_to_file ( ) [private]

Definition at line 4335 of file RKHS_preHessian_backup.f90.

#### 4.5.2.26 procedure rkhs::kernel::save_to_file ( ) [private]

Definition at line 5779 of file RKHS.f90.

#### 4.5.2.27 procedure rkhs::kernel::write_grid ( ) [private]

Definition at line 4339 of file RKHS_preHessian_backup.f90.

#### 4.5.2.28 procedure rkhs::kernel::write_grid ( ) [private]

Definition at line 5783 of file RKHS.f90.

### 4.5.3 Member Data Documentation

#### 4.5.3.1 real(kind(0d0)), dimension(:), allocatable rkhs::kernel::alpha [private]

stores the kernel coefficients (obtained by matrix inversion)

Definition at line 5771 of file RKHS.f90.

```
05771     real(kind(0d0)), dimension(:),   allocatable :: alpha
```

#### 4.5.3.2 real(kind(0d0)), dimension(:), allocatable rkhs::kernel::gamma_new [private]

Definition at line 5776 of file RKHS.f90.

#### 4.5.3.3 real(kind(0d0)), dimension(:), allocatable rkhs::kernel::gamma_old [private]

stores intermediate results in fast evaluation

Definition at line 5776 of file RKHS.f90.

```
05776     real(kind(0d0)), dimension(:),   allocatable :: gamma_old, gamma_new
```

#### 4.5.3.4 type(grid), dimension(:), allocatable rkhs::kernel::grid [private]

stores the values of the grid points of each 1-dimensional grid

Definition at line 5763 of file RKHS.f90.

```
05763     type(grid),      dimension(:),   allocatable :: grid
```

#### 4.5.3.5 real(kind(0d0)), dimension(:,:), allocatable rkhs::kernel::invq [private]

stores the inverse of the kernel matrix (needed for calculating error bound)

Definition at line 5767 of file RKHS.f90.

```
05767     real(kind(0d0)), dimension(:,:), allocatable :: invq
```

#### 4.5.3.6 type(kernel_1d), dimension(:), allocatable rkhs::kernel::k1d [private]

array of 1-dimensional kernel functions

Definition at line 5761 of file RKHS.f90.

```
05761     type(kernel_1d), dimension(:),   allocatable :: k1d
```

#### 4.5.3.7 integer rkhs::kernel::ndim = 0 [private]

number of dimensions/1-dimensional kernel functions

Definition at line 5757 of file RKHS.f90.

```
05757     integer                                      :: ndim = 0
```

#### 4.5.3.8 integer rkhs::kernel::npoint = 0 [private]

total number of reference points

Definition at line 5759 of file RKHS.f90.

```
05759     integer                                      :: npoint = 0
```

#### 4.5.3.9 real(kind(0d0)), dimension(:,:), allocatable rkhs::kernel::sigma [private]

stores the lookup table of precomputed sums

Definition at line 5773 of file RKHS.f90.

```
05773     real(kind(0d0)), dimension(:,:), allocatable :: sigma
```

#### 4.5.3.10 logical, dimension(:), allocatable rkhs::kernel::valueispresent [private]

stores which values of the reference points are not missing

Definition at line 5769 of file RKHS.f90.

```
05769     logical,         dimension(:),   allocatable :: valueispresent
```

#### 4.5.3.11 real(kind(0d0)), dimension(:), allocatable rkhs::kernel::values [private]

stores the values of the Npoint reference points

Definition at line 5765 of file RKHS.f90.

```
05765     real(kind(0d0)), dimension(:),   allocatable :: values
```

## 4.6 reproducing_kernels::kernel_1d Type Reference

defines 1-dimensional kernels

### Public Member Functions

- procedure init => init_kernel
  *needs to be called to initialize the kernel and set it to a certain type*
- procedure init => init_kernel
  *needs to be called to initialize the kernel and set it to a certain type*

### Public Attributes

- integer kernel_type = -1
  *stores information about what type of kernel is used (-1 signals uninitialized kernels)*
- integer m2 = 0
  *how many f2k/f3k functions and p2k coefficients need to be used in the kernel decomposition*
- procedure(kdirect), pointer, nopass k
  *pointer to naive (slow) implementation of kernel function*
- procedure(kdirect), pointer, nopass dk
  *pointer to naive (slow) implementation of first derivative of kernel function*
- procedure(kdirect), pointer, nopass d2k
  *pointer to naive (slow) implementation of second derivative of kernel function*
- real(kind(0d0)), dimension(:), allocatable p2
  *array of the p2k coefficients*
- type(f_ptr), dimension(:), allocatable f2
  *array of the f2k function pointers*
- type(f_ptr), dimension(:), allocatable f3
  *array of the f3k function pointers*
- type(f_ptr), dimension(:), allocatable df2

*array of the first derivative of f2k function pointers*

- type(f_ptr), dimension(:), allocatable df3

  *array of the first derivative of f3k function pointers*
- type(f_ptr), dimension(:), allocatable d2f2

  *array of the first derivative of f2k function pointers*
- type(f_ptr), dimension(:), allocatable d2f3

  *array of the first derivative of f3k function pointers*
- integer npar = 0

  *needed for supporting kernels with parameters*
- real(kind(0d0)), dimension(:), allocatable par

  *array of parameters (only needed when kernel function has parameters)*

#### 4.6.1 Detailed Description

defines 1-dimensional kernels

Definition at line 4366 of file RKHS.f90.

#### 4.6.2 Member Function/Subroutine Documentation

##### 4.6.2.1 procedure reproducing_kernels::kernel_1d::init ( )

needs to be called to initialize the kernel and set it to a certain type

Definition at line 3226 of file RKHS_preHessian_backup.f90.

##### 4.6.2.2 procedure reproducing_kernels::kernel_1d::init ( )

needs to be called to initialize the kernel and set it to a certain type

Definition at line 4397 of file RKHS.f90.

#### 4.6.3 Member Data Documentation

##### 4.6.3.1 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::d2f2

array of the first derivative of f2k function pointers

Definition at line 4388 of file RKHS.f90.

```
04388         type(f_ptr),     dimension(:), allocatable :: d2f2
```

##### 4.6.3.2 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::d2f3

array of the first derivative of f3k function pointers

Definition at line 4390 of file RKHS.f90.

```
04390         type(f_ptr),     dimension(:), allocatable :: d2f3
```

##### 4.6.3.3 procedure(kdirect), pointer, nopass reproducing_kernels::kernel_1d::d2k

pointer to naive (slow) implementation of second derivative of kernel function

Definition at line 4376 of file RKHS.f90.

```
04376         procedure(kdirect), pointer, nopass :: d2k
```

#### 4.6.3.4 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::df2

array of the first derivative of f2k function pointers

array of the derivative of f2k function pointers

Definition at line 4384 of file RKHS.f90.

```
04384         type(f_ptr),     dimension(:), allocatable :: df2
```

#### 4.6.3.5 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::df3

array of the first derivative of f3k function pointers

array of the derivative of f3k function pointers

Definition at line 4386 of file RKHS.f90.

```
04386         type(f_ptr),     dimension(:), allocatable :: df3
```

#### 4.6.3.6 procedure(kdirect), pointer, nopass reproducing_kernels::kernel_1d::dk

pointer to naive (slow) implementation of first derivative of kernel function

pointer to naive (slow) implementation of derivative of kernel function

Definition at line 4374 of file RKHS.f90.

```
04374         procedure(kdirect), pointer, nopass :: dk
```

#### 4.6.3.7 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::f2

array of the f2k function pointers

Definition at line 4380 of file RKHS.f90.

```
04380         type(f_ptr),     dimension(:), allocatable :: f2
```

#### 4.6.3.8 type(f_ptr), dimension(:), allocatable reproducing_kernels::kernel_1d::f3

array of the f3k function pointers

Definition at line 4382 of file RKHS.f90.

```
04382         type(f_ptr),     dimension(:), allocatable :: f3
```

#### 4.6.3.9 procedure(kdirect), pointer, nopass reproducing_kernels::kernel_1d::k

pointer to naive (slow) implementation of kernel function

Definition at line 4372 of file RKHS.f90.

```
04372         procedure(kdirect), pointer, nopass :: k
```

#### 4.6.3.10 integer reproducing_kernels::kernel_1d::kernel_type = -1

stores information about what type of kernel is used (-1 signals uninitialized kernels)

Definition at line 4368 of file RKHS.f90.

```
04368         integer :: kernel_type = -1
```

#### 4.6.3.11 integer reproducing_kernels::kernel_1d::m2 = 0

how many f2k/f3k functions and p2k coefficients need to be used in the kernel decomposition

Definition at line 4370 of file RKHS.f90.

```
04370         integer :: m2 = 0
```

#### 4.6.3.12 integer reproducing_kernels::kernel_1d::npar = 0

needed for supporting kernels with parameters

Definition at line 4392 of file RKHS.f90.

```
04392         integer :: npar = 0
```

#### 4.6.3.13 real(kind(0d0)), dimension(:), allocatable reproducing_kernels::kernel_1d::p2

array of the p2k coefficients

Definition at line 4378 of file RKHS.f90.

```
04378         real(kind(0d0)), dimension(:), allocatable :: p2
```

#### 4.6.3.14 real(kind(0d0)), dimension(:), allocatable reproducing_kernels::kernel_1d::par

array of parameters (only needed when kernel function has parameters)

Definition at line 4394 of file RKHS.f90.

```
04394         real(kind(0d0)), dimension(:), allocatable :: par
```

## 4.7 rkhs::kernel_matrix Type Reference

wrapper for matrices, used to store a kernel matrix in tensor product form

### Private Attributes

- real(kind(0d0)), dimension(:,:), allocatable m

### 4.7.1 Detailed Description

wrapper for matrices, used to store a kernel matrix in tensor product form

Definition at line 5750 of file RKHS.f90.

### 4.7.2 Member Data Documentation

#### 4.7.2.1 real(kind(0d0)), dimension(:,:), allocatable rkhs::kernel_matrix::m [private]

Definition at line 5751 of file RKHS.f90.

```
05751     real(kind(0d0)), dimension(:,:), allocatable :: m
```