A Milonga's input file for the 2D IAEA benchmark

```
#                       BENCHMARK PROBLEM
#
# Identification: 11-A2          Source Situation ID.11
# Date Submitted: June 1976      By: R. R. Lee (CE)
#                                    D. A. Menely (Ontario Hydro)
#                                    B. Micheelsen (Riso-Denmark)
#                                    D. R. Vondy (ORNL)
#                                    M. R. Wagner (KWU)
#                                    W. Werner (GRS-Munich)
#
# Date Accepted:  June 1977      By: H. L. Dodds, Jr. (U. of Tenn.)
#                                    M. V. Gregory (SRL)
#
# Descriptive Title: Two-dimensional LWR Problem,
#                    also 2D IAEA Benchmark Problem
#
# Reduction of Source Situation
#           1. Two-groupo diffusion theory
#           2. Two-dimensional (x,y)-geometry
#


# -----8<----- milonga's solution begins here -----8<-----
# the expected arguments are:
#   $1 = number of the case
#   $2 = type of geometry ( quarter / eigth )
#   $3 = meshing algorithm ( delaunay / delquad )
#   $4 = basic shape ( triangs / quads )
#   $5 = discretization scheme ( volumes / elements )
#   $6 = characteristic length of the element/cells

# we first create a subdirectory that will hold the
# files that correspond to the current case
SHELL "mkdir␣-p␣$1-$2-$3-$4-$5-$6"
# and greet the user through the standard output
PRINT TEXT "case␣$1-$2-$3-$4-$5-$6␣␣␣␣␣␣" NONEWLINE
# tell milonga we face a two-dimensional two-groups
# problem with an unstructured grid
PROBLEM DIMENSIONS 2 GROUPS 2 MESH $1-$2-$3-$4-$5-$6/benchmark.msh

SCHEME $5     # spatial discretization scheme
Bg2 = 0.8e-4  # axial geometric buckling in the z direction

# materials and cross sections according to the two-group constants
# each material corresponds to a physical entity in the geometry file
# XS can be given as algebraic expressions of x and y if needed
MATERIAL fuel1 {
 D_1 1.5   SigmaA_1 0.010+1.5*Bg2   SigmaS_1->2 0.02
 D_2 0.4   SigmaA_2 0.080+0.4*Bg2   nuSigmaF_2  0.135 eSigmaF_2 0.135 }
MATERIAL fuel2 {
 D_1 1.5   SigmaA_1 0.010+1.5*Bg2   SigmaS_1->2 0.02
 D_2 0.4   SigmaA_2 0.085+0.4*Bg2   nuSigmaF_2 0.135  eSigmaF_2 0.135 }
MATERIAL fuel2+rod {
 D_1 1.5   SigmaA_1 0.010+1.5*Bg2   SigmaS_1->2 0.02
 D_2 0.4   SigmaA_2 0.130+0.4*Bg2   nuSigmaF_2 0.135  eSigmaF_2 0.135 }
MATERIAL reflector {
 D_1 2.0   SigmaA_1 0.000+2.0*Bg2   SigmaS_1->2 0.04
 D_2 0.3   SigmaA_2 0.010+0.3*Bg2  }

# boundary conditions as requested by the problem, applied
# to appropriate physical entities defined in the geometry file
PHYSICAL_ENTITY external  BC robin  -0.4692
PHYSICAL_ENTITY mirror    BC mirror

# based on a geometry template file named $1.tpl for the
# type of symmetry used, generate a .geo file with the
# selected characteristic length, meshing algorithm and
# basic element/cell shape
# TODO: future versions ought to implement these kind of
# conditionals natively instead of relying on the shell
SHELL "echo␣\"lc=$6;\"␣>␣$1-$2-$3-$4-$5-$6/benchmark.geo"
SHELL "cat␣$2.tpl␣␣␣␣␣>>␣$1-$2-$3-$4-$5-$6/benchmark.geo"
SHELL "if␣[␣\"$3\"␣=␣\"delaunay\"␣];␣then␣echo␣\"Mesh.Algorithm=5;\"␣␣␣␣␣ ↩
       ␣␣␣␣␣␣␣␣␣␣>>␣$1-$2-$3-$4-$5-$6/benchmark.geo;␣fi"
SHELL "if␣[␣\"$3\"␣=␣\"delquad\"␣];␣␣then␣echo␣\"Mesh.Algorithm=8;\"␣␣␣␣␣ ↩
       ␣␣␣␣␣␣␣␣␣␣>>␣$1-$2-$3-$4-$5-$6/benchmark.geo;␣fi"
SHELL "if␣[␣\"$3\"␣=␣\"delquad\"␣];␣␣then␣echo␣\"Mesh. ↩
       RecombinationAlgorithm=8;\"␣>>␣$1-$2-$3-$4-$5-$6/benchmark.geo;␣fi ↩
       "
SHELL "if␣[␣\"$4\"␣=␣\"quads\"␣];␣␣␣␣then␣echo␣\"Mesh.RecombineAll=1;\"␣␣ ↩
       ␣␣␣␣␣␣␣␣␣␣>>␣$1-$2-$3-$4-$5-$6/benchmark.geo;␣fi"

# call gmsh whilst measuring how much wall time it takes
t1 = clock()
SHELL "gmsh␣-2␣$1-$2-$3-$4-$5-$6/benchmark.geo␣>␣/dev/null"
mesh_time = clock()-t1

# to force milonga to normalize the fluxes as requested, we
# can set a power setpoint equal to the core volume and fix
# eSigmaF = nuSigmaF (as we did in the MATERIAL section)
# in order to compute the volume of the core, we need to
# have the XS as functions of (x,y) available for integration
# this can only happen if we ask milonga to explicitly read
# the mesh before solving the eigenvalue problem with the
# READ_MESH keyword. Note that the mesh file for the problem
# was defined above with the PROBLEM keyword
READ_MESH

eps = 5e-3   # relative allowed integration error
key = 1      # GSL_INTEG_GAUSS15 (see GSL's documentation)
# we take the core as the geometric place of the
# points (x,y) that have a non-zero fission XS
incore(x,y) := greater(nuSigmaF_2(x,y),0)
power = integral(integral(incore(x,y), x, 0, 170, eps, key), y, 0, 170,  ↩
       eps, key)

# we finally ask milonga to solve the eigenvalue problem
SOLVE_PROBLEM

# and then we proceed to answer the requested items

#-----------------------------------------------------
# item 1
# we save the effective multiplication factor with
# six decimal digits in a text file
OUTPUT_FILE keff $1-$2-$3-$4-$5-$6/1-keff.txt
PRINT  FILE keff %.6f keff
# and the equivalent static reactivity in another one
rho = 1e5*(keff-1)/keff
OUTPUT_FILE rho  $1-$2-$3-$4-$5-$6/1-rho.txt
PRINT  FILE rho  %.2f rho

# by the way, inform the user we are almos done
wall = mesh_time+read_mesh_time+build_matrices_time+solve_time
PRINT SEPARATOR "␣" {
  TEXT "rho␣=" %.2f rho
  TEXT "pcm"
  TEXT "(" wall TEXT "segs␣)" }

#-----------------------------------------------------
# item 2 (not asked)
# write the two-dimensional flux distribution both in
# ASCII and in gmsh post-processing format
# the difussion coefficient is included to help decide
# if a point is in the core or in the reflector
OUTPUT_FILE flux-dist-ascii $1-$2-$3-$4-$5-$6/2-flux-dist.dat
OUTPUT_FILE flux-dist-post  $1-$2-$3-$4-$5-$6/2-flux-dist.pos

PRINT_FUNCTION FILE   flux-dist-ascii phi_1 phi_2 power_density incore
WRITE_OUTPUT_FOR_POST flux-dist-post  phi_1 phi_2 power_density

#-----------------------------------------------------
# item 2a
# the radial flux traverses phi(x,0) and phi(x,x)

OUTPUT_FILE flux-axis  $1-$2-$3-$4-$5-$6/2a-flux-axis.dat
OUTPUT_FILE flux-diag  $1-$2-$3-$4-$5-$6/2a-flux-diag.dat

phi1_axis(x) := phi_1(x,0)
phi2_axis(x) := phi_2(x,0)

phi1_diag(x) := phi_1(x,x)
phi2_diag(x) := phi_2(x,x)

PRINT_FUNCTION FILE flux-axis phi1_axis phi2_axis MIN 0 MAX 170 STEP 0.1
PRINT_FUNCTION FILE flux-diag phi1_diag phi2_diag MIN 0 MAX 170 STEP 0.1



#-----------------------------------------------------
# item 2b
# the maximum thermal flux has to be located in one of the
# solution points, and as computing the maximum of a 2-dimensional
# function is rather expensive, we use GNU sort over the
# ASCII file computed in item 2, filtering between the core
# and the reflector by the value of the fast diffusion coefficient
VAR x_max y_max phi1_max phi2_max

OUTPUT_FILE maximums $1-$2-$3-$4-$5-$6/2b-maximums.txt

SHELL "cat␣$1-$2-$3-$4-$5-$6/2-flux-dist.dat␣|␣grep␣1\.000000e+00$␣|␣sort ↩
       ␣-g␣-k3␣-r␣|␣head␣-n1␣>␣$1-$2-$3-$4-$5-$6/2b-maximum_core.dat"
IMPORT ASCII_FILE $1-$2-$3-$4-$5-$6/2b-maximum_core.dat x_max y_max  ↩
       phi1_max phi2_max
PRINT FILE maximums TEXT "maximum␣thermal␣flux␣in␣core␣␣␣␣␣␣is␣phi2␣=␣"  ↩
       phi2_max TEXT "at␣x=" x_max TEXT "y=" y_max

SHELL "cat␣$1-$2-$3-$4-$5-$6/2-flux-dist.dat␣|␣grep␣0\.000000e+00$␣|␣sort ↩
       ␣-g␣-k3␣-r␣|␣head␣-n1␣>␣$1-$2-$3-$4-$5-$6/2b-maximum_refl.dat"
IMPORT ASCII_FILE $1-$2-$3-$4-$5-$6/2b-maximum_refl.dat x_max y_max  ↩
       phi1_max phi2_max
PRINT FILE maximums TEXT "maximum␣thermal␣flux␣in␣reflector␣is␣phi2␣=␣"  ↩
       phi2_max TEXT "at␣x=" x_max TEXT "y=" y_max

#-----------------------------------------------------
# item 3 (and 7)
# we build a table with the average subassembly powers as item 7
# asks the average fluxes, we compute the two of them here

# the file assemblies.coords contains five columns with the
# coordinates that define each of the 38 subassemblies
FUNCTION xmin(k) FILE assemblies.coords COLUMNS 1 2
FUNCTION xmax(k) FILE assemblies.coords COLUMNS 1 3
FUNCTION ymin(k) FILE assemblies.coords COLUMNS 1 4
FUNCTION ymax(k) FILE assemblies.coords COLUMNS 1 5

# te allow the 8-symmetry geometry, we define "symmetric
# functions" that mirror the power and the fluxes around the
# line y = x, otherwise for those assemblies that contain a
# small area with y > x, their averages would be underestimated
simpow(x,y)  := if(greater(x,y), power_density(x,y), power_density(y,x))
simphi1(x,y) := if(greater(x,y), phi_1(x,y), phi_1(y,x))
simphi2(x,y) := if(greater(x,y), phi_2(x,y), phi_2(y,x))


# volume of the k-th subassembly
vol(k) := (xmax(k)-xmin(k))*(ymax(k)-ymin(k))

# number of points used in the gauss-legendre quadrature
points = 8
```

```
mean_power(k)  := 1/vol(k) * gauss_legendre(gauss_legendre( simpow(x,y),  ↩
       x, xmin(k), xmax(k), points), y, ymin(k), ymax(k), points)
mean_flux_1(k) := 1/vol(k) * gauss_legendre(gauss_legendre(simphi1(x,y),  ↩
       x, xmin(k), xmax(k), points), y, ymin(k), ymax(k), points)
mean_flux_2(k) := 1/vol(k) * gauss_legendre(gauss_legendre(simphi2(x,y),  ↩
       x, xmin(k), xmax(k), points), y, ymin(k), ymax(k), points)

OUTPUT_FILE table $1-$2-$3-$4-$5-$6/3-table.dat
PRINT TEXT "\#␣␣k␣␣␣␣P_k␣␣␣fast_k␣␣␣thermal_k" FILE table
PRINT_FUNCTION  mean_power mean_flux_1 mean_flux_2 MIN xmin_a MAX xmax_b  ↩
       STEP 1 FILE table FORMAT %.3f


#-----------------------------------------------------
# items 4, 5 & 6
# the requested information is written by milonga if
# the keyword DEBUG is used
DEBUG FILE_PATH $1-$2-$3-$4-$5-$6/4-info.txt

#-----------------------------------------------------
# item 7
# included in point 3

#-----------------------------------------------------
# item 8
# it is not clear what does "results" mean, but we
# write a text file with some information we can later
# use to compare all the cases as a function of the
# size of the problem matrices (number of unknowns)
OUTPUT_FILE times $1-$2-$3-$4-$5-$6/8-results.dat
PRINT FILE times  {
  %g unknowns nodes cells elements
  %e keff rho
     mesh_time read_mesh_time build_matrices_time solve_time
  %.2f wall }

# -----8<----- milonga's solution ends here -----8<-----
```