GENERAL PURPOSE DYNAMIC MONTE CARLO
WITH CONTINUOUS ENERGY
FOR TRANSIENT ANALYSIS

Bart L. Sjenitzer and J. Eduard Hoogenboom
Delft University of Technology
Department of Radiation, Radionuclide and Reactors
Mekelweg 15, 2629JB Delft, the Netherlands
B.L.Sjenitzer@TUDelft.nl; J.E.Hoogenboom@TUDelft.nl

ABSTRACT

For safety assessments transient analysis is an important tool. It can predict maximum temperatures
during regular reactor operation or during an accident scenario. Despite the fact that this kind of
analysis is very important, the state of the art still uses rather crude methods, like diffusion theory
and point-kinetics. For reference calculations it is preferable to use the Monte Carlo method.

In this paper the dynamic Monte Carlo method is implemented in the general purpose Monte Carlo
code Tripoli4. Also, the method is extended for use with continuous energy. The first results of
Dynamic Tripoli demonstrate that this kind of calculation is indeed accurate and the results are
achieved in a reasonable amount of time. With the method implemented in Tripoli it is now possible
to do an exact transient calculation in arbitrary geometry.

Key Words: Monte Carlo, Neutronics, Transient, Kinetic, Dynamic

1. INTRODUCTION

When doing a transient analysis for a nuclear reactor, the state of the art is to do a determinis-
tic calculation. A deterministic method applies various approximations, such as discretization in
space, time, angle and energy[1][2]. Other approximations like diffusion theory and point kinetics
are also often used. In all cases the neutron flux is factorized in an amplitude and a shape function.
The shape function can be calculated by various means[3][4][5]. However, the coupling of these
factors is not straight forward and is still an ongoing research item[6].

When using these kind of approximations it is difficult to determine the validity of the answer.
Also, deterministic methods are problem specific. Consequently it can be challenging to apply
these methods to a unique type of reactor, such as a research reactor, which has specialized equip-
ment inside the reactor core, or a new type of reactor, like the Generation IV reactors.

Therefore a new method is being developed to use Monte Carlo simulation for transient calcula-
tions[7]. This method has been demonstrated in a purpose built Monte Carlo code[8]. It calculates
the transient in a mono-energetic homogeneous system. Also, the first steps in implementing this
method into a general purpose Monte Carlo code have been made[9]. In the present paper the
method will be further developed to apply it for more general use.
2. GENERAL DYNAMIC MONTE CARLO METHOD

In Dynamic Monte Carlo there are several issues to be addressed. The first issue is the sampling of precursors and neutrons in the same simulation. This is challenging, because neutrons and precursors live in two different time scales. The lifetime of a precursor can be between a few tenths of a second and a hundred seconds, whereas prompt neutrons live in the order of $10^{-7}$ s for fast reactors and around $10^{-4}$ s for thermal reactors. When taking into account prompt neutron chains instead of a single prompt neutron, this becomes $10^{-5}$ s and $10^{-2}$ s, respectively.

The second issue is the variance in the prompt neutron chain length. A prompt neutron chain can be finished after one collision, but also after $10^4$ collisions or more. This yields a high variance in power production per neutron chain. Also, the issues of the changing environment and the structure of a time dependent calculation must be addressed.

2.1. Simulation of Precursors

The different timescales for the different particles is a source of variance in the tallies[8]. To sample these different particles in the same time frame, the precursor particle is explicitly simulated. Instead of having delayed neutrons, a fraction $\beta$ of the new neutrons at a fission event is not a neutron, but a precursor.

The calculation is split in time intervals and these precursors are forced to create a delayed neutron in each time interval. The time when this delayed neutron is created is chosen to be uniformly distributed over the time interval. This can be done in a fair game when the statistical weight of the resulting delayed neutron is adjusted accordingly. This has been derived for a mono-energetic case[9]. A more general equation will be given in Sec. 2.1.2.

In this paper neutrons are not mono energetic and therefore the new energy of the delayed neutron has to be sampled as well. For the statistical weight of the delayed neutron the decay probabilities of all the different precursor families are used, but to sample an energy these cannot be combined. Therefore the probability of each family at the time of decay needs to be calculated. From these probabilities per family one of the families is selected and from this family the energy of the resulting neutron is sampled.

2.1.1. Spatial precursor distribution

There are two options for starting a dynamic calculation. It is possible to start from some user defined state or it is possible to start from a steady-state situation. When starting from steady state, calculating the precursor and prompt neutron distributions is an energy dependent problem. The number of neutrons in steady state is given by

$$n_0(r, E) = \frac{1}{v(E)}\phi_0(r, E)$$  \hspace{1cm} (1)

Here $n_0$ is the number of neutrons in steady state, $v$ is the velocity of the neutron and $\phi_0$ is the scalar neutron flux at steady state.
The number of precursors in steady state is given by:

$$C_{i,0}(r) = \int_{E=0}^{\infty} \frac{\beta_i(r, E)}{\lambda_i} \nu(r, E) \Sigma_f(r, E) \phi_0(r, E) dE$$  \hspace{1cm} (2)

The number of precursors of family $i$ in steady state is denoted by $C_{i,0}$. $\lambda_i$ is the decay constant of family $i$, $\beta_i$ is the delayed fraction of $i$, $\nu$ is the number of new fission neutrons and $\Sigma_f$ is the fission cross section. For all precursor families together this becomes:

$$C_0(r) = \sum_i \int_{E=0}^{\infty} \frac{\beta_i(r, E)}{\lambda_i} \nu(r, E) \Sigma_f(r, E) \phi_0(r, E) dE$$  \hspace{1cm} (3)

The number of precursors and neutrons cannot simply be calculated using the ratio between prompt neutrons and precursors in the way it has been done in the mono-energetic work[8]. However, it is possible to calculate the precursor distribution from a Monte Carlo estimation of the energy dependent neutron flux. Therefore a steady state precursor distribution can be calculated from a criticality calculation.

### 2.1.2. Time precursor distribution

For a steady-state situation the distribution between the different precursor families is different than at the time a precursor is created. Precursors are created with a

$$\frac{\beta_i}{\beta}$$

distribution. However, by definition the different precursor families have different decay constants. This ensures a higher presence of long living precursor families inside the reactor, compared to short living precursor families. In steady state the distribution of the different precursor families is given by

$$\frac{\lambda^b \beta_i}{\lambda_i \beta}$$

with the weighted average of the $\lambda_i$s:

$$\lambda^b = \frac{\beta}{\sum_j \frac{\beta_j}{\lambda_j}}$$  \hspace{1cm} (6)

The combined precursor particles can take this into account nicely by altering the starting distributions of the families. The starting distributions are denoted by the fraction delayed, $fd_i$:

$$fd_i = \begin{cases} \frac{\beta_i}{\beta}, & \text{precursor created during simulation,} \\ \frac{\lambda^b \beta_i}{\lambda_i \beta}, & \text{precursor created at start of simulation} \end{cases} \quad (t_0 > 0)$$  \hspace{1cm} (7)

Here $t_0$ is the time when the precursor was created.

A precursor also can have a statistical weight, $w_p$. This weight is treated similarly to neutron weights. If the precursor is created without biasing techniques, it is simply unity. It can be altered...
by variance reduction techniques. The weight $w_n$ of a delayed neutron after a forced decay in a
time interval of $\Delta t$ can now be calculated using:

$$w_n = w_p \Delta t \sum_i f d_i \lambda_i e^{-\lambda_i (t-t_0)}$$  \hspace{1cm} (8)$$

The expected weight of a delayed neutron will become lower over time, and if this becomes too
low, Russian roulette will be played on the precursor. If the precursor survives, its statistical weight,$w_p$, will be increased.

### 2.1.3. Selecting delayed neutron energy

When selecting the energy of the resulting delayed neutron, the different precursor families have
a different energy spectrum[10]. Since these energy spectra are given per family it is best to first
select a precursor family and then select an energy from the spectrum, applicable to that family.
The distribution of probabilities between the families changes over time and is given by:

$$\frac{P_i(t)}{P(t)} = \frac{f d_i \lambda_i e^{-\lambda_i (t-t_0)}}{\sum_j f d_j \lambda_j e^{-\lambda_j (t-t_0)}}$$ \hspace{1cm} (9)$$

### 2.2. Prompt Neutron Chains

Another large contributor to the variance in a dynamic calculation is the variance in the prompt
neutron chains. This variance is caused by the natural variance that is present in branching pro-
cesses[11]. A branch can be stopped after one collision but can also continue for many genera-
tions. This effect is especially prominent in dynamic calculations, since most dynamic calcula-
tions are done on (near) critical systems and especially near critical systems are affected by this
phenomenon[12]. In criticality calculations chain lengths are not an issue, since the particles are
simulated generation by generation and no chains are formed.

A method has been devised to reduce the variance in chain length statistics[13]. This method
has been extended and successfully introduced in the dynamic calculation. This method is called
the branchless-collision method and it is a combination of explicit absorption and implicit fission.
When using this method a neutron will at every collision either have a scattering interaction or
create a new fission neutron. If the neutron scatters, its statistical weight remains unchanged and it
continues its path. If the neutron is absorbed, one new fission neutron is born with weight:

$$w_f = w_n \frac{\nu_p(r, E) \Sigma_f(r, E)}{\Sigma_a(r, E)}$$ \hspace{1cm} (10)$$

with

$$\nu_p = (1 - \beta) \nu$$ \hspace{1cm} (11)$$

This refrains the neutron chain from splitting into multiple branches and from stopping premature;
the variance and the calculation time are reduced. Delayed neutrons are sampled in the normal
way, since a delayed neutron would terminate a prompt-neutron chain and that is undesirable.
This method works well in a purely fissile material, but it has been extended to be used in a more general case. When a neutron interacts with a purely absorbing material or a material with a small probability for fission: \( \nu_p \Sigma_f \ll \Sigma_a \) the resulting neutron can have a very low or zero statistical weight. Then the particle will be killed quickly by Russian roulette, terminating the neutron chain. The improved branchless method prevents this. In this method the resulting particle always gets a statistical weight of

\[
w_f = w_n \frac{\nu_p \Sigma_f + \Sigma_s}{\Sigma_t} \quad (12)
\]

Now the probability of having a scattering interaction becomes

\[
P_{\text{scattering}} = \frac{\Sigma_s}{\nu_p \Sigma_f + \Sigma_s} \quad (13)
\]

and for fission

\[
P_{\text{fission}} = \frac{\nu_p \Sigma_f}{\nu_p \Sigma_f + \Sigma_s} \quad (14)
\]

This extension ensures stable behavior, also for purely absorbing or purely scattering materials.

### 2.3. Calculation Scheme

The scheme for a dynamic calculation is a combination between the scheme of a criticality calculation and the scheme of a fixed-source calculation. To start the calculation a source distribution has to be specified. This can be an analytical distribution, but it is also possible to use a uniform source and calculate the correct distribution using a criticality calculation.

Next, the number of batches for the source has to be specified. This number needs to be large enough for the source to converge. At the end of this source convergence the precursor and neutron distributions are sampled during the last batch. The probability for sampling a neutron at a collision is given by:

\[
P_n = \frac{1}{v(E) \Sigma_t(r, E)} \quad (15)
\]

and for a precursor it is given by:

\[
P_C = \frac{\beta(r, E) \nu(r, E) \Sigma_f(r, E)}{\lambda(r, E) \Sigma_t(r, E)} \quad (16)
\]

The precursors are given the steady state distribution. This is the starting point of the actual dynamic calculation.

All particles synchronize their watches to \( t = 0 \) and start the dynamic part of the simulation. The particle histories are simulated time step by time step and the precursors are forced to have a decay in each of these time steps. This way all precursors will start a prompt neutron chain in each time interval and a minimum number of neutron chains per interval is guaranteed. This ensures comparable statistics in each interval.

The total number of particles is split into batches and the variance is calculated on the statistics between results of the batches. New prompt neutrons created in a fission event are also directly
followed in the same batch. This way the particles which are correlated are in the same batch, but the batches are truly independent. This also makes it easy to do the calculation in parallel.

After each time interval it is possible to change the system to incorporate dynamic behavior. These changes can be the movement of a control rod, but it can also be a change in temperature, or a LOCA scenario. The size of a time interval can be chosen freely, because the time interval size is not an approximation for the Monte Carlo simulation. Also, the time interval size can vary throughout the simulation. When coupling the Dynamic Monte Carlo scheme with a feedback code the time interval size can be important, but this is problem specific.

3. DYNAMIC TRIPOLI

The method described in the previous section has been implemented in the general purpose code Tripoli4.7[14]. To achieve this, the code had to be modified and extended. In this section the practical details of implementing this method in a general purpose code will be discussed.

3.1. New Functionalities

There are a number of new functionalities that have to be added to the code, for example: the simulation of precursors. In a traditional calculation, precursors are not simulated. In some codes the fraction delayed neutrons will simply be created at a later time, but many codes create the delayed neutron together with the prompt neutrons. Now it is needed to actually simulate the precursors. Also, the new simulation order and the calculation of the source distribution have to be implemented.

3.1.1. Precursors

Compared to other Monte Carlo particles, the precursor has a few extra features. It is needed to store the delayed fractions, \( f_{d,i} \), the decay constants and the time of creation of the precursor. These properties are all needed for the calculation of the weight of a delayed neutron. The precursor needs to remain for the next time step after the forced creation of its delayed neutron.

To make sure the delayed neutron of a precursor will not be killed directly by Russian roulette, weight monitoring is performed on the precursor. This weight monitor is not done directly, but the expected weight of the forced decay neutron is used for the weight windows. This can be calculated using:

\[
w_{n,\text{exp}} = w_p \sum_i f_{d,i} e^{-\lambda_i (t_0)} (e^{-\lambda_i t_1} - e^{-\lambda_i (t_1 + \Delta t)})
\]

Here \( t_1 \) is the starting time of the next time interval. If the weight has to be changed, it is the actual weight of the precursor, \( w_p \), which is altered.

3.1.2. Dynamic mode

The dynamic mode is different from the shielding, fixed-source or criticality modes in a few aspects. First, the fission neutrons are directly simulated in the same batch. This is different from...
a shielding calculation, where fission neutrons are not simulated, or a criticality simulation where fission neutrons are stored for the next batch.

Second, the simulation is done per time step. This implies that a neutron and its progeny are followed until the end of a time interval. If the time boundary is crossed the particle is stopped and stored for the next step. When all particles in the time interval are simulated, the simulation continues with the first particle in the next interval. At the start of the new time interval the geometry is scanned for changes in materials and temperatures and if there are any changes the new geometry and materials are used.

### 3.1.3. Eigensource

To start the calculation there are two options. A source can be given analytically or an eigenfunction can be used. In both cases the distribution of precursors and prompt neutrons must be sampled from the source distribution.

When using an eigenfunction distribution, this eigenfunction must be calculated using a traditional eigenvalue calculation. For the eigenvalue problem, precursors are simulated as ordinary neutrons, since precursors play only a role in a time-dependent problem. At the last batch the precursor and prompt neutron distributions are sampled and the time of all particles is set to zero.

### 3.2. Modifications

Next to these new features, there are also some parts which have to be modified in order to perform well in a dynamic calculation. These are existing features of a general purpose code, but special attention must be paid, and it is possible that they should be changed for optimized results in a dynamic calculation.

#### 3.2.1. Time boundary crossing

The first important feature to look into is the time boundary crossing. For most calculation the crossing of a time boundary is of no importance. In steady state the system will not change over time and therefore the cross sections remain constant. For the tallying it is only important for a track length tally to determine which part of the track was in which time bin. The particle is therefore allowed to cross a time boundary freely.

In a time dependent calculation, however, this is not the case. A system can change over time and therefore the cross sections a particle encounters can change. This can be done by stopping a particle on a time boundary in a similar way it is stopped at the boundary of a surface; the particle is stored for the new time step and in this next time step a new path length will be sampled with the updated cross sections.

#### 3.2.2. Branchless-collision method

Another aspect that needs to be handled carefully is the variance reduction techniques. In a dynamic calculation the system is usually close to critical and therefore the prompt neutron chains...
length has a high variance. This creates variance in the number of neutrons per time step and this causes variance in the tallying. Also, the calculation time can become unpredictable and long.

To implement the branchless-collision method, introduced in Sec. 2.2, it is needed to alter the collision kernel. When a neutron has an interaction, it gets a new weight, which can be calculated using Eq. 12. Then it has to be decided if the interaction is fission or a scattering according to the probability distribution of Eqs. 13 and 14.

If the interaction is a scattering interaction, the new energy and direction have to be sampled from the scattering kernel and otherwise a new fission energy and direction have to be sampled from the fission distribution.

4. RESULTS

The dynamic calculation scheme has been tested in a number of sample cases. The first case is a subcritical mono-energetic system. This example shows not only the ability of the scheme to simulate a changing neutron flux, but also that the time mesh size can vary. The second case is a mono-energetic homogeneous system which has a reactivity insertion. This demonstrates the ability to do calculations on a dynamic system.

To demonstrate the more general applicability of the method, the next system is made of $^{235}\text{U}$. The JEFF3.1.1[15] cross section set is used and this example demonstrates the possibility to do a dynamic calculation with continuous energy with real material, instead of the artificial materials used before.

Finally a demonstration is given of the calculation on a fuel assembly, where the movement of control rods is simulated. This example exhibits the potential of the Dynamic Monte Carlo method to simulate the exact geometry of a system with continuous energy.

The computation cost required is dependent on many aspects of the problem: the number of particles started, the number of time intervals, the total physical time simulated, the multiplication factor of the system, etc. Therefore it is difficult to compare computation costs, still in this paper the calculation time will be supplied as an indication of the cost of a dynamic calculation. All computation times are converted to single cpu times (CPU hours), although the calculations have been done in parallel. In fact, the method uses the intrinsic parallelization scheme of Tripoli and can therefore scale easily massively parallel calculations. The processors used were 2.3 GHz processors.

4.1. Sub-critical Calculation

The first test case is a subcritical system. The system is a small cube of 8 by 18 by 22 cm. It is made of a homogeneous artificial material, which has a total cross section of 1 cm$^{-1}$, an absorption cross section of 0.5882 cm$^{-1}$ and a fission cross section of 0.25 cm$^{-1}$. The average yield $\nu$ is 2.5 neutrons per fission and the fraction of delayed neutrons, $\beta$, is 0.685 %. The delayed fractions are given in Table I. The system starts from steady state conditions and then development of the neutron flux is calculated for 1000 s. The neutron flux is sampled in time intervals which grow exponentially in size, starting with a bin of $10^{-4}$ s. The results are plotted in Fig. 1.
Table I. The precursors are divided in six families. Here the fractions and decay constants per precursor family \(i\) are given. Also, the total delayed fraction and average decay constant are shown.

<table>
<thead>
<tr>
<th>Group</th>
<th>(\lambda (s^{-1}))</th>
<th>(\beta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0127</td>
<td>0.00026</td>
</tr>
<tr>
<td>2</td>
<td>0.0317</td>
<td>0.001459</td>
</tr>
<tr>
<td>3</td>
<td>0.1156</td>
<td>0.001288</td>
</tr>
<tr>
<td>4</td>
<td>0.311</td>
<td>0.002788</td>
</tr>
<tr>
<td>5</td>
<td>1.4</td>
<td>0.000877</td>
</tr>
<tr>
<td>6</td>
<td>3.87</td>
<td>0.000178</td>
</tr>
<tr>
<td>av/tot</td>
<td>0.0784</td>
<td>0.00685</td>
</tr>
</tbody>
</table>

Figure 1. The evolution of the neutron flux in a subcritical system, calculated with a point-kinetics calculation and with Dynamic Tripoli.

It can be seen from this figure that the dynamic mode in Tripoli can handle the changing neutron flux over time. Also, it produces correct results in all the domains. First the domain where the prompt neutrons are most important. The size of the time intervals is around the lifetime of a neutron at the beginning of the calculation. Second at the end of the calculation the size of a time bin is in the seconds scale. Here Dynamic Tripoli demonstrates that it can simulate these long term effects, dominated by the delayed neutrons, well.

The calculation has been done with 3 time intervals per decade. The total problem time was 1000 s, and the simulation started with \(10^6\) different neutrons. The total simulation time was 10 CPU hours.
4.2. Simulation of a Dynamic System

The next challenge is to do a calculation on a system with changing properties. For this a similar system has been used. The system is now 10 by 20 by 24 cm and is at the beginning very near critical. The material properties are the same as in the subcritical test. After 10 s the absorption cross section is reduced, reducing also the total cross section to $0.9988 \text{ cm}^{-1}$. The system is returned to its critical state after 40 s. The results are shown in Fig. 2.

![Graph showing power vs time](image)

Figure 2. The evolution of the neutron flux in a critical system with a reactivity insertion from 10 to 40 s, calculated with a point-kinetics calculation and with Dynamic Tripoli.

This figure shows that the Dynamic Tripoli code can handle a dynamic system and do a transient analysis. The results agree nicely with the point-kinetics results, which is to be expected in such a homogeneous system. The results were achieved in 270 CPU hours, the number of starting particles was $10^6$, and the time interval was 100 ms.

4.3. Demonstration of the Continuous Energy Model

To demonstrate the Dynamic Monte Carlo scheme in a more realistic setting, while keeping it simple enough to compare it with the point-kinetics model, the cuboid of the previous sections is made of pure $^{235}\text{U}$. The density has been adjusted to make a critical system and the reactivity insertion is generated by a increase in density. Although there will not be a lot of moderation, the energy dependence of the cross sections will play an important role in this problem. The density has been set to $0.044925$ atoms per $10^{-24} \text{ cm}^3$ and increased to $0.04500$ atoms per $10^{-24} \text{ cm}^3$.

The results of this calculation are shown in Fig. 3 and the results of a point-kinetics calculation for the same problem are also given in this plot. It demonstrates the possibility of solving a continuous energy transient problem with the Dynamic Monte Carlo method. This calculation took 1152 CPU hours and had $10^7$ starting particles, with 100 ms time intervals.
4.4. A Transient in a Fuel Assembly

Finally there is a demonstration of a transient analysis of a fuel assembly. This assembly consists of 17 by 17 pins. At certain locations the fuel pin is replaced by a control rod. The center pin is replaced by a guide tube for measurement equipment. The fuel consists out of uranium oxide and the control rods are boron based. The layout of the fuel assembly is shown in Fig. 4. Reflective boundary conditions are used for the side boundaries and vacuum is assumed on the top and bottom. The model is based on the transient benchmark of Kozlowski and Downar[16], but some simplifications are used and there is no feedback present.

The total neutron flux is shown for a time evolution of 10.0 s. The control rods are slowly moved at $t = 3.0$ s, increasing the reactivity of the system until $t = 3.5$ s. Then the rods are stopped again until there is a scram at $t = 7.5$ s. The calculated power production is plotted in Fig. 5 and it shows a critical system to begin with. Then there is a prompt jump when the control rods start to pull up. Also the scram can be clearly seen, with the residual power production from the precursor source.

It is also seen that the variance between the different time steps is high during the higher reactivity. This can be explained by the extra variance that is caused by the longer prompt neutron chain length. After the scram the variance is still high, but the variance between the different time steps is smaller. This can be explained by the fact that there is less variance on the prompt chain lengths, but there is still variance in the delayed source. This variance is the same for all the time steps, there is a strong correlation between those time intervals.

The calculation cost of this problem was 900 CPU hours, with $10^6$ starting particles and a time interval of 100 ms.
(a) Vertical cross section of fuel assembly, (b) Horizontal cross section of fuel assembly x- and z-axis are not in the same scale.

Figure 4. Layout of a fuel assembly, the fuel is colored red, the control rods are in green, the borated water is in blue, the top and bottom are in black and the cladding is in gray.

Figure 5. The evolution of the neutron flux in a fuel assembly with a reactivity insertion from 3.0 to 3.3 s and a scram after 7.5 s, calculated with the Dynamic Monte Carlo method.
5. CONCLUSIONS AND RECOMMENDATIONS

In this paper the Dynamic Monte Carlo method is further developed. An extension of the method to incorporate continuous energy is described. Also, the dynamic method is now implemented in the general purpose code Tripoli. A demonstration of the dynamic mode in Tripoli is given and this is done in a simple system to be able to compare the results with a point-kinetics calculation. The results agree nicely.

Since a Monte Carlo code is governed by local equations, like the scattering probability and angles and not by the meshing, it is not very complicated to apply this method to more complex geometries. A demonstration is given of a simulation of a complex geometry. It demonstrates the feasibility to do a simulation in an arbitrary geometry. The next step is to compare the results with a deterministic calculation. Afterwards some form of feedback can be coupled to this method to make the simulation truly dynamic. The computational cost would most likely scale linear with its size, therefore if a calculation would be done on a core of 193 fuel assemblies, the computation time would go up to 7237 computation days. This seems to be a lot, but Monte Carlo parallelizes very well and although a full-core calculation would be expensive, it is not imperceivable.

ACKNOWLEDGEMENTS

This work is partially funded by the Integrated Project NURISP (contract nº 232124) in the 7th Euratom Framework Programme of the European Union.

REFERENCES


