Annals of Nuclear Energy 87 (2016) 282–289

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Technical note

Solving the multigroup integro-differential equation of the neutron diffusion kinetics in 3D-Cartesian geometry

Barbaro Quintero-Leyva *

119 SW 6th Ave., Miami, FL 33130, USA

ARTICLE INFO

Article history:
Received 10 August 2015
Received in revised form 5 September 2015
Accepted 8 September 2015

Keywords:
Space–time kinetics
Integro-differential equation
Neutron diffusion
Lagrange interpolation
Progressive polynomial approximation
Finite difference

ABSTRACT

The multigroup time-integro-differential equation of the neutron diffusion kinetics (IDE-NDK) was solved numerically in 3D Cartesian geometry with the use of the basic-progressive polynomial approximation (BP_n) using the finite difference method (FDM) for the spatial discretization. Applications involving ramp and instantaneous change of the thermal removal or fission macroscopic cross sections were used to assess the accuracy of the BP_2 algorithm. Two reactor models were used: a homogeneous and the 3D-TWIGL reactor. The BP_2 algorithm showed good accuracy when it is compared to the results of other codes. Also the static neutron diffusion equation was solved numerically with the Lagrange interpolation polynomial to assess the K_{eff} accuracy of the FDM used for the steady state problem. In some applications calculations were performed as function of the time integration step and mesh size. Extrapolations to infinitesimal mesh size were performed in some cases.

1. Introduction

The multigroup time-integro-differential equations of the neutron diffusion kinetics (IDE-NDK) was solved numerically in 3D Cartesian geometry using the basic-progressive polynomial approximation (BP_n) for the time discretization and the standard finite difference method for the spatial discretization.

The BP_n algorithm, presented in Quintero-Leyva (2009) for point kinetics and implemented in Quintero-Leyva (2010, 2012) for solving the IDE-NDK in 1D and 2D Cartesian geometry, is extended in this work to 3D. The objective of this manuscript is to make an assessment of the BP_2 algorithm (in the context of a finite difference-spatial discretization) in 3D kinetics problems where the requirements of memory and/or computing time could be very challenging when it is compared to lower dimensional problems. To address the CPU time requirements a customized (for the sparse matrices in question) Gauss–Seidel method was implemented which saved significant amount of CPU time when compared to the non-customized Gauss–Seidel or to the Gauss elimination method used in the works just mentioned.

Applications involving a ramp, and an instantaneous change of the thermal removal or fission macroscopic cross sections were analyzed. Two reactor models were used: a homogeneous reactor and the 3D-TWIGL reactor. The BP_2 algorithm showed good accuracy when compared to the results of other codes. In some applications the calculations were performed as function of the time integration step and the mesh size and for some cases extrapolations to infinitesimal mesh size were performed.

The static neutron diffusion equation was solved numerically with the Lagrange interpolation polynomial to assess the K_{eff} accuracy of the spatial finite difference method used for the steady state condition.

The essence of the BP_n algorithm as applied to the solution of the 3D IDE-NDK equation is as follows:

(a) Calculate the steady state flux and K_{eff}, and determine the target degree of the polynomial approximation.
(b) Solve the IDE-NDK for the first interval of time using a linear approximation (the unknown flux is expressed in term of the steady state flux using a 1st degree polynomial interpolation equation)
(c) Increase the degree of the polynomial approximation and solve the IDE-NDK for the next time interval (e.g. for a 2nd degree polynomial the unknown flux is expressed in terms of the already calculated flux for the last two time-intervals using the 2nd degree polynomial interpolation equation).

* Tel.: +1 786 301 5915.
E-mail address: doserate2002@yahoo.com

http://dx.doi.org/10.1016/j.anucene.2015.09.005
0306-4549/

Repeat step c until the target polynomial degree and simulation time is reached.

Note that, unlike in other kinetics methods, the BP$_n$ algorithm does not need iterations intrinsic to the kinetics, it uses already calculated values of the flux to extrapolate (using the integro-differential equation) to the next time interval, it does not need to solve special system of algebraic linear equations (SALE) inherent to the kinetics method (the interpolation equation solutions are well known, for higher than a 2nd degree polynomial, it can easily be solved with Lagrange or Newton interpolation polynomial that can be reduced to basic polynomial), it does not use any series expansion, it does not time-integrate the prompt neutron equation, and it does not need time-numerical integration (all time-integrals are exact and simple).

2. The IDE-NDK

By time-integrating the equations of the delayed neutron precursor concentrations, the multi-group NDK equation without external source (Aviles et al., 1991) can be written with the following single multigroup integro-differential equation (Quintero-Leyva, 2010):

$$\frac{\partial \phi_g}{v_g \partial t} = \nabla \cdot D_g \nabla \phi_g - \Sigma_{R,g}\phi_g + S_g^S + S_g^p + \frac{1}{K_{eff}} \sum_{l=1}^{Nd} \chi_{gl}^d \lambda_l \beta_l e^{-\lambda_l t} \left(\int_0^t \sum_{g'=1}^{G} \nu\Sigma_{F,g'}\phi_{g'}\ e^{\lambda_l t'}\ dt' + \frac{p_0}{\lambda_l} \right) \quad (1)$$

where,

$\Sigma_{R,g}$ represents the neutron macroscopic removal cross section (absorption and out scatters).

$$S_g^S = \sum_{1,g' \neq g}^{G} \Sigma_{S,g' \to g}\phi_{g'} \quad S_g^p = \frac{(1-\beta)\chi_g^p}{K_{eff}} \sum_{g'=1}^{G} \nu\Sigma_{F,g'}\phi_{g'}$$

$$p_0 = \sum_{g=1}^{G} \nu\Sigma_{Fg}\phi_g \text{ at } t = 0$$

K_{eff} is the eigenvalue of the steady state problem (t = 0) which is a constant for Eq. (1). Using this eigenvalue and the corresponding steady state neutron flux as input, allows the direct use of the neutron cross section of the steady state problem without an initial adjustment of the NDK equation to assure critical state as the initial condition.

The rest of the terms are well described in the literature. The independent variables were dropped for convenience where possible.

2.1. Determination of the initial condition and K_{eff}

The initial condition (steady state flux distribution) and K_{eff} is obtained by solving the typical multigroup static neutron diffusion equation (1st 4 terms of Eq. (1) with β = 0), which in 3D Cartesian geometry is written as

$$-\frac{\partial}{\partial x}\left(D_g(x,y,z)\frac{\partial \phi_g(x,y,z)}{\partial x}\right) - \frac{\partial}{\partial y}\left(D_g(x,y,z)\frac{\partial \phi_g(x,y,z)}{\partial y}\right) - \frac{\partial}{\partial z}\left(D_g(x,y,z)\frac{\partial \phi_g(x,y,z)}{\partial z}\right) + \Sigma_{R,g}(x,y,z)\phi_g(x,y,z) = S_g(x,y,z) \quad (2)$$

$$S_g(x,y,z) = \sum_{g' \neq g}^{G} \Sigma_{S,g' \to g}(x,y,z)\phi_{g'}(x,y,z) + \frac{1}{K_{eff}}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{F,g'}(x,y,z)\phi_{g'}(x,y,z)$$

which is solved by integrating Eq. (2) between $x_i - \frac{\Delta_i}{2}$ and $x_i + \frac{\Delta_{i+1}}{2}$ in the x-direction between $y_j - \frac{\Delta_j}{2}$ and $y_j + \frac{\Delta_{j+1}}{2}$ in the y-direction, and between $z_k - \frac{\Delta_k}{2}$ and $z_k + \frac{\Delta_{k+1}}{2}$, applying continuity of current and flux, using a finite difference (FDM) for the 1st derivative discretization, and considering average values of the nuclear constants inside each cell (cube). The following system of algebraic linear equations (SALE) is obtained:

$$b_{g,i-1,j,k}\phi_{g,i-1,j,k} + b_{g,i,j-1,k}\phi_{g,i,j-1,k} + b_{g,i,j,k-1}\phi_{g,i,j,k-1} + b_{g,i,j,k}\phi_{g,i,j,k} + b_{g,i+1,j,k}\phi_{g,i+1,j,k} + b_{g,i,j+1,k}\phi_{g,i,j+1,k} + b_{g,i,j,k+1}\phi_{g,i,j,k+1} = S_{gi,j,k} \quad (3)$$

where,

$\phi_{gi,j,k}$ and similar are the unknowns (nodal/point quantities, they are not averages over the node).

i, j, k are indexes of the x, y, z coordinates respectively (i = 1, …, I − 1; j = 1, …, J − 1; k = 1, …, K − 1).

I, J, K are the number of partitions in x, y, and z coordinates respectively.

$$b_{g:i-1,j,k} = -\frac{D_{g,i,\underline{jk}}}{\bar{\delta}_i \Delta_i} \quad b_{g:i,j-1,k} = -\frac{D_{g,j,\underline{ik}}}{\bar{\delta}_j \Delta_j} \quad b_{g:i,j,k-1} = -\frac{D_{g,k,\underline{ij}}}{\bar{\delta}_k \Delta_k}$$

$$b_{g:i,j,k} = \frac{D_{g,i,\underline{jk}}}{\bar{\delta}_i \Delta_i} + \frac{D_{g,i+1,\underline{jk}}}{\bar{\delta}_i \Delta_{i+1}} + \frac{D_{g,j,\underline{ik}}}{\bar{\delta}_j \Delta_j} + \frac{D_{g,j+1,\underline{ik}}}{\bar{\delta}_j \Delta_{j+1}} + \frac{D_{g,k,\underline{ij}}}{\bar{\delta}_k \Delta_k} + \frac{D_{g,k+1,\underline{ij}}}{\bar{\delta}_k \Delta_{k+1}} + \overline{\Sigma}_{Rg,i,j,k}$$

$$b_{g:i+1,j,k} = -\frac{D_{g,i+1,\underline{jk}}}{\bar{\delta}_i \Delta_{i+1}} \quad b_{g:i,j+1,k} = -\frac{D_{g,j+1,\underline{ik}}}{\bar{\delta}_j \Delta_{j+1}} \quad b_{g:i,j,k+1} = -\frac{D_{g,k+1,\underline{ij}}}{\bar{\delta}_k \Delta_{k+1}}$$

$$S_{gi,j,k} = \sum_{g' \neq g}^{G} \overline{\Sigma}_{S,g' \to g,i,j,k}\phi_{g',i,j,k} + \frac{1}{K_{eff}}\chi_g \overline{\nu\Sigma}_{F,g',i,j,k}\phi_{g',i,j,k}$$

$$D_{g,i,\underline{jk}} = \frac{D_{g,i,j,k} + D_{g,i,j+1,k} + D_{g,i,j,k+1} + D_{g,i,j+1,k+1}}{4}$$

$$D_{g,i+1,\underline{jk}} = \frac{D_{g,i+1,j,k} + D_{g,i+1,j+1,k} + D_{g,i+1,j,k+1} + D_{g,i+1,j+1,k+1}}{4}$$

$$D_{g,j,\underline{ik}} = \frac{D_{g,i,j,k} + D_{g,i+1,j,k} + D_{g,i,j,k+1} + D_{g,i+1,j,k+1}}{4}$$

$$D_{g,j+1,\underline{ik}} = \frac{D_{g,i,j+1,k} + D_{g,i+1,j+1,k} + D_{g,i,j+1,k+1} + D_{g,i+1,j+1,k+1}}{4}$$

$$D_{g,k,\underline{ij}} = \frac{D_{g,i,j,k} + D_{g,i+1,j,k} + D_{g,i,j+1,k} + D_{g,i+1,j+1,k}}{4}$$

$$D_{g,k+1,\underline{ij}} = \frac{D_{g,i,j,k+1} + D_{g,i+1,j,k+1} + D_{g,i,j+1,k+1} + D_{g,i+1,j+1,k+1}}{4}$$

$$\overline{\Sigma}_{Xg,i,j,k} = \frac{\Sigma_{X,gi,j,k} + \Sigma_{X,gi,j,k+1} + \Sigma_{X,gi,j+1,k} + \Sigma_{X,gi,j+1,k+1} + \Sigma_{X,gi+1,j,k} + \Sigma_{X,gi+1,j,k+1} + \Sigma_{X,gi+1,j+1,k} + \Sigma_{X,gi+1,j+1,k+1}}{8}$$