Reactor Physics: Numerical Methods

prepared by
Wm. J. Garland, Professor, Department of Engineering Physics,
McMaster University, Hamilton, Ontario, Canada

More about this document

Summary:

The one speed neutron diffusion equation for fixed sources is solved numerically. First the time component is investigated to show how simple explicit and implicit numerical treatments can be used to intuitively and effectively advance the time solution. Then the diffusion term is modeled by a central point difference scheme and steady state solution techniques are investigated. Finally, space and time are brought together and a simple numerical scheme is offered. This offers a solid basis for reactor modeling (non-fixed sources) that will come later.

Table of Contents

List of Figures

List of Tables

Error! No table of figures entries found.

1 Introduction

1.1 Overview

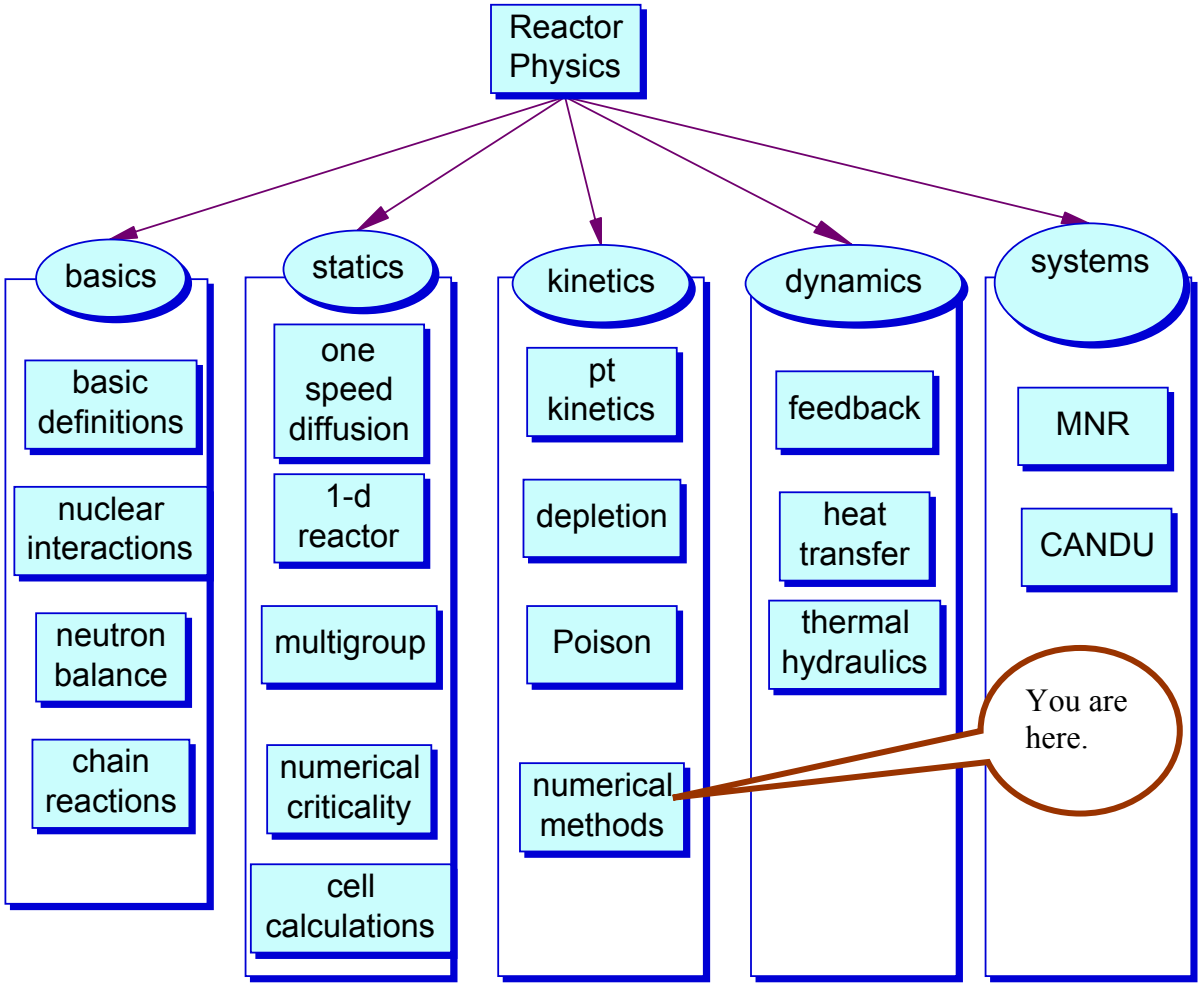


Figure 1 Course Map

1.2 Learning Outcomes

The goal of this chapter is for the student to understand:

- The simple Euler time integration method for modeling transients
- The advantages and disadvantages of the explicit and implicit treatments of numerical approximations to differential equations
- The centered-difference discretation for modeling diffusion
- Specialized methods for numerically calculating the steady state and transient flux as a function of position.

2 Modelling Time - Simple Ordinary Differential Equations (ODEs)

2.1 The Explicit Euler Method

Consider the simple equation:

$$\frac{dY}{dt} = -\alpha Y, \text{ where } \alpha = \alpha(t), \text{ in general.} \quad (1)$$

Y represents some unknown, say a nuclide concentration, subject to the initial condition Y(t)=Y(0) at time, t=0. This equation arises in the attenuation of a particle beam of intensity I, ie:

$$\frac{dI}{dx} = -\Sigma I, \text{ where } \Sigma = \Sigma(x), \text{ in general.} \quad (2)$$

This equation also arises in radioactive decay:

$$\frac{dN}{dt} = -\lambda N, \text{ where } \lambda = \text{constant.} \quad (3)$$

When the coefficient (α, Σ or λ) is a constant, we can solve the equation analytically. But, in general, we must resort to numerical methods. We'll see the time derivative arise often in this course, particularly in the time-dependent neutron diffusion equation and in the point kinetics equations. Only rarely can we solve these equations analytically. So we need to know how best to deal with transient equations (whether space dependent or space independent) numerically.

To construct a solution strategy, we note that equation 1 is a simple rate equation that gives directly the rate of change of Y. Since we have the initial value of Y, we can project what the value of Y will be at the next instant in time, as depicted in figure 2.

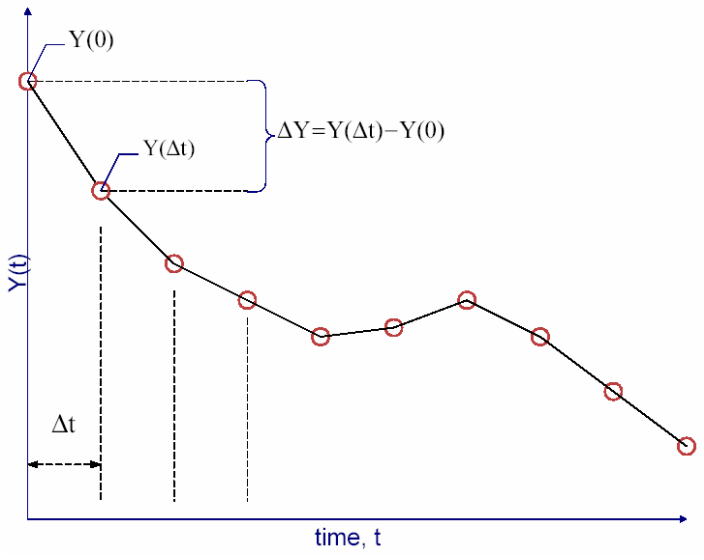


Figure 2 Y as a function of time, t.

This is intuitively satisfying and is mathematically equivalent to the Taylor series expansion for a finite step size:

$$Y(t+\Delta t) = Y(t) + \left(\frac{dY}{dt}\right)\cdot\Delta t + \frac{1}{2!}\left(\frac{d^2Y}{dt^2}\right)\cdot(\Delta t)^2 + \ldots + \frac{1}{k!}\left(\frac{d^kY}{dt^k}\right)\cdot(\Delta t)^k + \ldots \quad (4)$$

which, when we solve explicitly for dY/dt, we have:

$$\frac{dY}{dt} = \frac{Y(t+\Delta t) - Y(t)}{\Delta t} + O(\Delta t)^2 \quad (5)$$

Now, the slope of Y, dY/dt, is known explicitly from equation 1 since we know the value of the right hand side. For a sufficiently small Δt, then, we can drop the higher order terms in equation 5 and substitute equation 4 into equation 1. Rearranging, we have:

$$Y(t+\Delta t) = Y(t) - \Delta t\,\alpha Y \quad (6)$$

So we can advance our solution step by step, using the value of Y at t and the estimate of the slope at t to calculate the value of Y at t+Δt. The last term on equation 6 requires an evaluation of Y. The simplest approach is to use the value at t, ie Y(t). Thus we have:

$$Y(t+\Delta t) = Y(t) - \Delta t\,\alpha Y(t) = (1-\Delta t\alpha)Y(t)$$

or (7)

$$Y^{t+\Delta t} = (1-\Delta t\alpha)Y^t, \text{ using a more compact notation.}$$

This is the **explicit** method - ie, the future value of Y is calculated based only on parameters whose values are known at time t. The **Euler** method involves the use of onle the fist term in the Taylor's Series expansion, dropping all higher order terms. So the above method is the Explicit Euler method.

This is a good time to make an important observation about the explicit method as portrayed by equation 7. If the coefficient $\alpha > 0$ then for a sufficiently large Δt the bracketed term goes negative and the solution in time will oscillate in sign and grow in magnitude in an unbounded fashion. We call this an unstable solution. Now, we know that when $\alpha>0$, the solution decays uniformly so we know that this unstable solution is clearly incorrect. Figure 3 depicts this situation.

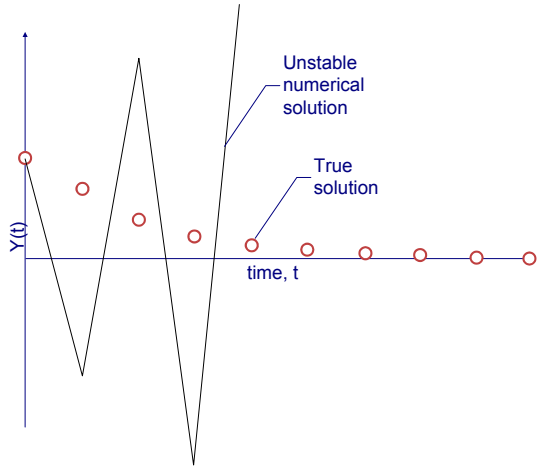


Figure 3 Example of an unstable explicit solution.

If $\alpha<0$, the true solution would grow and the numerical solution would be stable (though inaccurate) even for large Δt. So we place a restriction on Δt to ensure stability:

$$\left|\Delta t\,\alpha\right| < 1 \tag{8}$$

You may need to further restrict Δt to ensure an accurate solution. If α is a large positive number, say for the case of a rapidly decaying species, then Δt would have to be tiny. This means that many small steps would need to be taken to simulate the decay of the species. This would not be a problem if we were only interested in this one species because we could stop the calculation once the concentration approached zero. But what about the case where we are modeling a decay chain in which some of the species are long lived (ie, slowly decaying) and some are short lived (rapidly decaying)? We would be saddled with computing in tiny Δt's for a very long time. Such a system is called a stiff system. There are a number of solutions to this problem. One, which involves altering our physical model to make it less stiff, is discussed in the chapter *Solving Stiff Reactor Physics Problems in a Physically Intuitive Way*. But the classical solution is to alter the mathematics as follows.

2.2 The Implicit Method

If we were to evaluate Y associated with the α coefficient in the right hand side of equation 7 at the future time, we get:

$$\begin{aligned} &Y(t+\Delta t) = Y(t) - \Delta t\,\alpha Y(t+\Delta t) \\ &\text{or} \\ &(1+\Delta t\alpha)Y^{t+\Delta t} = Y^{t} \end{aligned} \tag{9}$$

which gives:

$$Y^{t+\Delta t} = (1+\Delta t\alpha)^{-1} Y^{t} \tag{10}$$

This is the **Implicit** integration method. Usually the coefficient, α, is much more slowly varying than Y so it can be evaluated explicitly. We see that for the case where $\alpha>0$, the solution is stable, ie the solution does not grow unbounded or oscillate, for any positive Δt. If $\alpha<0$, the true solution is an exponential growth and the implicit method is potentially unstable but since we'd want to take a small Δt to ensure an accurate solution, this is usually not a problem. So the implicit approach solves the stability problem. But it comes at a cost, as we shall see in a minute.

Before we do, let's look at the analytical solution for constant α and compare it to the numerical solutions:

$$Y(t) = Y(0)e^{-\lambda t} = Y(0)(1-\lambda t+...) \tag{11}$$

If we are expanding about a single time step (ie $t = \Delta t$) and we do a truncation of the higher order terms in the expansion of the exponential, we see that the explicit solution is similar to the analytical solution. This is comforting.

If we were to rearrange equation 10 slightly as follows,

$$e^{+\lambda t}Y(t) = Y(0) = (1+\lambda t+...)Y(t)$$

ie

$$Y(t) = Y(0)(1+\lambda t+...)^{-1} \tag{12}$$

So we see that the implicit form is also a valid approximation of the analytical solution.

2.3 The Crank Nicolson Method

We can mix the two forms (called the Crank-Nicolson method):

$$Y^{t+\Delta t} = Y^t - \Delta t\lambda Y^t(1-\theta) - \Delta t\lambda Y^{t+\Delta t}(\theta) \tag{13}$$

where θ is a weighting factor whose value is between 0 and 1, ie $\theta \in (0,1)$. Solving for the unknown $Y^{t+\Delta t}$:

$$Y^{t+\Delta t} = \frac{(1-\Delta t\lambda(1-\theta))Y^t}{(1+\Delta t\lambda\theta)} \tag{14}$$

We can vary θ to get a blend of the explicit and implicit methods as desired. Setting $\theta = 0.5$ simulates using an evaluation of αY at mid step, which is probably the most accurate value to use. Just make sure that the numerator does not go negative.

2.4 Systems of ODE's and the computational tradeoffs – wherein it is better to be effective than efficient

Now for the downside of the implicit method. For <u>systems</u> of O.D.E.'s the single variable Y become a vector **Y** and we can no longer simply divide through by the implicit factor. Now we have to do a matrix inversion or something equivalent to that, ie:

$$\left[\mathbf{I}+\Delta t\lambda\theta\mathbf{I}\right]\mathbf{Y}^{t+\Delta t} = \left[\mathbf{I}-\Delta t\lambda(1-\theta)\mathbf{I}\right]\mathbf{Y}^t, \text{ where } \mathbf{I} \text{ is the identity matrix.} \tag{15}$$

This is a $\mathbf{Ax} = \mathbf{b}$ type problem and any of the standard matrix solvers can be used to get **x.** So in any practical problem, the implicit method offers stability at a computational cost. If α varies as the solution proceeds (as it usually does), then we must perform the matrix inversion at each time step. This can get very costly for large systems involving many thousands of equations that typically arise in reactor physics. In addition, the coding is more complex (and, therefore, error prone) than the explicit method.

The user will have to evaluate whether it is worthwhile coding up the implicit method or not. In practical problems, although it may be enticing to use the implicit method so that big Δt's can be taken, there is often some physical aspect of the situation to be simulated, such as control rod movement, thermalhydraulic feedback, etc, that will place an upper limit on the Δt for simulation fidelity reasons. In that case, you may end up spending time and resources for a benefit that cannot be realized. You will have to decide on a case by case basis.

You should now be able to handle any set of ODE's and crank out an answer. I always like to keep the simple Euler explicit transient method in my back pocket as a quick and dirty method

that will always work provided Δt is restricted. You will see in ??? that we can usually get around the stiff nature (and consequential long computation times inherent in the explicit method) by a judicious manipulation of the physical model.

One thing to always keep in mind when embarking on a simulation task. The job is usually to solve a problem, conduct a design, do an analysis, etc. There is a bigger context than just writing a code and crunching some numbers. Data gathering, quality assurance, team meetings, setting up the physical model, etc. are all part of the overall job. It is this overall job that needs to be optimized, not any one specific part. So it makes little sense to spend days or weeks in building a beautifully efficient numerical scheme that will save hours of computation time when the computation is just a small portion of the overall job that may span weeks or months. Better to spend the time on ensuring the physical model is accurate and that the right questions are being asked overall (like, why is the computation really needed and to what accuracy).

It is far better to be effective than efficient.

3 Modelling Space

3.1 The Grid Point and it Nearest Neighbours

We now consider how we numerically model the spatial aspects of a typical nuclear reactor core. For the purposes of discussion, we will consider the steady state, one group neutron diffusion in 2 dimensions with unequal grid spacing and D(**r**) and Σ(**r**). Extension to 3D or simplification to 1D is trivial.

The general one group neutron diffusion equation was given in the chapter on *The Diffusion of Neutrons* as:

$$\frac{\partial}{\partial t} n(\mathbf{r},t) = \frac{1}{v}\frac{\partial}{\partial t}\phi(\mathbf{r},t) = S(\mathbf{r},t) - \Sigma_a(\mathbf{r})\phi(\mathbf{r},t) + \nabla\bullet D(\mathbf{r})\nabla\phi(\mathbf{r},t) \tag{16}$$

which in the steady state is:

$$0 = S(r) - \Sigma_a(r)\phi(r) + \nabla\bullet D(r)\nabla\phi(r)$$

or

$$+\Sigma_a(r)\phi(r) - \nabla\bullet D(r)\nabla\phi(r) = S(r) \tag{17}$$

In 2D Cartesian geometry, this is

$$+\Sigma_a(r)\phi(r) - \frac{\partial}{\partial x} D(r)\frac{\partial}{\partial x}\phi(r) - \frac{\partial}{\partial y} D(r)\frac{\partial}{\partial y}\phi(r) = S(r) \tag{18}$$

To numerically attack this equation, we need to set up a 2 dimensional Cartesian grid. Figure 4 shows a typical grip point, P, at grid location (i,j). The x direction is denoted by the index j and the y dimension is denoted by the index i. This follows the (row, column) convention of matrices, rather than the (x,y) convention of plotting. Sorry for the confusion.

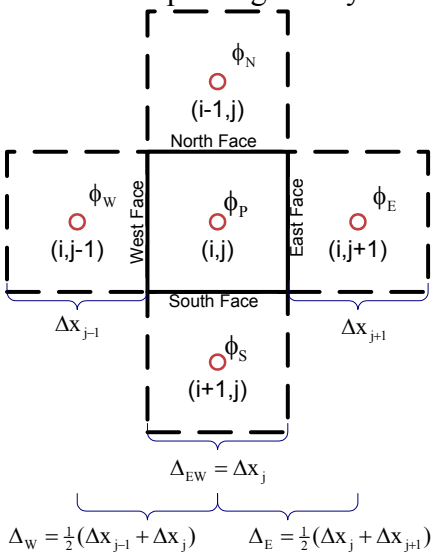


Figure 4 Typical point in a 2 dimensional grid.

Note that we have divided up our physical space into blocks. All parameters are discretised and the values at the block centers represent the values for the whole block. If we maintain enough grid points, the discretisation errors are small and acceptable. It is up to the user where the grid boundaries are placed but my preference is to place the grid boundaries to coincide with actual physical boundaries as much as possible. Thus we try to make exterior walls and interior interfaces fall on the grid faces. This means that parameters will not be evaluated right at the interfaces since the point values are located in the center of the grid blocks. It is equally valid to place the grid point centers at the interfaces and have the grid blocks span across an interface. Just be careful to label everything carefully and to consistently apply the differencing formulas. Herein though, we will stick with the rule of aligning the grid faces with the physical boundaries.

As an example, let's say that our reactor is to be modeled by a 9x6 grid as shown in figure 5.

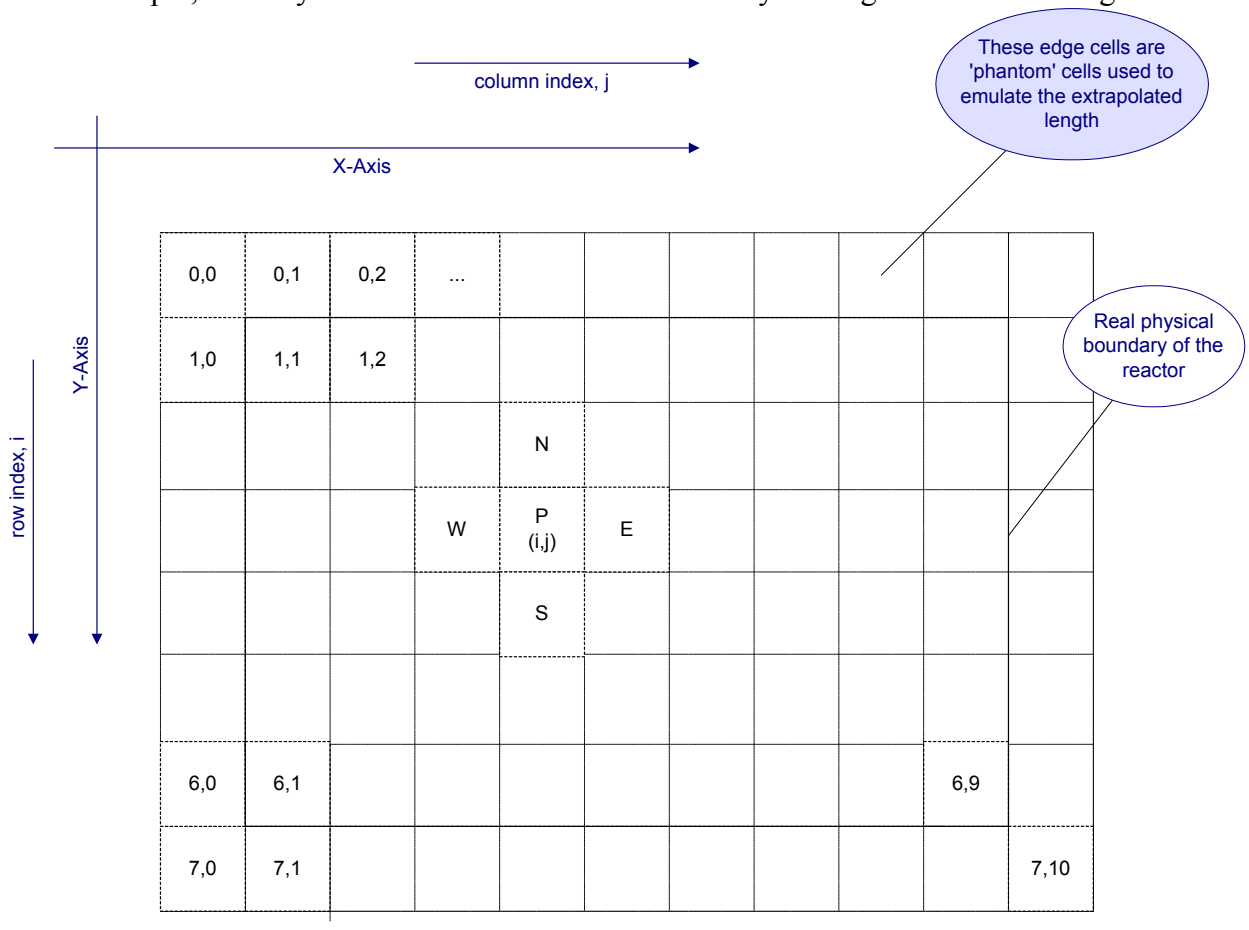


Figure 5 Grid layout for the full core.

3.2 The Diffusion Term

Now the diffusion term is just:

$$\frac{\partial}{\partial x}\left(D\frac{\partial\phi}{\partial x}\right)=\frac{\left(D\frac{\partial\phi}{\partial x}\right)_{x+\frac{\Delta x}{2}}-\left(D\frac{\partial\phi}{\partial x}\right)_{x-\frac{\Delta x}{2}}}{\Delta x_j}$$
$$=\frac{\text{differences of } D\frac{\partial\phi}{\partial x} \text{ at the east and west faces}}{\text{face separation}} \tag{19}$$

So what we need now are estimates of those terms.

Define: D_{WF} = estimate of D at the west face = $\frac{1}{2}\left(D_W+D_P\right)$ and $D_{EF}=\frac{1}{2}\left(D_E+D_P\right)$. Therefore,

$$\frac{\partial}{\partial x}\left(D\frac{\partial\phi}{\partial x}\right)=\frac{\frac{D_{EF}\left(\phi_E-\phi_P\right)}{\Delta_E}-\frac{D_{WF}\left(\phi_P-\phi_W\right)}{\Delta_W}}{\Delta_{EW}}$$
$$=\frac{D_{EF}\phi_E}{\Delta_E\Delta_{EW}}-\left(\frac{D_{EF}}{\Delta_E}+\frac{D_{WF}}{D_W}\right)\frac{\phi_P}{\Delta_{EW}}+\frac{D_{WF}}{\Delta_W}\frac{\phi_W}{\Delta_{EW}} \tag{20}$$

Similarily, we can form an expression for north-south $\left(\text{i.e.}\frac{\partial}{\partial y}D\frac{\partial\phi}{\partial y}\right)$.

Thus for $-\nabla\cdot D\nabla\phi+\sum_a\phi=S$ we have for the 1-D case:

$$-\frac{D_{WF}}{\Delta_W}\frac{\phi_W}{\Delta_{EW}}+\left[\left(\frac{D_{EF}}{\Delta_E}+\frac{D_{WF}}{\Delta_W}\right)\frac{1}{\Delta_{EW}}+\sum_a\right]\phi_P-\frac{D_{EF}}{\Delta_E}\frac{\phi_E}{\Delta_{EW}}=S_P \tag{21}$$

i.e.

$$a_{PW}\phi_W+a_{PP}\phi_P+a_{PE}\phi_E=S \tag{22}$$

The point P is arbitrary. Extension to 2 and 3 dimensions is trivial and we will see it in section 3.4 soon. We have arrived at an equation relating the flux at point P to be the flux at the nearest neighbours.

$$a_{1,0}\begin{pmatrix} a_{11} & a_{12} & & & & & \\ a_{21} & a_{2,2} & a_{2,3} & & & & \\ & & \ddots & & & & \\ & & a_{PW} & a_{PP} & a_{PE} & & \\ & & & & \ddots & & \\ & & & & a_{N,N-1} & a_{N,N} & \end{pmatrix} a_{N,N+1}\begin{pmatrix}\phi_1\\ \phi_2\\ \vdots\\ \phi_P\\ \vdots\\ \phi_N\end{pmatrix}=\begin{pmatrix}S_1\\ S_2\\ \vdots\\ S_P\\ \vdots\\ S_N\end{pmatrix} \tag{23}$$

Note that there are 2 terms hanging off the sides of the matrix. These terms are associated with the boundary conditions. Each case will likely be different so we have to pay careful attention to

the first and last equations to ensure that they are properly treated. For the case of a reactor modeled by zero flux at the boundaries, the $a_{1,0}$ and $a_{N,N}$ terms can be safely dropped since the associated ϕ_0 and ϕ_N are zero and so do not contribute to the first and last equations in the matrix.

In the special case where $D \neq \text{fn}(\mathbf{r})$ and grid spacing is equidistant then we get:

$\frac{-D}{\Delta^2}\phi_W + \left(\frac{2D}{\Delta^2} + \Sigma_a\right)\phi_P - \frac{D}{\Delta^2}\phi_E = S_P$ which may be more recognizable. It is still of tridiagonal form.

In setting up the code for the general case, we note that the coefficient a_{PP} (for ϕ_P) is a function of the east and west neighbours. So for, say, the west-most grid point:

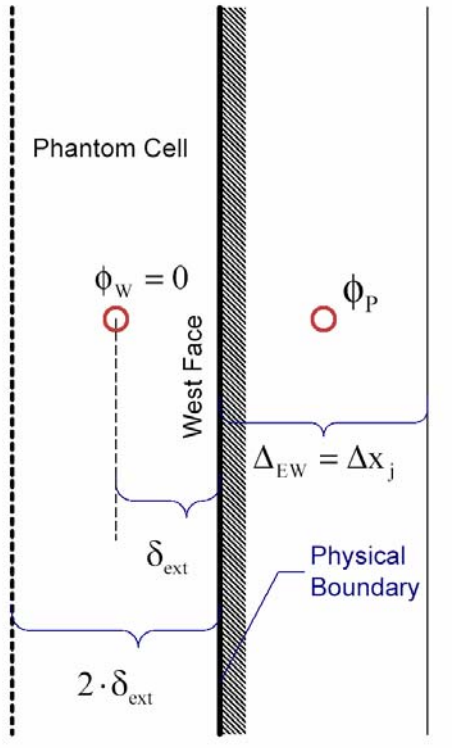
$$\phi_W = \phi_0 = 0 \tag{24}$$

and equation 21 becomes

$$+\left[\left(\frac{D_{EF}}{\Delta_E} + \frac{D_{WF}}{\Delta_W}\right)\frac{1}{\Delta_{EW}} + \Sigma_a\right]\phi_P - \frac{D_{EF}}{\Delta_E}\frac{\phi_E}{\Delta_{EW}} = S_P \tag{25}$$

But what is D_{WF}? There is no west cell to have a face!

My solution is to set up phantom cells that lie outside the grid as shown in the figure 5 previously and figure 6 below.



The phantom cell centre is located at the extrapolation length, δ_{ext}, i.e. at the point where the extrapolated flux=0.

Set the value of D for the phantom cell=D_P and $\Delta_W = \delta_{ext} + \frac{1}{2}\Delta_{EW}$. Now we can proceed to crunch numbers.

The numerical algorithm is now simply:

```
For all P:
```

$$\text{calculate}\ \phi_P = \frac{1}{a_{PP}}\left(-a_{PW}\phi_W - a_{PE}\phi_E + S_P\right)$$

```
Loop until ϕ's converge.
```

Figure 6 Treatment of the boundary using a phantom cell

We need to look more closely at S and we need to look more closely at the solution algorithm.

3.3 Modelling the Source Term when it is a δ Function

In the equation $-D\frac{\partial^2\phi}{\partial x^2}-\Sigma_a\,\phi=S$ where $S=S_0\delta(x)$ in the case of a planar source of neutrons and S has units of #/cm^3-sec. Now we know that $\int\delta(x)dx=1$, $\therefore$ units of $\delta(x)=\frac{1}{cm}$,

$\therefore S_0\equiv\frac{1}{cm^2\,sec}$.

<u>Watch out</u> when doing numerical solution. Integral through:

$$-\int_{\Delta x} D\frac{\partial^2\phi}{\partial x^2}dx+\int_{\Delta x}\Sigma_a\,\phi dx=\int_{\Delta x}S_0\underbrace{\delta(x)dx}_{\int=1}=S_0 \tag{26}$$

Consider $D\frac{\partial^2\phi}{\partial x^2}$, etc. const over Δx. Thus

$$-D\frac{\partial^2\phi_i}{\partial x^2}\Delta x+\Sigma_a\,\phi_i\Delta x=S_0 \text{ at the planar source cell}$$
$$= 0 \text{ otherwise.} \tag{27}$$

So, the equation to be numerically simulated is

$$-D\frac{\partial^2\phi_i}{\partial x^2}+\Sigma_a\,\phi_i=\frac{S_0}{\Delta x} \tag{28}$$

not

$$-D\frac{\partial^2\phi_i}{\partial x^2}+\Sigma_a\,\phi_i=S \tag{29}$$

This can be physically interpreted as the planar source term of infinitesimal thickness being spread over a finite thickness cell.

3.4 The Multidimensional Diffusion Equation and the Matrix Form

We have seen how to discretize the one dimensional diffusion equation. Extension to 2 and 3 dimensions is trivial. For a less cluttered illustration, consider the sace of constant D and equidistant grid spacing for the 2 dimensional case. We have, as the governing PDE:

$$-\mathrm{D}\frac{\partial^2\phi}{\partial \mathrm{x}^2}-\mathrm{D}\frac{\partial^2\phi}{\partial \mathrm{y}^2}+\Sigma_\mathrm{a}\,\phi=\mathrm{S}(\mathrm{x},\mathrm{y}) \tag{30}$$

which, when turned into a finite difference equation (FDE) is:

$$-\mathrm{D}\left(\frac{\phi_{\mathrm{i-i,j}}-2\phi_{\mathrm{i,j}}+\phi_{\mathrm{i+i,j}}}{(\Delta \mathrm{x})^2}\right)-\mathrm{D}\left(\frac{\phi_{\mathrm{i,j-1}}-2\phi_{\mathrm{i,j}}+\phi_{\mathrm{i,j+1}}}{(\Delta \mathrm{y})^2}\right)+\Sigma_{\mathrm{a\,i,j}}\,\phi_{\mathrm{i,j}}=\mathrm{S}_{\mathrm{i,j}} \tag{31}$$

In 3 dimensions, just add the z axis and an additional index, k. All the concepts are the same but everything gets messier.

It is instructive to cast these messy equations into a matrix form so we can see some overall behaviour without getting caught up in the details. We have seen the matrix form already in equation 22 and 23.

$$\mathrm{a_{PW}}\phi_\mathrm{W}+\mathrm{a_{PP}}\phi_\mathrm{P}+\mathrm{a_{PE}}\phi_\mathrm{E}=\mathrm{S} \tag{22}$$

$$\begin{pmatrix} \mathrm{a}_{11} & \mathrm{a}_{12} & & & & & \\ \mathrm{a}_{21} & \mathrm{a}_{2,2} & \mathrm{a}_{2,3} & & & & \\ & & \ddots & & & & \\ & & \mathrm{a_{PW}} & \mathrm{a_{PP}} & \mathrm{a_{PE}} & & \\ & & & & \ddots & & \\ & & & & \mathrm{a}_{\mathrm{N,N-1}} & \mathrm{a}_{\mathrm{N,N}} \end{pmatrix}\begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_\mathrm{P} \\ \vdots \\ \phi_\mathrm{N} \end{pmatrix}=\begin{pmatrix} \mathrm{S}_1 \\ \mathrm{S}_2 \\ \vdots \\ \mathrm{S}_\mathrm{P} \\ \vdots \\ \mathrm{S}_\mathrm{N} \end{pmatrix} \tag{23}$$

This is just $\mathbf{A\phi=S}$, a form of the $\mathbf{Ax=b}$ linear algebra equation that you are familiar with. All the usual rules and techniques for solving this matrix equation apply. In one dimension, it is easy to set up the matrix **A**, composed of the coefficients a_{ij}. But when additional dimensions are added, setup is fussy. In 2 dimensions, equation 22 becomes:

$$\mathrm{a_{PN}}\phi_\mathrm{N}+\mathrm{a_{PW}}\phi_\mathrm{W}+\mathrm{a_{PP}}\phi_\mathrm{P}+\mathrm{a_{PW}}\phi_\mathrm{W}+\mathrm{a_{PS}}\phi_\mathrm{S}=\mathrm{S} \tag{32}$$

and the $\boldsymbol{\phi}$ vector is constructed by stringing together M rows (M being the number of grid points in the y direction) of $\boldsymbol{\phi}$ subvectors of length N (N being the number of grid points in the x direction). So we have:

$$\boldsymbol{\phi}=\{\boldsymbol{\phi}_1,\boldsymbol{\phi}_2,\cdots\boldsymbol{\phi}_\mathrm{j},\cdots\boldsymbol{\phi}_\mathrm{M}\} \tag{33}$$

where each $\boldsymbol{\phi}_\mathrm{j}$ is a vector of size N for a total length of MxN. The resulting matrix setup looks like:

$$\begin{pmatrix} \ddots & & \ddots & & \\ \ddots & \ddots & & \ddots & \\ \ddots & \ddots & \ddots & & \ddots \\ a_{PN} & & a_{PW} \; a_{PP} \; a_{PE} & & a_{PS} \\ \ddots & & \ddots & \ddots & \ddots & \ddots \\ & \ddots & & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_P \\ \vdots \\ \phi_{NxM} \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_P \\ \vdots \\ S_{NxM} \end{pmatrix} \tag{34}$$

Only the p^{th} equation has been detailed. Notice how the east and west nearest neighbours form central tridiagonal stripes. The north and south neighbours form off-diagonal stripes and make matrix inversion and manipulation more costly. If a third dimension were modeled, there would be another 2 stripes in the matrix, making 7 in all.

Note: if $S = \nu \Sigma_f \phi$ then the source term is proportional to ϕ and we can more the source terms over to the diagonal of **A** and rewrite the matrix equations to get $\mathbf{A}'\boldsymbol{\phi} = 0$. To get a non-trivial solution we must has if $|\mathbf{A}'| = 0$, by Cramer's Rule. In a later chapter we encounter this as the search for reactor criticality. For now, just note that for this frequently occurring case we need to conduce a numerical search to find a consistent set of parameters (geometry and material properties) such that $|\mathbf{A}'| = 0$.

4 Iterative schemes for Ax=b

4.1 Jacobi – Richardson scheme

Separate out the diagonal part:

$$\mathbf{A} = \mathbf{D} - \mathbf{B}$$

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} = \begin{pmatrix} x & & & \\ & x & & \\ & & x & \\ & & & x \end{pmatrix} - \begin{pmatrix} & x & x & x \\ x & & x & x \\ x & x & & x \\ x & x & x & \end{pmatrix} \tag{35}$$

Thus:

$$\mathbf{A}\phi = \mathbf{S} \Rightarrow \mathbf{D}\phi = \mathbf{B}\phi + \mathbf{S} \tag{36}$$

Solving for ϕ**:**

$$\phi = \mathbf{D}^{-1}\mathbf{B}\phi + \mathbf{D}^{-1}\mathbf{S} \tag{37}$$

Inverting the diagonal matrix is trivial so this solution scheme is quick to program and fast to solve per iteration. Note that you have to iterate because ϕ appears on the right hand side of the equation. So whether this turns out to be an effective scheme depends on how quickly the solution converges, ie, on how many iterations are necessary before a steady state is reached.

Written out in full, the scheme is:

$$\phi_i^{(m+1)} = \frac{1}{a_{ii}}\left[S_i - \sum_{\substack{j=1 \\ i \neq j}}^{N} a_{ij}\phi_j^{(m)} \right] \tag{38}$$

for the general matrix. In the simple two dimensional reactor case that we had before, the A matrix was quite sparse so the sum is only over 4 terms, not the whole row, ie, since:

$$a_{PN}\phi_N + a_{PW}\phi_W + a_{PP}\phi_P + a_{PW}\phi_W + a_{PS}\phi_S = S \tag{32}$$

we can rewrite equation 38 as:

$$\phi_P^{(m+1)} = \frac{1}{a_{PP}}\left[S_P - a_{PN}\phi_N^{(m)} - a_{PS}\phi_S^{(m)} - a_{PE}\phi_E^{(m)} - a_{PW}\phi_W^{(m)} \right] \tag{39}$$

The one and three dimensional cases should be obvious.

The iterative scheme, where the superscript represents the iteration number, is:

$\phi^{(0)}$ = guess

$\phi^{(1)} = \mathbf{D}^{-1}\mathbf{B}\phi^{(0)} + \mathbf{D}^{-1}\mathbf{S}$

etc. until

$\phi^{m+1} = \phi^m =$ the converged ϕ

This works but converges slowly. We look for an improved scheme.

4.2 Gauss-Seidel or successive relaxation

In this scheme, we take advantage of the fact that as we sweep though the grid, we can use the updated values of the fluxes that we have just calculated. Thus the iteration scheme is:

$$\phi_i^{(m+1)} = \frac{1}{a_{ii}}\left[S_i - \sum_{j=1}^{i-1} a_{ij}\phi_j^{(m+1)} - \sum_{j=i+1}^{N} a_{ij}\phi_j^{(m)} \right] \tag{40}$$

or

$$\phi_P^{(m+1)} = \frac{1}{a_{PP}}\left[S_i - a_{PS}\phi_S^{(m)} - a_{PE}\phi_E^{(m)} - a_{PN}\phi_N^{(m+1)} - a_{PW}\phi_W^{(m+1)} \right] \tag{41}$$

where it is assumed that the sweep is from the north to the south, west to east, so that the north and west points have newly updated values available. Actually, programming this is quite easy: just always use the latest available values for the fluxes!

Compare this to the Jacobi-Richardson scheme just encountered. In the J-R scheme, only the old values were used.

In matrix form, the Gauss-Seidel method is equivalent to:

$$\mathbf{A} = \mathbf{L} - \mathbf{U}$$

$$\begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix} = \begin{pmatrix} x & & & \\ x & x & & \\ x & x & x & \\ x & x & x & x \end{pmatrix} - \begin{pmatrix} & x & x & x \\ & & x & x \\ & & & x \\ & & & \end{pmatrix} \tag{42}$$

where **L** contains the diagonal. Thus:

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{S} \Rightarrow \mathbf{L}\boldsymbol{\phi} = \mathbf{U}\boldsymbol{\phi} + \mathbf{S} \tag{43}$$

Solving for $\boldsymbol{\phi}$**:**

$$\boldsymbol{\phi} = \mathbf{L}^{-1}\mathbf{U}\boldsymbol{\phi} + \mathbf{L}^{-1}\mathbf{S} \tag{44}$$

The iterative scheme, where the superscript represents the iteration number, is:

$\boldsymbol{\phi}^{(0)}$ = guess

$\boldsymbol{\phi}^{(1)} = \mathbf{L}^{-1}\mathbf{U}\boldsymbol{\phi}^{(0)} + \mathbf{L}^{-1}\mathbf{S}$

etc. until

$\boldsymbol{\phi}^{m+1} = \boldsymbol{\phi}^{m}$ = the converged $\boldsymbol{\phi}$

L is not that hard to invert and the iteration converges more quickly that the J-R method. Overall, there is a net gain so that G-S is faster than J-R but convergence is still slow.

4.3 SOR (Successive Over-Relaxation)

If the convergence of the steady state reactor diffusion calculation is slow and if we have the

change from one iteration to the next, could we not extrapolate ahead and anticipate the upcoming changes? Yes we can. The method is called the Successive Over-Relaxation (SOR) scheme. Basically the scheme is to first calculate as per Gauss Seidel and then extrapolate, ie first calculate an intermediate solution, $\boldsymbol{\phi}^*$ as per GS:

$$\boldsymbol{\phi}^* = \mathbf{L}^{-1}\mathbf{U}\boldsymbol{\phi}^{(m)} + \mathbf{L}^{-1}\mathbf{S} \tag{45}$$

then weigh the intermediate solution with the old solution:

$$\boldsymbol{\phi}^{(m+1)} = \omega\boldsymbol{\phi}^* + (1-\omega)\boldsymbol{\phi}^{(m)}, \omega \in (1-2) \tag{46}$$

Since ω is between 1 and 2, this is an extrapolation procedure. If it were between 0 and 1, it would be an interpolation procedure but we are trying to speed up the process, not stabilize it. The parameter ω is varied to give optimum convergence rate. It is suggested that you start out with ω close to 1 (ie heavy reliance on the GS solution) at the beginning and to increase ω as you become more confident that the extrapolation won't lead to overstepping the situation. Convergence rates ~100 times better than Jacobi are reported. Figure 7 illustrates this idea.

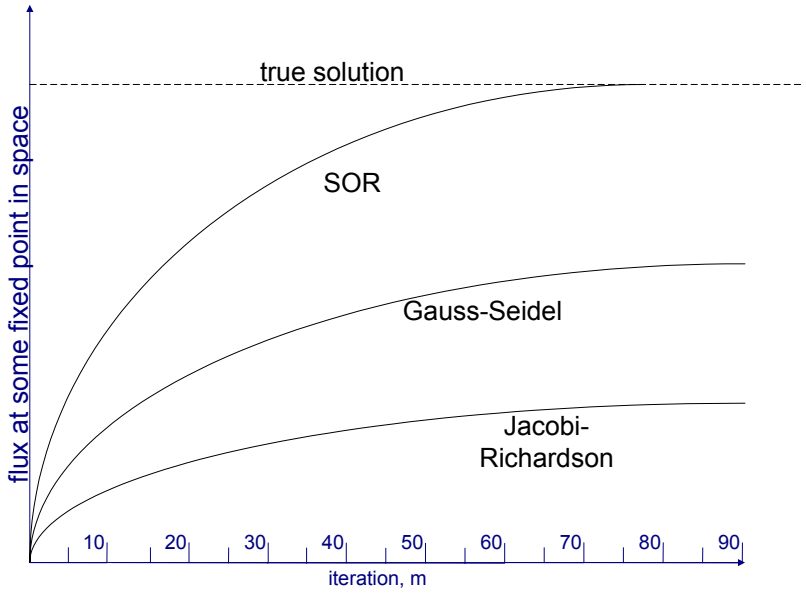


Figure 7 Convergence rate for iterative schemes.

These iterations are referred to as **inner** iterations. **Outer** or source iterations refer to varying parameters to achieve criticality and occur when the fixed source term, S, is replaced by a fission term that is proportional to the flux. More on that in a later chapter.

5 Space-time Modelling

We have looked at how to model the time behaviour independent of space and how to model space independent of time. Now let's model time and space dependencies together. For illustrative purposes, assume one dimension, D is constant and the grid is equally spaced. The one speed transient neutron diffusion equation in one dimension is:

$$\frac{1}{v}\frac{\partial\phi}{\partial t} = D\frac{\partial^2\phi}{\partial x^2} - \Sigma_a\,\phi + S(x) \tag{47}$$

which in finite difference form is:

$$\frac{1}{v}\frac{\left(\phi_p^{\,t+\Delta t} - \phi_p^{\,t}\right)}{\Delta t} = \frac{D}{\Delta^2}\phi_w - \frac{2D}{\Delta^2}\phi_p + \frac{D}{\Delta^2}\phi_E - \Sigma_a\,\phi_p + S_p \tag{48}$$

Rearranging:

$$\phi_p^{\,t+\Delta t} = \phi_p^{\,t} + v\Delta t\left[\frac{D}{\Delta^2}\phi_w - \frac{2D}{\Delta^2}\phi_p + \frac{D}{\Delta^2}\phi_E - \Sigma_a\,\phi_p + S_p\right] \tag{49}$$

The fully explicit solution would evaluate the ϕ's on right hand side (RHS) of equation 49 at time t. We can just solve directly for ϕ at the point in question:

. $$\phi_p^{\,t+\Delta t} = \phi_p^{\,t} + v\Delta t\left[\frac{D}{\Delta^2}\phi_w^{\,t} - \frac{2D}{\Delta^2}\phi_p^{\,t} + \frac{D}{\Delta^2}\phi_E^{\,t} - \Sigma_a\,\phi_p^{\,t} + S_p\right] \tag{50}$$

But this is a waste since if we are scanning from west to east, we have already called $\phi_w^{\,t+\Delta t}$. We might as well use it. Further we can make it implicit in ϕ_p by evaluating ϕ_p at $t+\Delta t$ on the RHS to give:

$$\left[1 + v\Delta t\left(\frac{2D}{\Delta^2} + \Sigma_a\right)\right]\phi_p^{\,t+\Delta t} = \phi_p^{\,t} + v\Delta t\left[\frac{D}{\Delta^2}\phi_w^{\,t+\Delta t} + \frac{D}{\Delta^2}\phi_E^{\,t} + S_p\right] \tag{51}$$

i.e.:

$$\phi_p^{\,t+\Delta t} = \frac{\phi_p^{\,t} + v\Delta t\left[\frac{D}{\Delta^2}\phi_w^{\,t+\Delta t} + \frac{D}{\Delta^2}\phi_E^{\,t} + S_p\right]}{\left[1 + v\Delta t\left(\frac{2D}{\Delta^2} + \Sigma_a\right)\right]} \tag{52}$$

We can use this to track $\phi(t)$ dynamically to steady state. This follows a physically real path and always gets a solution. We can play the usual explicit/implicit games.

……

There is an alternate approach. The transient equation can be rearranged to give:

$$\frac{-D}{\Delta^2}\phi_w^{\,t+\Delta t} + \left[\frac{1}{v\Delta t} + \frac{2D}{\Delta^2} + \Sigma_a\right]\phi_p^{\,t+\Delta t} - \frac{D}{\Delta^2}\phi_E^{\,t} = S_p + \frac{\phi_p^{\,t}}{v\Delta t} \tag{53}$$

This is identical to the steady state equation, except for the terms in $v\Delta t$. Surprise, surprise!

This is just $\underline{\underline{A}}\underline{\phi} = \underline{S}$ as before and can be solved by the techniques previously discussed. This needs to be done for each time step. The one advantage that I can see with this form is that you can reuse code that you have written for the steady state.

I don't like the above form because it blows up as $\Delta t \to 0$ and it is not as intuitive as the forms used in equation 50 and 52 which maintain their direct relationship to the PDE. I prefer to see the time solution evolve step by step in a physically obvious manner. But, ultimately, the choice is yours.

About this document:

back to page 1

Author and affiliation: Wm. J. Garland, Professor, Department of Engineering Physics, McMaster University, Hamilton, Ontario, Canada

Revision history:
Revision 1.0, 2001.07.10, initial creation from hand written notes.
Revision 1.1, 2004.08.17, minor typo correction.

Source document archive location:
D:\TEACH\EP4D3\text\4-numeric\numeric-r1.doc

Contact person: Wm. J. Garland

Notes: