Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXIII, págs. 3017-3036 (artículo completo)
Graciela Bertolino, Mariano Cantero, Mario Storti y Federico Teruel (Eds.)
San Carlos de Bariloche, 23-26 Setiembre 2014

PUMA VERSIÓN 6, MULTIPLATAFORMA CON FACILIDADES PARA SU ACOPLAMIENTO CON OTROS MODELOS DE SIMULACIÓN

Carlos R. Grant

Comisión Nacional de Energía Atómica, Centro Atómico Constituyentes, Avenida General Paz 1499, 1650 San Martín, Prov. de Buenos Aires, Argentina, grant@cnea.gov.ar,

http://www.cnea.gov.ar

Palabras Clave: Teoría de difusión, Gestión de combustible, Cinética espacial, PYTHON, ADA, LINUX, Acoplamiento, Verificación Experimental, Comparación con Otros Sistemas.

Resumen. PUMA es un sistema para la simulación del funcionamiento de reactores que es utilizado en todas las instalaciones nucleares de nuestro país para evaluar comportamientos en la gestión de combustible, ciclos de potencia y transitorios mediante la cinética espacio temporal por medio de la resolución de la ecuación de difusión tanto en régimen estacionario como en no estacionario.

Se ha utilizado para la gestión de combustible de los reactores de las centrales nucleares de ATUCHA I, ATUCHA II, EMBALSE, el reactor RA3 y para el cálculo alternativo del reactor CAREM. Ha sido verificado con datos experimentales de todos los reactores en funcionamiento y en instalaciones experimentales como el reactor experimental brasileño IPEN/MB-01.

En las versiones programadas hasta el momento se utilizó principalmente en la plataforma WINDOWS con muy buenos resultados. Atendiendo a las nuevas tendencias en el desarrollo de software y las posibilidades que nos dan las computadoras actuales, se hace necesario que PUMA funcione en diferentes sistemas operativos y además que tenga facilidades para su acoplamiento con otros modelos.

Para dar la posibilidad de que PUMA funcione en cualquier plataforma, LINUX entre otras, en esta versión 6 fue reprogramado totalmente en el lenguaje ADA, el cual está orientado hacia una programación segura y se puede aplicar en todos los sistemas operativos.

Las versiones anteriores de PUMA se ejecutaban a través de macro instrucciones basadas en el lenguaje LOGO. En la versión actual es posible usar también instrucciones desde las versiones del lenguaje PYTHON a través del puerto de comunicaciones local.

Se muestran ejemplos de manejo de entrada de datos y control del proceso en PUMA a través de PYTHON y se discute la implementación de esta metodología en otros modelos de simulación para poder acoplarlos con PUMA, tanto en su versión WINDOWS como LINUX.

1 INTRODUCCIÓN

PUMA es un sistema para la simulación del funcionamiento de reactores utilizado en todas las instalaciones nucleares de nuestro país.

En las versiones programadas hasta el momento se utilizó principalmente en la plataforma WINDOWS con muy buenos resultados.

Atendiendo a las nuevas tendencias en el desarrollo de software y las posibilidades que nos dan las computadoras actuales, se hace necesario que PUMA funcione en diferentes sistemas operativos y además que tenga facilidades para su acoplamiento con otros modelos que simulen, por ejemplo, la termohidráulica, el comportamiento de combustible o el cálculo de celda en cada lugar del reactor a fin de evaluar el quemado y las secciones eficaces, entre otras alternativas.

Para dar la posibilidad de que PUMA funcione en cualquier plataforma, LINUX entre otras, esta versión 6 fue reprogramada totalmente en el lenguaje ADA, el cual está orientado hacia una programación segura y se puede procesar en todos los sistemas operativos.

Las versiones anteriores de PUMA se ejecutaban a través de macro instrucciones basadas en el lenguaje LOGO. En la versión actual es posible usar también instrucciones desde las versiones del lenguaje PYTHON instalada en todos los sistemas operativos.

A través este recurso se puede acceder en tiempo de ejecución desde PYTHON a datos que se manejan dentro de PUMA, es decir a cualquiera de sus variables importantes tales como la reactividad, distribuciones de flujo, potencia, posiciones de barras de control y proveer los valores de otras variables tomadas de otros modelos externos, tales como distribuciones de temperatura, distribución de concentraciones isotópicas, etc.

Este tipo de acople se hace a través del puerto de comunicaciones local y si se cuenta con otros modelos que usan la misma metodología, el acoplamiento de dos o más programas en tiempo de ejecución se puede realizar directamente sin interrumpir la ejecución de cada uno de ellos. De este modo no es necesario realizar la comunicación a través de archivos externos, arrancando e interrumpiendo alternativamente la ejecución de los mismos.

Se muestran ejemplos de manejo de entrada de datos y control del proceso en PUMA a través de PYTHON y se discute la implementación de esta metodología en otros modelos de simulación para poder acoplarlos con PUMA, tanto en su versión WINDOWS como LINUX.

2 CAMBIO DEL LENGUAJE DE PROGRAMACIÓN

El progreso que ha ocurrido en los últimos años en el desempeño de las computadoras, su velocidad y su disponibilidad de memoria tanto periférica como interna permiten programar con mayor comodidad y darle mejores recursos al usuario de grandes sistemas de computación.

Por otra parte, también ha habido una evolución muy notable en el desarrollo de lenguajes de programación. Para el caso de PUMA el mejor lenguaje disponible durante el desarrollo de las versiones anteriores para la programación científica era PL/I y por este motivo fue el lenguaje elegido.

En la actualidad PL/I está casi en desuso por diversas razones. Prácticamente se lo utiliza sólo dentro del ámbito de la empresa que lo creó y promovió y no ha tenido ninguna evolución importante. En INTERNET hay pocos lugares donde se encuentra información útil sobre el mismo y además los compiladores no son demasiado accesibles a los usuarios y tienen un costo asociado.

Se ha llegado a la conclusión de que para que en el futuro PUMA pueda ser correctamente

mantenido por otros programadores además del autor, resultaba conveniente traducirlo a otro lenguaje, siendo escogido, luego de un cuidadoso análisis, el lenguaje de programación ADA a estos efectos.

El lenguaje de programación ADA tiene en la comunidad internacional una enorme cantidad de usuarios agrupados en varios países en organizaciones tales como ADA ESPAÑA, ADA EUROPE (con sus filiales en Francia, Dinamarca, Alemania, Bélgica, Suiza, Suecia) y ADACORE, que son organizaciones que lo promueven y ponen a disposición de los programadores un excelente compilador sin cargo. El nombre de este lenguaje, fue puesto en honor a Ada Byron, condesa de Lovelace, pionera de la Programación y de la Computación. Realiza el primer programa, describe la entonces llamada “Máquina Analítica" de Charles Babbage, e intuye que los desarrollos y operaciones de la Matemática son susceptibles de ser ejecutados por máquinas.

ADA fue creado alrededor de 1980 con el énfasis puesto en la confiabilidad, utilizado para la programación de sistemas donde este aspecto es esencial, tal como, por ejemplo, una terminal aeronáutica. Fue pensado, entre otras aplicaciones, para los grandes sistemas de programación científica y periódicamente se realizan congresos sobre programación en este lenguaje especialmente en lo que se refiere a "programación segura".

Por todo lo expuesto es que se tradujo la totalidad de la codificación de PUMA en PL/I al lenguaje ADA y además, se reformaron y reformularon muchos aspectos de la misma con el propósito de tener un programa más legible y más fácil de mantener.

Existen en INTERNET numerosos tutoriales para aprender el lenguaje ADA, entre otros:

http://www.infres.enst.fr/~pautet/Ada95/a95list.htm

El aspecto más importante que interesa para el desarrollo de sistemas de computación para problemas científicos en este leguaje es la organización del programa en "paquetes" o "módulos" en los que cada uno de ellos consiste en dos archivos, uno que contiene las especificaciones del mismo y muestra todo lo que puede ser accedido desde otros módulos y otro que contiene el "cuerpo" o la implementación de los procedimientos y funciones declarado en las especificaciones y allí están las instrucciones para la ejecución de cada uno de ellos.

3 ECUACIÓN DE DIFUSIÓN

El sistema PUMA resuelve la ecuación de difusión a G grupos de energía y M precursores de neutrones retardados de la forma:

$$\nabla\cdot(D_g\nabla\phi_g)-\Sigma_{Tg}\phi_g+\sum_{g'\neq g}\Sigma_{g'g}\phi_g+(1-\beta)\chi_{pg}\sum_{g'}\nu\Sigma_{fg'}\phi_{g'}+$$

$$+\sum_m\lambda_m S_{mg}C_m+Q_g=\frac{1}{V_g}\frac{\partial\phi_g}{\partial t}\qquad(g=1..G)\qquad(1)$$

$$\beta_m\sum_{g'}\nu\Sigma_{fg'}\phi_{g'}-\lambda_m C_m=\frac{\partial C_m}{\partial t}\qquad(m=1..M)$$

Las Σ significan secciones eficaces macroscópicas. La notación de las diferentes magnitudes es la usual en toda la literatura escrita sobre el tema. G es el número de grupos de energía y M el número de precursores de neutrones retardados.

Las secciones eficaces, flujos y concentraciones dependen en general del espacio y del tiempo.

Las ecuaciones (1) corresponden al estado no estacionario. En estado estacionario se transforma a las (1) en el problema:

$$\nabla \cdot (D_g \nabla \phi_g) - \Sigma_{Tg}\phi_g + \sum_{g' \neq g} \Sigma_{g' g}\phi_g + \chi_{Eg} \sum_{g'} \nu\Sigma_{fg'}\phi_{g'} = Q_g \tag{2}$$

donde χ_{Eg} es el espectro estático dado por:

$$\chi_{Eg} = (1-\beta)\chi_{Pg} + \sum_m \beta_m S_{mg} \tag{3}$$

para todos los valores de g .

En el caso de que las fuentes fijas Q_g se anulen se introduce en la (2) un autovalor, lo que resulta en:

$$\nabla \cdot (D_g \nabla \phi_g) - \Sigma_{Tg}\phi_g + \sum_{g' \neq g} \Sigma_{gg'}\phi_g + \frac{\chi_{Eg}}{k_e} \sum_{g'} \nu\Sigma_{fg'}\phi_{g'} = 0 \tag{4}$$

donde k_e es el factor de multiplicación.

También es de especial interés la ecuación adjunta de la (4) que nos propone la función importancia ϕ_g^*:

$$\nabla \cdot (D_g \nabla \phi_g^*) - \Sigma_{Tg}\phi_g^* + \sum_{g' \neq g} \Sigma_{gg'}\phi_{g'}^* + \frac{\nu\Sigma_{fg}}{k_e} \sum_{g'} \chi_{Eg'}\phi_{g'}^* = 0 \tag{5}$$

Las ecuaciones (2), (4) y (5) se resuelven por los métodos iterativos diseñados para este tipo de problemas una vez que se ha discretizado el modelo.

Para la resolución de la ecuaciones (1) existen varios métodos que están descriptos con más detalle en el informe correspondiente (Grant, 2013b).

PUMA resuelve tres tipos de problemas de simulación del funcionamiento de un reactor nuclear:

3.1 Evolución del reactor a largo plazo. Gestión de combustible

Para este tipo de simulaciones los tiempos se miden generalmente en días y se calculan para cada instante las nuevas distribuciones de quemado. Los estados del reactor son todos estacionarios y se resuelve la ecuación (4). Si se incluyen operaciones de recambio de combustible se tratan entonces, de cálculos de gestión de combustible.

3.2 Ciclos de Potencia

Los tiempos se miden normalmente en horas y en general no se tiene en cuenta la variación del quemado. Se calculan sucesivos estados estacionarios resolviendo la ecuación (4) tomando sólo en cuenta el cambio no estacionario de las concentraciones de algunos isótopos tales como el Xenón y el Iodo. Se parte de un estado inicial en el que se pone todo el sistema en equilibrio, incluido el Xenón.

3.3 Cinética espacial

Se resuelven en cada instante las ecuaciones cinéticas espaciales acopladas con los parámetros adicionales por medio de los modelos físico - matemáticos acoplados. Este es el caso del análisis de accidentes o del control del reactor. En este caso se resuelve directamente la ecuación (1) en estado no estacionario (C. Grant, 2013b).

Se parte siempre de un estado estacionario inicial en el que se pone estacionario todo el sistema. Resulta de este cálculo un valor de k efectivo que dividirá siempre a las fuentes en la evolución no estacionaria. Luego, durante esta evolución, se varían los parámetros de entrada del reactor y las inserciones de las barras en función del tiempo mediante las perturbaciones. Los tiempos se suelen medir en segundos.

4 CONTROL DEL PROCESO

4.1 Uso de LOGO. Ejemplo

El sistema PUMA, en las versiones anteriores, se procesaba activando los diferentes módulos desde un control de proceso escrito en lenguaje LOGO. En la versión 6 se agregó la posibilidad de utilizar también el lenguaje PYTHON.

Un ejemplo simple de procesamiento de PUMA sería:

```
* ESTADO DEL REACTOR
 … (definición de los materiales)
* SECCIONES EFICACES
* SECCIONES EFICACES PARA LA RED
* CONSTANTES DE MALLA
* FLUJO ESTACIONARIO
  500 iteraciones, beta 1.80,
  precisión Keff 1.0E-6,
  precisión flujo 1.0e-5;

* POTENCIA
  potencia total 0.04 MW ;

* IMPRIMIR
  parámetros globales , distribución de potencia;
```

Esto mismo se puede generalizar mediante procedimientos. Por ejemplo, si definimos:

```
+ PROC Flujos :IT :be :fp :P;
*   SECCIONES EFICACES
*   SECCIONES EFICACES PARA LA RED
*   CONSTANTES DE MALLA
*   FLUJO ESTACIONARIO
     :IT iteraciones, beta :be,
      precisión Keff 1.0E-6,
      precisión flujo :fp ;
*   POTENCIA
      potencia total :P MW ;
+ FIN PROC ;
```

Y ahora un ciclo de potencia se puede generar por medio de un procedimiento:

```
+ PROC ciclo :dt :pot ;
* XENON
   paso  :dt horas  ;

+  niter := 1  maxiter := 4  convergido := "FALSO" ;

+ MIENTRAS (NO :convergencia) & (niter <= maxiter) :
*   SECCIONES EFICACES
      1 parámetro acoplado  'XENON'  ;
+   Flujos 300 1.6 1.0E-6  :pot ;
*   XENON
      Cálculo transitorio , paso  :dt horas  ;
+     convergido := convergencia ;
+     niter := niter + 1 ;
+ FINMIENTRAS ;
+ FIN PROC ;
```

El signo "+" en la primera columna indica que esa línea contiene una instrucción en LOGO.

4.2 Ejemplo en PYTHON

El sistema PUMA permite en la versión VI utilizar PYTHON, un lenguaje del tipo script que está implementado en todos los sistemas operativos y provee una cantidad enorme de posibilidades gráficas y de intercambio de información con todo tipo de programas.

En el caso anterior el procedimiento PYTHON para calcular la distribución de flujo es:

```
def Fluxes (IT,be,fp,P):
  """
* SECCIONES EFICACES
* CONSTANTES DE MALLA
* FLUJO ESTACIONARIO
  :IT iteraciones , beta :be,
  precisión del flujo fp ;
* POTENCIA
  Potencia total :P ,
  """
```

Lo que está entre dos triples comillas para PYTHON es un comentario. Como PYTHON define los bloques lógicos con la indentación, las comillas se deben poner en el lugar justo donde va la indentación del bloque siguiente.

Ahora se utiliza un preprocesador de PYTHON que traduce lo escrito arriba a:

```
def Fluxes (IT,be,fp,P):
  transfer ("* SECCIONES EFICACES")
  transfer ("* CONSTANTES DE MALLA")
```

```
transfer ("* FLUJO ESTACIONARIO")
transfer ("  :IT iteraciones , beta :be,")
transfer ("  precisión del flujo :fp ;")
transfer ("* POTENCIA")
transfer ("  Potencia total :P , ")
```

Se observa que la indentación del procedimiento es colocada donde originalmente se pusieron las triples comillas, en este caso dos espacios hacia adentro. La instrucción "transfer" transfiere a PUMA la instrucción dada como argumento.

La conexión entre PUMA y PYTHON se realiza mediante el puerto de comunicación local (0,0,0,127) en forma muy sencilla.

Para el ciclo de potencia escrito antes en LOGO tenemos:

```
def cycle (dt,pot) :
  """
  transfer ("* XENON")
  transfer ("step :dt hours ")
  """
  niter = 1 ;  maxiter = 4  ;      converged = False
  while (not converged)            and (niter <= maxiter):
    """
   * CROSS SECTIONS
     1 coupled parameter  'XENON',
    """
    Fluxes 300 1.6 1.0E-6 pot ;
    """
   * XENON
      Transient calculation , step :dt hours;
    """
   converged = convergence ;
    niter = niter + 1
```

Este procedimiento se muestra ahora traducido directamente por el preprocesador:

```
def cycle (dt,pot) :
  transfer ("* XENON")
  transfer ("step :dt hours ")
  niter = 1 ;  maxiter = 4  ;
  converged = False

  while (not converged) and (niter <= maxiter):
    transfer ("* CROSS SECTIONS ")
    transfer ("1 coupled parameter  'XENON',")
    Fluxes 300 1.6 1.0E-6 pot ;
    transfer ("* XENON ")
    transfer ("Transient calculation , step :dt hours;")
    converged = convergence ;
    niter = niter + 1
```

Se debe observar , como en los otros casos, que la terminación del procedimiento y del ciclo "while" no están explícitamente definidos porque los bloques abarcados por los mismos están determinados por la indentación. Ésta es la que define el flujo lógico del programa.

4.3 Ejemplo en PYTHON para la gestión de combustible de EMBALSE

Aquí damos un ejemplo para la gestión de combustible en embalse:

```
def principal():
  hallar_flujos ([46, 69, 56, 56, 48, 57, 52,
                  39, 45, 18, 76, 35, 66, 54])
  """
  evolucionar hasta 3386.432  dias ,
  """
  mover ("L11","","H001 H002 H003 H004 H005 H006 H007 H008",
                  "16/02/95-17.18","tabla 2")
  """
  evolucionar hasta 3387.441  dias ,
  """
  mover ("L12","","H009 H010 H011 H012 H0013 H0014 H015 H0016",
                  "17/02/95-02.08","tabla 2")
  """
  evolucionar hasta 3391.500 dias ;
  """
  hallar_flujos ([44,48,62,55,59,50,62,
                  28,20,55,51,57,55,47])

  """
* IMPRIMIR
  parametros globales ,
  potencias especificas volumetricas ,
  potencias totales por canal ;

* FIN DEL PROCESO
  """
```

Los procedimientos marcados en color azul están definidos en otro archivo que se carga con el programa. "hallar_flujos" pide calcular la distribución de flujo y potencia dando el nivel de las zonas líquidas. "mover" realiza el movimiento de combustible. Si suponemos que estos procedimientos están en un archivo de nombre **"aux.py"** la ejecución se realiza mediante:

```
PYTHON pumaw.py input.py aux.py output.txt
```

donde **“pumaw.py”** es un programa PYTHON auxiliar para preparar **“input.py”** junto con **“aux.py”** para su ejecución.

5 ACOPLAMIENTO CON OTROS SISTEMAS

En la actualidad es frecuente acoplar sistemas de cálculo de distribución de flujo y potencia en un reactor nuclear con modelos acoplados que pueden ser termohidráulicos, de comportamiento y desgaste de los combustibles y pastillas, representar parte del primario de una central nuclear, etc.

Para este tipo de acoplamiento existen varios métodos que describiremos a continuación:

5.1 Acoplamiento externo

Se hace mediante el mismo sistema operativo, por ejemplo con los archivos de control de proceso del DOS en WINDOWS con la extensión "*.BAT". Estos archivos llaman alternativamente a PUMA y al programa acoplado y cada paso consiste en:

1) Leer la entrada de PUMA producida por el utilitario que adapta la salida del modelo acoplado y ejecutar PUMA. PUMA evolucionará un paso de tiempo en la simulación.

2) Procesar la salida de PUMA del paso anterior y con un utilitario producir la entrada para el modelo acoplado.

3) Procesar el modelo acoplado con la entrada producida en 2) y evolucionar un paso de tiempo.

4) Procesar la salida del modelo acoplado con otro utilitario y con otro utilitario producir la entrada para PUMA para un nuevo paso.

Estos cuatro pasos se ejecutan cada paso de tiempo y requiere la programación de dos utilitarios que armen las entradas de cada uno de los modelos que contendrán toda la información necesaria para continuar el cálculo paso por paso. Pero en cada uno de estos pasos, tanto la ejecución de PUMA como la del modelo acoplado se interrumpen y se reinicia con el estado anterior.

Obviamente, este sistema es muy engorroso y requiere mucho trabajo. PUMA ha sido acoplado con CATHENA y FIREBIRD mediante esta metodología con buen resultado.

5.2 Uso de un solo utilitario

En este caso se programa un solo utilitario y es éste que hace la tarea de adaptación mutua entre PUMA y el sistema acoplado.

Figura 1: Ventana de diálogo para el acoplamiento de PUMA con THERMIT utilizada en la simulación del ciclo de combustible del CAREM

Por ejemplo, PUMA y el sistema de cálculo termohidráulico THERMIT fueron acoplados mediante un utilitario programado en DELPHI (Object PASCAL) para el cálculo de la gestión de combustible del CAREM con buen resultado.

En la figura 1 se puede ver cómo es la interfaz del utilitario donde se puede elegir varias opciones, fijar valores de parámetros o mover barras. Este utilitario llama alternativamente a PUMA y THERMIT preparando la entrada de datos de cada uno de ellos a partir de la salida del otro. Cada vez que estos modelos se llaman, se ejecutan un paso y se detiene la ejecución de los mismos.

5.3 Acoplamiento interno

Consiste en no interrumpir en ningún momento la ejecución de PUMA o el modelo acoplado, es decir, acceder directamente a los valores de las variables dentro de la memoria de los mismos y pasarles en cada caso las instrucciones para la ejecución de éstos.

Eso se hace usando directamente alguna herramienta de programación que permita mantener en ejecución los modelos y en estado de accesibilidad permanente de los valores de sus variables, cosa que en PUMA se realiza usando PYTHON.

En PUMA hay instrucciones en las cuales se puede cargar a una variable PYTHON, por ejemplo, la distribución espacial de flujo y potencia, la potencia total y muchas otras distribuciones de parámetros puntuales, bidimensionales o tridimensionales. Por ejemplo:

```
def cargar
  """
* ESTADO DEL REACTOR
   leer distribución de potencia
           en la variable PYTHON 'PWR_DISTR' ,
   leer distribución flujo térmico
           en la variable PYTHON 'FL_DISTR' ;
  """
```

en las variables PYTHON **'PWR_DISTR'** y **'FL_DISTR'** se cargan las distribuciones de flujo y potencia. Si el modelo acoplado permite el mismo tipo de acceso a sus variables que PUMA se transfieren directamente estos valores y si no está presente esta opción se prepara la entrada y se ejecuta dicho modelo como en los casos anteriores.

Todas las posibilidades de acople entre programas descriptas hasta aquí son importantes hoy en día porque hay una tendencia a la "física múltiple", es decir, acoplar programas para evaluar simultáneamente la evolución de muchos parámetros que no dependen de un solo modelo sino de varios que además varían según el tipo de reactor a considerar.

6 COMPARACIÓN CON RESULTADOS EXPERIMENTALES Y OTROS SISTEMAS

Se han hecho numerosas comparaciones de PUMA con resultados experimentales y otros sistemas de cálculo. Aquí mencionaremos algunas.

6.1 Experimentos en el reactor brasileño IPEN/MB-01

Se trata de experiencias realizadas en el reactor brasileño de investigación IPEN/MB-01 arriba mencionado en el marco de un convenio de cooperación argentino brasileño COBEN (Dos Santos, Grant, Siqueira, Tarazaga, Andrade e Silva, Barberis, 2011b).

En esas experiencias se compararon 56 configuraciones críticas y para un núcleo para el que se midieron distribuciones de potencia y parámetros cinéticos se compararon los resultados de estas mediciones con lo calculado en PUMA.

El núcleo consiste en un arreglo rectangular de 28 por 26 barras de combustible de UO2 enriquecido en 4.3486% (en peso),y una vaina de acero inoxidable (SS-304), inmerso en un tanque de agua liviana desmineralizada.

El control del reactor es realizado por medio de dos bancos de barras opuestos entre sí diagonalmente; cada banco de control es compuesto por 12 barras absorbentes de Ag-In-Cd, y cada banco de seguridad consiste en 12 barras absorbentes de B4C. La ubicación de de los bancos en el plano X-Y puede verse en la figura 2. El reticulado del reactor es rectangular con un paso de 1.5 cm. La altura del núcleo es de 54.84 cm.

La figura 2 muestra algunos detalles en el plano XY.

Todos los datos geométricos provistos por las IPEN, junto con sus composiciones de materiales han sido usados como datos de entrada para HUEMUL, PUMA y MCNP5.

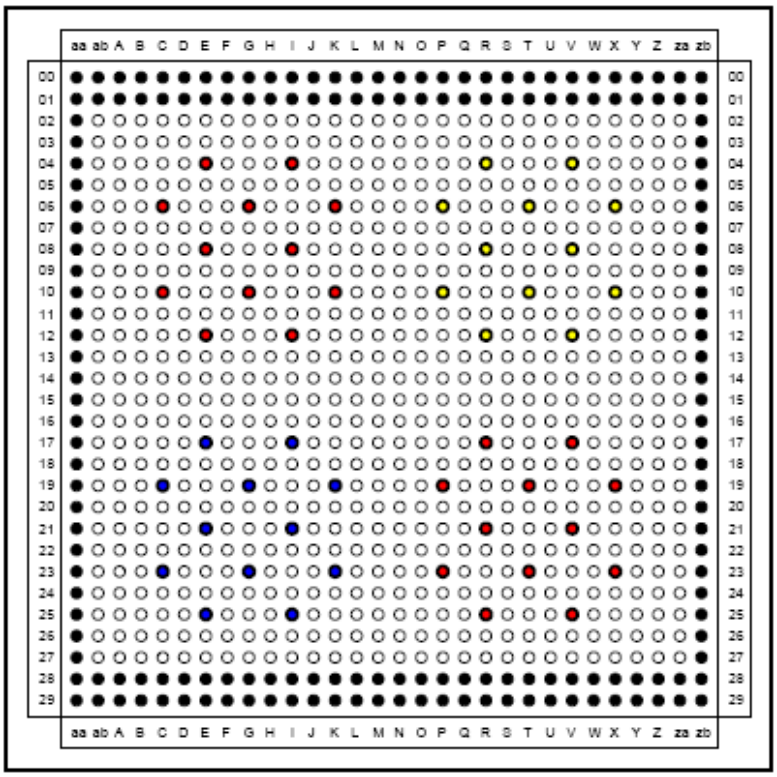

Figura 2. Esquema detallado del núcleo; plano transversal. Las posiciones en blanco corresponden a lugares para elementos combustibles que según sea el caso, pueden estar ocupados por elementos combustibles, por barras con venenos quemables, barras de cobre o acero o llenas de agua. Las posiciones en rojo corresponden a los elementos absorbentes del sistema de seguridad o a agua en su ausencia. Las posiciones en amarillo y azul corresponden a los elementos absorbentes de los bancos de control 1 y 2, BC1 y BC2 respectivamente o a agua en su ausencia. Las posiciones en negro corresponden a agua que funciona aquí como reflector.

Para más detalles del reactor y de las experiencias ver el informe de COBEN (Dos Santos, Grant, Siqueira, Tarazaga, Andrade e Silva, Barberis, 2011b).

En las figuras 3, 4 y 5 se puede ver una bastante buena coincidencia entre la distribución de potencia calculada y la medida. Debe tenerse en cuenta de que se ha normalizado con el valor medio de todos los elementos medidos y no cada uno en particular.

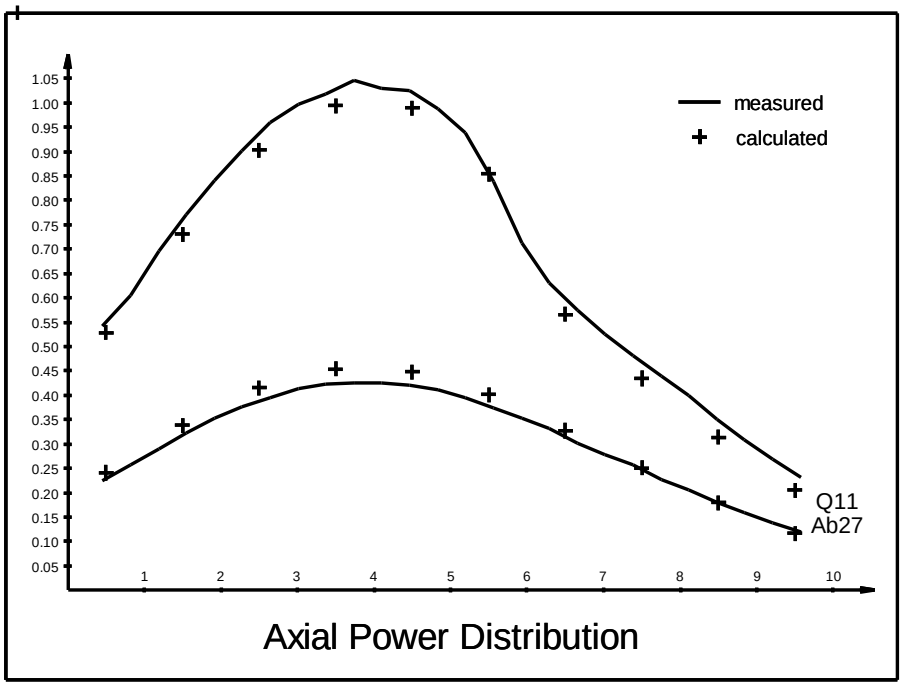

Figura 3. Distribución de potencia normalizada para valores medidos y calculados con HUEMUL-PUMA para los elementos Q11 y Ab27.

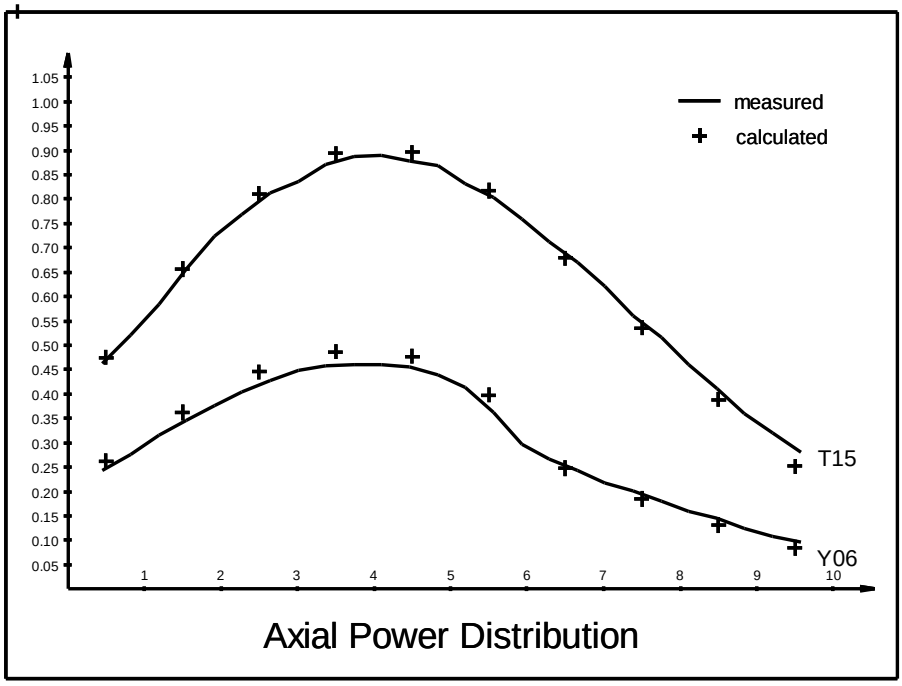

Figura 4. Distribución de potencia normalizada para valores medidos y calculados con HUEMUL-PUMA para los elementos T15 y Y06.

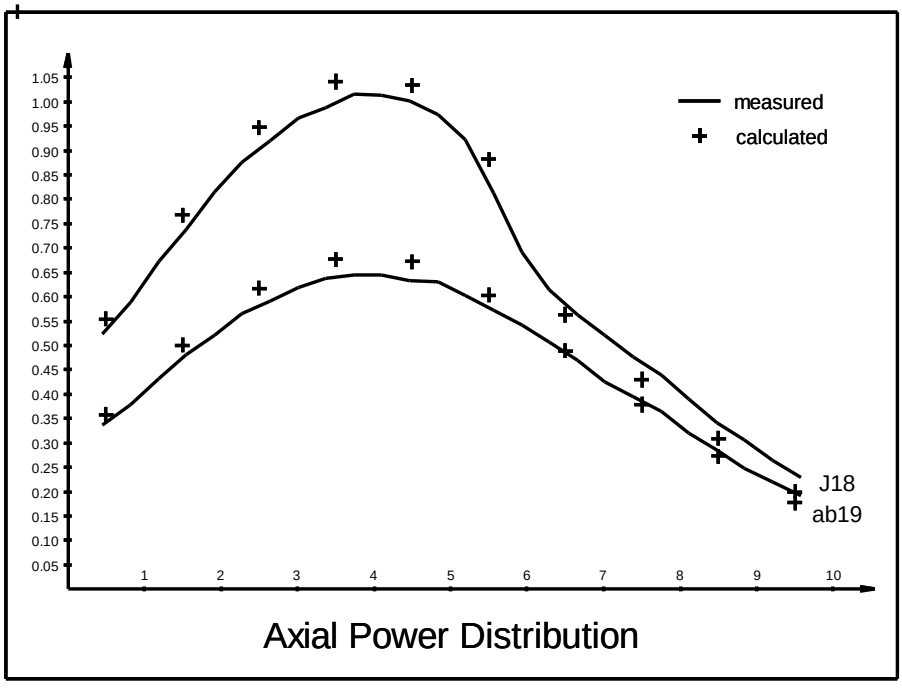

Figura 5. Distribución de potencia normalizada para valores medidos y calculados con HUEMUL-PUMA para los elementos J18 y ab19.

En cuanto a los resultados obtenidos para los parámetros cinéticos, la cadena HUEMUL-PUMA, junto a los aportados por el IPEN, se muestran en la tabla 1.

PUMA calcula estos parámetros cinéticos utilizando sus expresiones integrales (C. Grant, 2013b).

Parámetro cinético	IPEN	CNEA HUEMUL-PUMA
β nuclear	--	0.00688
β_{eff}	0.00750	0.00741
Λ [μs]	31.96	30.72

Tabla 1: Comparación de los resultados obtenidos por medio de HUEMUL y PUMA, con los obtenidos por el IPEN, para el cálculo de parámetros cinéticos (Dos Santos, Grant, Siqueira, Tarazaga, Andrade e Silva, Barberis, 2011b)

6.2 Simulación de un experimento de caída de barras en el RA1

Damos aquí el ejemplo de la simulación de una experiencia de caída de barras en el RA1. La descripción del experimento puede verse en el informe correspondiente (Tarazaga, Grant, Gómez, 2013c). En la figura 5 se puede ver un esquema del RA1 y la ubicación de los dos detectores.

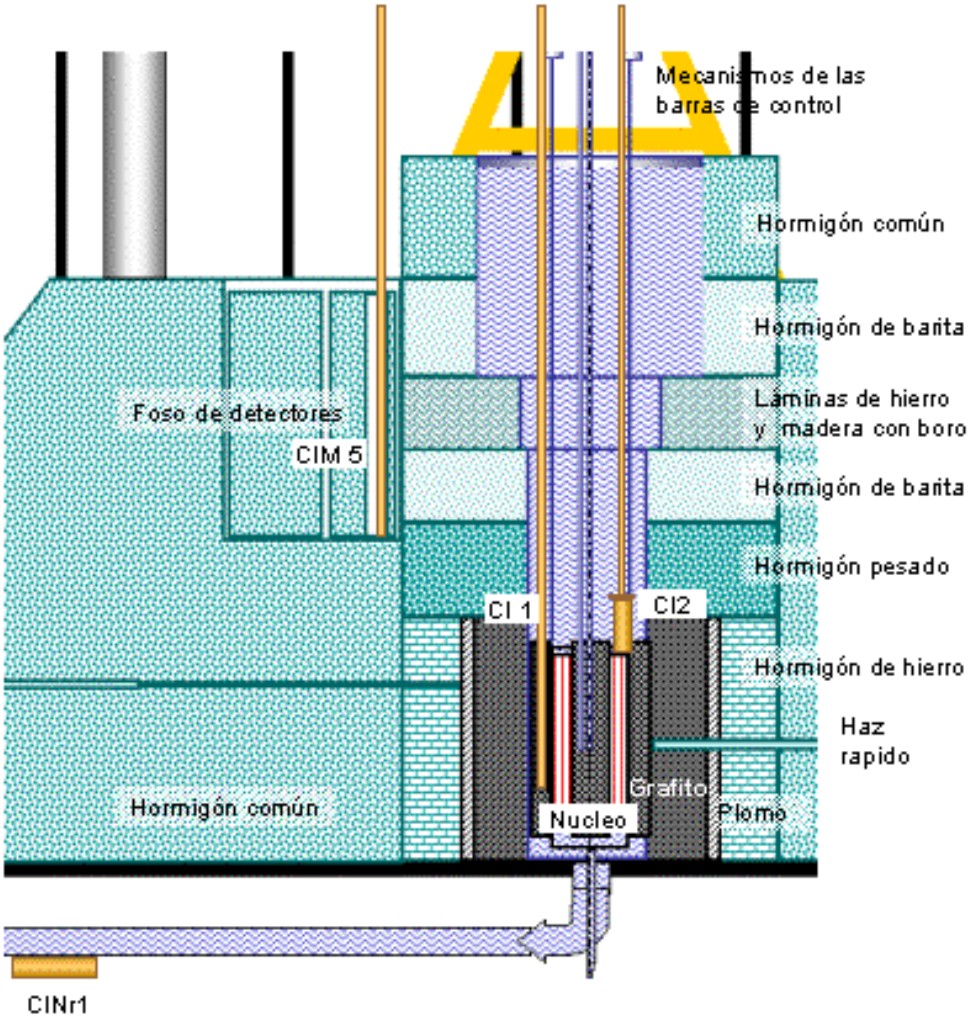

Figura 6: Ubicación de los detectores CI1 y CI2 en el reactor RA1

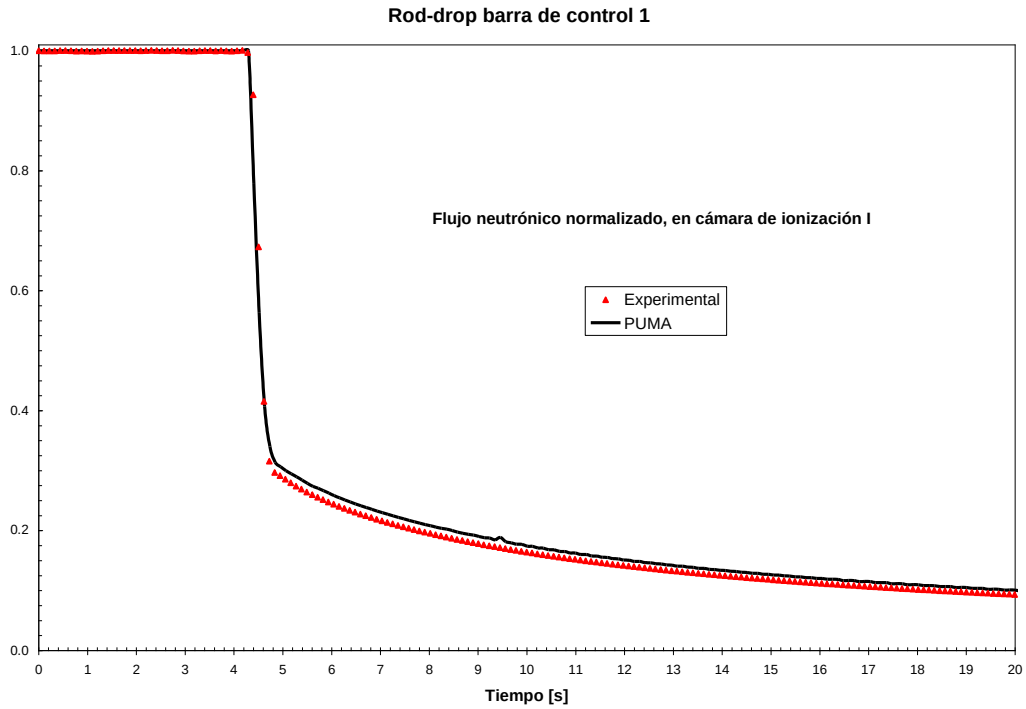

Figura 7: Valores experimentales y calculados para la caída de BC1 en el detector CI1.

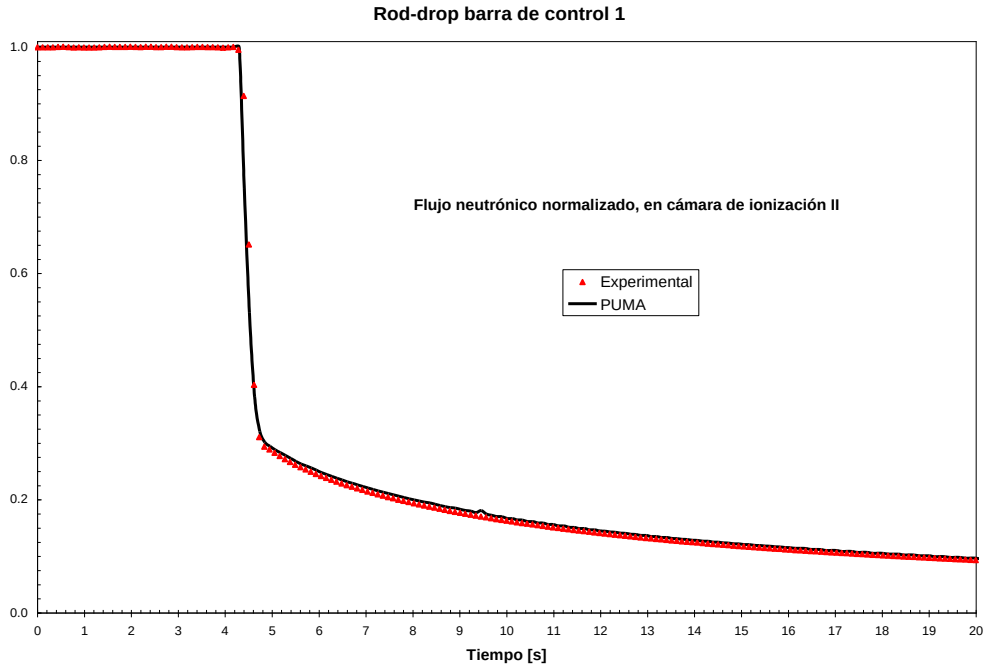

Figura 8: Valores experimentales y calculados para la caída de BC1 en el detector CI2.

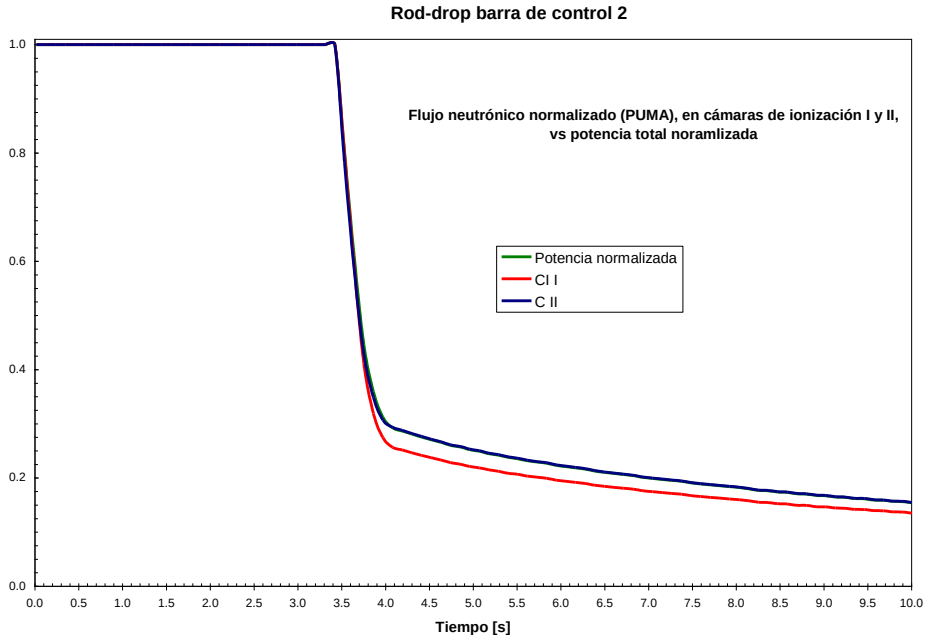

Figura 9: Valores calculados ambas cámaras
y de la potencia normalizada, para la caída de BC2.
(La curva verde y la azul coinciden)

Aquí vemos el resultado comparado entre las cámaras CI1 y CI2. Están normalizados al primer instante y se nota una diferencia muy notable entre ambas. Sin embargo la cámara CI2 coincida prácticamente en los mismos valores con la potencia total del reactor en todo instante, por lo que se puede concluir que son precisamente esos valores los que deben ser tenidos en cuenta para obtener el valor más representativo de la reactividad del reactor con todas las barras introducidas.

6.3 Simulación de un experimento hipotético de caída de barras en ATUCHA II

En la figura 10 se muestran cómo serían los resultados de experimentos de caída de barras en el reactor de ATUCHA II. Se supone la caída de todas las 18 barras y se estudian los posibles valores que producirían cuatro detectores colocados, uno en el centro del reactor y los otros tres en la zona de reflector externo abajo, en el centro y arriba.

La n(t) en la figura está dada por la expresión:

$$n(t) = \int_V \sum_g \frac{\phi_g(\vec{r},t)}{v_g} dv$$

donde V es el volumen de todo el reactor y g el número de grupo. La variable n(t) es la que aparece en las ecuaciones de la cinética puntual exacta, en las cuales los parámetros β y Λ dependen del tiempo. Según se observa en la Figura 3, el detector ubicado en P2 es que mejor evalúa el comportamiento global del reactor, ya que es la curva que más se aproxima a la de n(t).

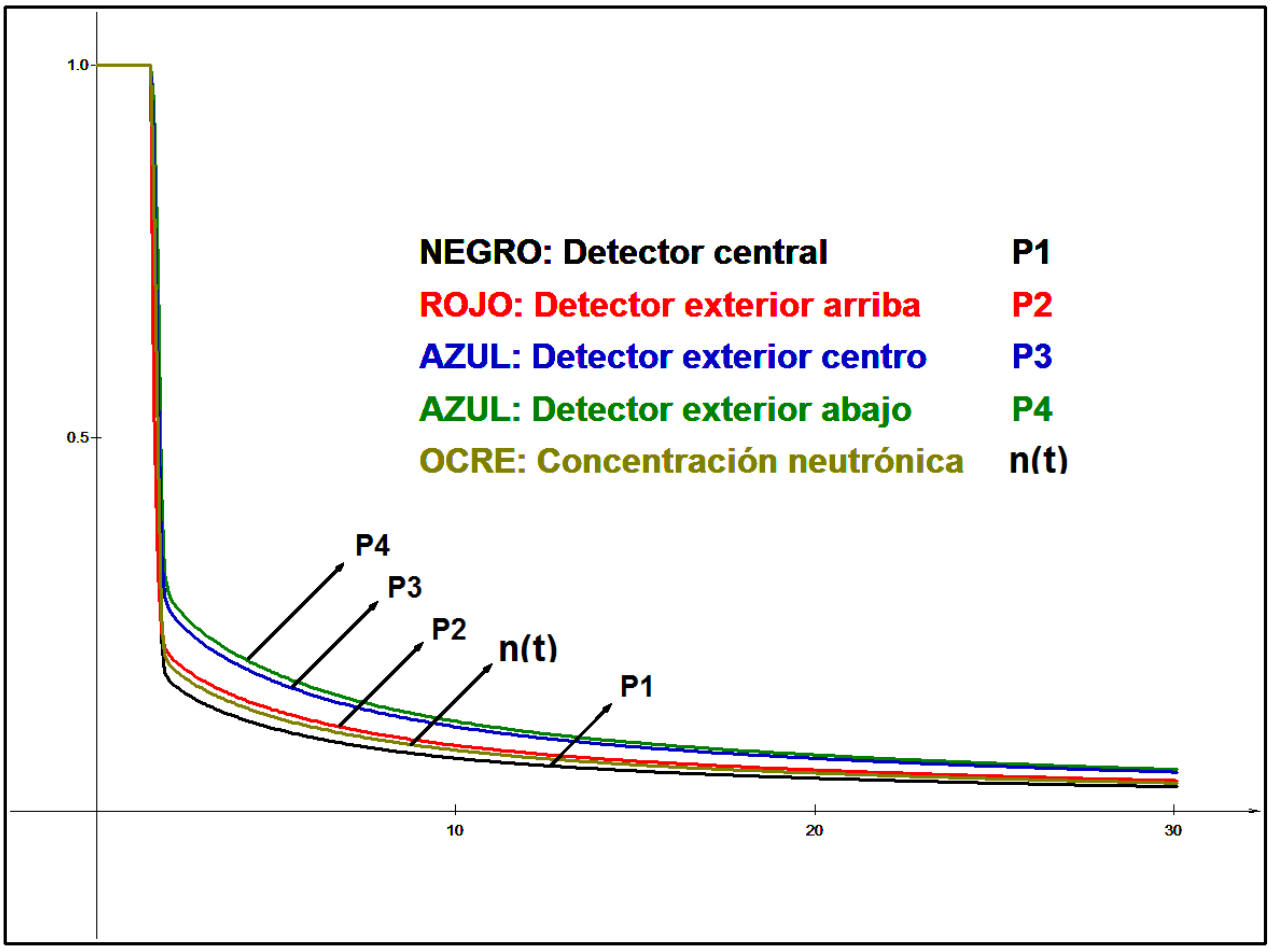

Figura 10: *Evolución de CN(t) en los primeros 30 segundos para las cuatro posiciones de los detectores, para la caída de todas las barras).*

6.4 Comparación con cálculos de MCNP para un caso de barras en ATUCHA I

Se hicieron cálculos para ATUCHA I con barras oblicuas y se compararon los resultados con MCNP (Montecarlo) y DELFIN (elementos finitos, Grant, 2014) . (Grant, Marconi, Serra, Fink, 2005).

En la tabla 2 podemos ver valores para la distribución de la potencia por canal y sus diferencias entre DELFIN , PUMA y MCNP. Las barras están introducidas un 50% (Fig. 11).

PUMA no puede evaluar correctamente los canales periféricos, por lo que su error cuadrático medio es mayor que el de DELFIN. Sin embargo se nota que en canales como el N26 y el O25 cercanos a las barras los valores de PUMA son algo mejores. Esto se debe a que PUMA modela la dirección axial con 20 trozos, lo que le da una mejor precisión.

El error en los canales periféricos se corrige en las versiones actuales de PUMA con los factores de continuidad (C. Grant, 2013b).

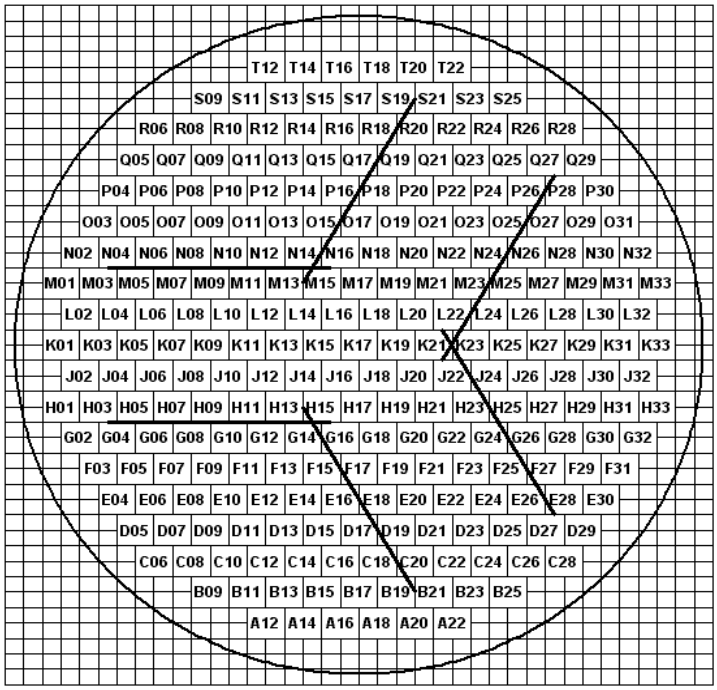

Figura 11: Posición de las barras en ATUCHA I para seis barras oblicuas en una representación cartesiana.

Canal	K17	K19	L18	K21	L20	M19	K23	L22	M21	N20	K25	L24	M23
MCNP	10.013	9.811	9.721	9.166	9.326	9.073	8.216	8.522	8.481	8.062	7.070	7.440	7.464
DELFIN	10.001	9.787	9.777	9.169	9.354	9.123	8.216	8.548	8.503	8.095	7.049	7.454	7.488
Err PUMA	-1.21	-1.33	-0.46	-1.02	-0.74	-0.40	-0.94	-0.70	-0.77	-0.46	-1.03	-0.62	-0.80
Err DELFIN	0.12	-0.24	0.58	0.03	0.30	0.56	-0.00	0.31	0.26	0.41	-0.29	0.18	0.32

Canal	N22	O21	K27	L26	M25	N24	O23	P22	K29	L28	M27	N26	O25
MCNP	7.299	6.808	5.793	6.240	6.212	5.980	5.974	5.482	4.436	4.975	5.157	4.838	4.732
DELFIN	7.307	6.823	5.774	6.240	6.231	6.036	5.961	5.478	4.424	4.970	5.156	4.896	4.728
Err PUMA	-1.05	-0.55	-0.81	-0.55	-0.11	-0.66	-1.31	-0.66	-0.40	-0.39	-0.48	0.32	-0.33
Err DELFIN	-0.11	0.22	-0.33	-0.00	0.30	0.94	-0.22	-0.08	-0.28	-0.10	-0.03	1.19	-0.09

Canal	P24	Q23	M29	N28	O27	P26	Q25	R24	K33	L32	M31	N30	O29
MCNP	4.666	4.152	3.968	4.007	3.767	3.628	3.391	2.823	1.674	2.259	2.680	2.884	2.872
DELFIN	4.640	4.144	3.956	3.996	3.811	3.649	3.366	2.816	1.661	2.249	2.662	2.870	2.857
Err PUMA	-0.93	-0.50	-0.08	-0.37	0.11	-0.18	-0.73	0.02	2.53	1.47	0.34	0.24	0.05
Err DELFIN	-0.57	-0.20	-0.31	-0.28	1.17	0.57	-0.72	-0.24	-0.80	-0.44	-0.69	-0.47	-0.51

Canal	P28	Q27	R26	S25	M33	N32	O31	P30	Q29	R28
MCNP	2.703	2.488	2.113	1.551	1.399	1.686	1.800	1.760	1.590	1.305
DELFIN	2.637	2.458	2.088	1.545	1.389	1.673	1.792	1.745	1.568	1.289
Err PUMA	0.16	0.01	0.45	2.82	2.89	2.15	2.23	2.00	1.99	2.63
Err DELFIN	-2.46	-1.21	-1.17	-0.38	-0.75	-0.79	-0.44	-0.85	-1.37	-1.21

Mean Quadratic Error DELFIN = 0.519 KEFF DELFIN 1.00027 KEFF PUMA 1.00097
Mean Quadratic Error PUMA = 0.76 KEFF MCNP 0.99826

Tabla 2: Comparación DELFIN – MCNP - PUMA para seis barras negras oblicuas.

En las figuras 12 y 13 se pueden ver las distribuciones de potencia para los canales N6 y O25 calculados por DELFIN, PUMA y MCNP, para barras negras. La coincidencia entre los tres modelos nos muestra que los modelos de DELFIN y PUMA dan una muy buena

evaluación de los canales cerca de las barras. Sólo en la figura 12 se ven diferencias algo mayores entre PUMA y MCNP. Estas diferencias no son importantes y aún el modelo de 10 trozos axiales es aceptable.

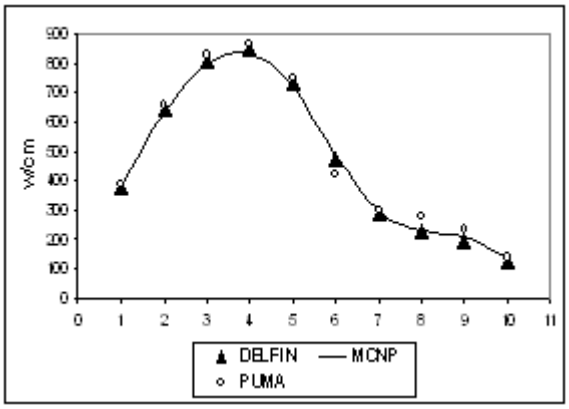

Figura 12: Canal O25, barras oblicuas

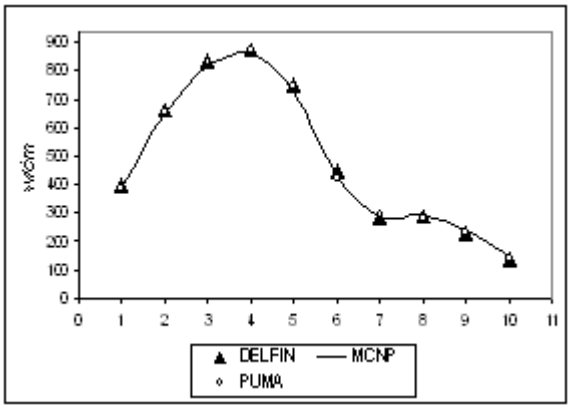

Figura 13: Canal N26, barras oblicuas

7 DESARROLLOS FUTUROS

El sistema PUMA ha tenido desde su primera versión en 1980 un uso muy intenso en todos los reactores usados en nuestro país. La gestión del combustible de la Central Nuclear de Atucha I, la Central Nuclear de Embalse y del RA3 están siendo simuladas con esta herramienta de cálculo y se ha utilizado como sistema alternativo para el CAREM. Se han hecho una gran cantidad de comparaciones exitosas de sus resultados con experiencias y está disponible una versión en lenguaje ADA que la hace adecuada para su mantenimiento futuro.

La versión actual funciona en WINDOWS o LINUX y se puede con cierta facilidad acoplar con otros modelos que complementan la descripción del funcionamiento de un reactor nuclear.

Como ocurre con cualquier tipo de software, no existe ninguna versión definitiva y todas son pasibles de perfeccionamiento o de adaptaciones a nuevas necesidades. PUMA no es la excepción y hay muchas tareas que pueden ser realizadas, por ejemplo:

a) Mejorar los procesos iterativos para una mejor convergencia. Se ha agregado un nuevo proceso iterativo basado en propiedades de los espacios de Krilov, pero no funciona bien para redes cada vez más detalladas. Es una tarea pendiente.

b) El método numérico actual es el de sobrerrelajación, que funciona correctamente pero se hace muy pesado en redes cada vez más detalladas usadas hoy en día. Habrá que reemplazarlo por otros métodos que incluso permitan una paralelización, tema en lo que se puede trabajar, tanto en el cálculo estacionario como en el cálculo de transitorios.

c) Trabajar en la implementación de la gestión de combustible para ATUCHA II con PUMA.

d) Es necesario rehacer los cálculos de la gestión de combustible de las centrales nucleares

de ATUCHA I y EMBALSE con la nueva versión. **Las versiones anteriores no tienen mantenimiento y no se puede asegurar que funcionen en las futuras versiones de WINDOWS.**

e) Implementar el acceso interno de variables en otros sistemas, por ejemplo RELAP, para poder acoplarlo más fácilmente con PUMA a través de los puertos de comunicación.

Sin embargo, el tema más importante es el futuro para que el mantenimiento de PUMA no dependa de una sola persona, sino que haya un grupo o equipo de gente que sea responsable de esa tarea. Los programas fuente están todos en un lenguaje ADA cuyo compilador puede obtenerse de INTERNET y la tarea de mantenimiento puede ser encomendada a personal profesional que parta desde una tarea ya realizada. Que no se cometa el desatino de comenzar a desarrollar una nueva herramienta que en realidad, ya está disponible.

REFERENCIAS

C. Grant - *Manual del Código HUEMUL Versión 4*, MCO-06REC-1, 2013a.

C. Grant - *Manual del Código PUMA Versión 6*, MCO-06REC-2, 2011a.

C. Grant - *Descripción de los métodos y modelos utilizados en el sistema PUMA*, MCO-06REC-2, 2013b.

Adimir Dos Santos, Carlos Grant, Paulo de Tarso D. Siqueira, Ariel E. Tarazaga, Graciete Simões de Andrade e Silva, Claudia Barberis - *Validation of Neutron Models and Calculation Systems by means of Experimental Results in the IPEN/MB-01 Reactor*, COBEN, Informe final, 2011b.

Ariel E. Tarazaga, Carlos Grant, Ángel Gómez - *Comparación Cálculo - Experiencia en Experimentos de Caída de Barras de Control en el Reactor RA-1-* AATN 2013, 2013c.

Carlos Grant, Javier Marconi, Oscar Serra and José Fink - *Comparison of Finite Element and Finite Difference for 2D and 3D Calculation with Montecarlo Results for Idealizes Cases of a Heavy Water Reactor,* 2005 International Nuclear Atlantic Conference - INAC 2005, Santos, BRASIL, 2005.

Carlos Grant - *Manual del Sistema DELFIN*, Versión 4, en preparación, 2014