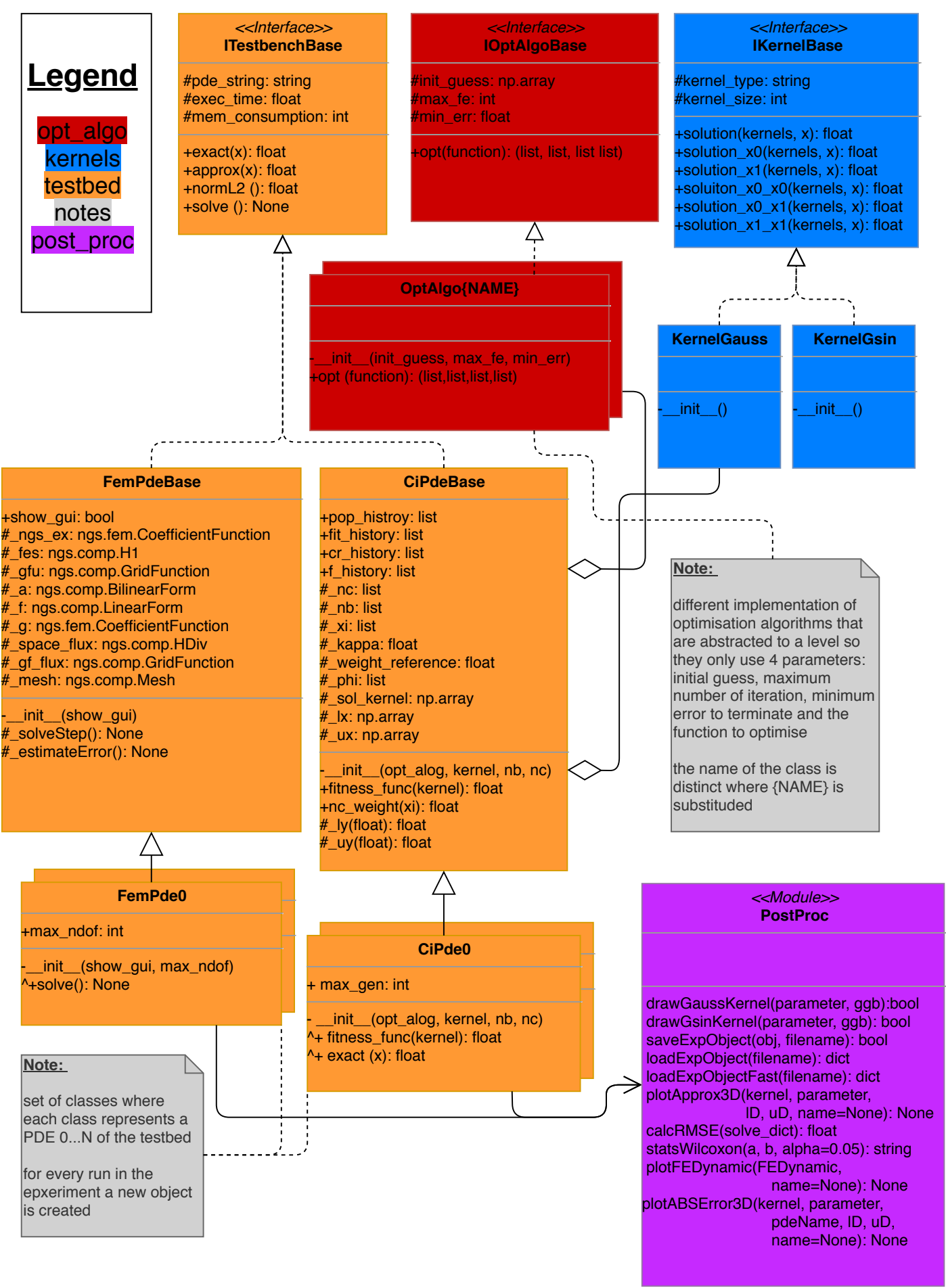
Legend
opt_algo
kernels
testbed
notes
post_proc
<<Interface>>
ITestbenchBase
#pde_string: string
#exec_time: float
#mem_consumption: int
+exact(x): float
+approx(x): float
+normL2 (): float
+solve (): None
<<Interface>>
IOptAlgoBase
#init_guess: np.array
#max_fe: int
#min_err: float
+opt(function): (list, list, list list)
<<Interface>>
IKernelBase
#kernel_type: string
#kernel_size: int
+solution(kernels, x): float
+solution_x0(kernels, x): float
+solution_x1(kernels, x): float
+soluiton_x0_x0(kernels, x): float
+solution_x0_x1(kernels, x): float
+solution_x1_x1(kernels, x): float
OptAlgo{NAME}
-__init__(init_guess, max_fe, min_err)
+opt (function): (list,list,list,list)
KernelGauss
-__init__()
KernelGsin
-__init__()
FemPdeBase
+show_gui: bool
#_ngs_ex: ngs.fem.CoefficientFunction
#_fes: ngs.comp.H1
#_gfu: ngs.comp.GridFunction
#_a: ngs.comp.BilinearForm
#_f: ngs.comp.LinearForm
#_g: ngs.fem.CoefficientFunction
#_space_flux: ngs.comp.HDiv
#_gf_flux: ngs.comp.GridFunction
#_mesh: ngs.comp.Mesh
-__init__(show_gui)
#_solveStep(): None
#_estimateError(): None
CiPdeBase
+pop_histroy: list
+fit_history: list
+cr_history: list
+f_history: list
#_nc: list
#_nb: list
#_xi: list
#_kappa: float
#_weight_reference: float
#_phi: list
#_sol_kernel: np.array
#_lx: np.array
#_ux: np.array
-__init__(opt_alog, kernel, nb, nc)
+fitness_func(kernel): float
+nc_weight(xi): float
#_ly(float): float
#_uy(float): float
Note:
different implementation of optimisation algorithms that are abstracted to a level so they only use 4 parameters: initial guess, maximum number of iteration, minimum error to terminate and the function to optimise
the name of the class is distinct where {NAME} is substituded
FemPde0
+max_ndof: int
-__init__(show_gui, max_ndof)
^+solve(): None
CiPde0
+ max_gen: int
- __init__(opt_alog, kernel, nb, nc)
^+ fitness_func(kernel): float
^+ exact (x): float
Note:
set of classes where each class represents a PDE 0...N of the testbed
for every run in the epxeriment a new object is created
<<Module>>
PostProc
drawGaussKernel(parameter, ggb):bool
drawGsinKernel(parameter, ggb): bool
saveExpObject(obj, filename): bool
loadExpObject(filename): dict
loadExpObjectFast(filename): dict
plotApprox3D(kernel, parameter, ID, uD, name=None): None
calcRMSE(solve_dict): float
statsWilcoxon(a, b, alpha=0.05): string
plotFEDynamic(FEDynamic, name=None): None
plotABSError3D(kernel, parameter, pdeName, ID, uD, name=None): None