FERMILAB-CONF-05-481-CD

Track Clustering and Vertexing algorithm for L1 Trigger

Gustavo I. Cancelo

Abstract-**One of the keystones of the canceled BTeV experiment (proposed at Fermilab's Tevatron) was its sophisticated three-level trigger. The trigger was designed to reject 99.9% of light-quark background events and retain a large number of B decays. The BTeV Pixel Detector provided a 3-dimensional, high resolution tracking system to detect B signatures. The Level 1 pixel detector trigger was proposed as a two stage process, a track-segment finder and a vertex finder which analyzed every accelerator crossing. In simulations the track-segment finder stage outputs an average of 200 track-segments per accelerator crossing (2.5MHz). The vertexing stage finds vertices and associates track-segments with the vertices found. This paper proposes a novel adaptive pattern recognition model to find the number and the estimated location of vertices, and to cluster track-segments around those vertices. The track clustering and vertex finding is done in parallel. The pattern recognition model also generates the estimate of other important parameters such as the covariance matrix of the cluster vertices and the minimum distances from the tracks to the vertices needed to compute detached tracks.**

I. Introduction

THE problem of event vertexing requires identifying vertices and estimating their position. Generally, both tasks are done with the help of data supplied by subdetectors with tracking capabilities such as pixel detectors or silicon strips detectors (SSD). Before the vertexing stage, track finder stages find tracks or track segments for use in the vertexing process. Since there may be more than one primary vertex and or secondary vertices, identifying the number of vertices is a clustering task that associates tracks to vertices. Once the tracks and vertices have been sorted out, each group of tracks can be used to fit corresponding vertex parameters. We observe that this is a "chicken and egg" kind of problem because we need good vertex estimations to be able to separate the tracks into clusters and we cannot have good vertex estimations if we don't know what tracks belong in which cluster. The problem is typical in pattern recognition and is usually approached by iterative adaptive methods. In these methods the parameters of the cluster representative (e.g. vertex coordinates) are allowed to move and the cluster members (e.g. tracks) are allowed to change clusters during each iteration. Most adaptive clustering algorithms are based on models, where the model is a probability distribution function. However, most of the time the order of the model is unknown (e.g. the number of vertices). This is a serious complication because most algorithms are not good at detecting the order of the model. Another important problem in clustering is finding a good metrics to sort the data out.

Data is clustered based on their distance to the cluster representative vectors. Distance is defined by: $\boldsymbol{D} = \|\boldsymbol{x}-\boldsymbol{q}\|_{\boldsymbol{M}}$, where M is a certain metric. The two most used metrics are the Euclidean and the Mahalanobis metrics. In the later case, $\boldsymbol{D} = \|\boldsymbol{x}-\boldsymbol{q}\|_{\boldsymbol{M}} = (\boldsymbol{x}-\boldsymbol{q})^{\boldsymbol{T}}\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{q})$, where x and q are d-dimensional vectors and C is the covariance matrix of the sample data. The Euclidean distance is a special case of the Mahalanobis metrics where $C=\sigma^2 I$, being I the identity matrix. The Mahalanobis distance is preferred when the variance along each dimension of x is different. However, neither one of those metrics is able to capture clustering in the data.

II. Model and Metric Selection

Clustering processes use parametric or non-parametric models. For the vertexing process we have chosen the Gaussian mixture model due to the clustered structure of the data. The vertexing process clusterizes tracks based on their distance of closest approach to the vertices, which are the cluster representatives. The track distance of closest approach to the vertices depends on the detector resolution and errors. It is logical to think that the error in measuring distances from tracks to an associated vertex is Gaussian distributed, and the probability distribution function (pdf) of all the distances between tracks and vertices is a sum of Gaussians. Hence, the pdf of all the distances to all the cluster centers is,

$$\boldsymbol{p}(\boldsymbol{x}) = \sum_{\boldsymbol{j}=1}^{\boldsymbol{k}} \boldsymbol{\alpha_j} \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\mu_j}, \boldsymbol{C_j}) \quad (1)$$

where G(*) is a single d-dimensional Gaussian of mean μ_j and covariance matrix C_j. α_j is the probability that data x belongs to cluster j, μ_j and C_j are the mean and covariance matrix of the Gaussian associated to cluster j. For the vertexing problem, α_j is the probability that track x comes from vertex j, μ_j are the coordinates of vertex j, and C_j is the covariance matrix of all the tracks associated to that vertex. In principle, α_j is model free and is one of the parameters that must be estimated. The only constrains on α_j are:

$\alpha_j > 0$, and $\sum_{j=1}^{k} \alpha_j = 1$ (i.e. every track must belong to a cluster)

A natural metric to use in a clustering problem modeled by a Gaussian mixture is the Kullback-Leibler divergence. The KL divergence measures the divergence between two pdf distributions

$$\boldsymbol{KL}(\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{q}) \| \boldsymbol{p}(\boldsymbol{x})) = \int \boldsymbol{p}(\boldsymbol{x}|\boldsymbol{q}) \log\left(\frac{\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{q})}{\boldsymbol{p}(\boldsymbol{x})}\right) \boldsymbol{dx}$$

Note that KL(*) meets the conditions to be a metric:
KL(0)=0 (i.e. KL=0 for p(x|q)=p(x))

KL>0 for p(x|q)≠p(x) because p(x|q) is always greater or equal than p(x), hence log(p(x|q)/p(x))>0.

Parameter optimization is done minimizing the KL divergence of the parametric distribution for instance $\frac{dKL(*)}{d\mu_j}=0$ *and* $\frac{dKL(*)}{dC_j}=0$. However, since KL divergence usually has unknowns, the later calculations must be turned into estimations.

The KL divergence metric leads us to the Maximum Likelihood algorithm. The problem of estimating vertex coordinates and their covariance matrices becomes a Maximum Likelihood estimation. In particular we have applied the Expectation-Maximization algorithm (E-M) which is a well known variant of Maximum Likelihood. The E-M algorithm is an efficient iterative way of computing the Maximum Likelihood when the data model has missing information. The missing information concept helps the estimations of certain parameters in the model that otherwise would complicate the algorithm. The missing information parameters can be chosen as convenient.

We illustrate the missing information concept in the vertexing problem as follow. The data model for the vertexing problem is the mixture Gaussian model defined in equation (1). For the moment we will assume that we know the number of clusters k (i.e. number of vertices). We define the missing information as j, the cluster index, which has a probability α_j. As said, the α_j's are not model constrained. The missing information will help us overcome the problem of having this unstructured unknown in the model. The α_j's can be added to the list of parameters to be estimated by the ML algorithm. However, in the derivation of the algorithm we need to compute the conditional probability P(j|x) which represents the probability of each cluster given a particular sample x. For the vertexing problem P(j|x) represents the probability mass function that assigns tracks (represented by their distances of maximum approach) to each vertex.

Section 3 introduces the EM algorithm, section 4 shows that the KL divergence metrics that measures the distance between the pdfs of the data (including the missing data) and the current estimation of the data leads to the EM algorithm, section 5 applies the theory to the vertexing problem section 6 shows some simulation results.

III. THE EXPECTATION MAXIMIZATION ALGORITHM

Let x represent the data sample and θ the unknown vector of parameters. Maximum Likelihood estimates θ maximizing the likelihood function P(x|θ). Since the distributions are Gaussian, ML uses the log likelihood L(θ) = ln(P(x|θ)). The EM algorithm is an iterative procedure that finds successive estimations of θ (i.e. $\theta^{(1)}, \theta^{(2)},.., \theta^{(N)}$) while guaranties that L(θ) increases in every iteration (i.e. $L(\theta^{(n+1)}) > L(\theta^{(n)})$, n=1,..,N.

This is equivalent to maximizing the difference

$$L(\theta^{(n+1)}) - L(\theta^{(n)}) = \ln(P(x|\theta^{(n+1)})) - \ln(P(x|\theta^{(n)})) \qquad (2)$$

We incorporate the missing information in the data model using the total probability theorem,

$$P(x|\theta)=\sum_{j=1}^{k} P(x|j,\theta).P(j|\theta)$$

Then equation (2) can be expressed by

$$L(\theta^{(n+1)})-L(\theta^{(n)})=\ln\left(\sum_{j=1}^{k} P(x|j,\theta^{(n+1)}).P(j|\theta^{(n+1)})\right)-\ln(P(x|\theta^{(n)})) \quad (3).$$

Since the –ln(.) is a convex function, we use Jensen's inequality

$$\ln\left(\sum_{j=1}^{k}\lambda_j . z_j\right)=\sum_{j=1}^{k}\lambda_j .\ln(z_j) \qquad (4)$$

Now we introduce the cluster's posterior probability P(j|x, θ_n) in equation (3). P(j|x,θ_n) represents cluster's j probability given a certain data and set of parameters (i.e. during the n-th iteration of the algorithm). Then equation (3) becomes

$$L(\theta_{n+1})-L(\theta_n)=\ln\left(\sum_{j=1}^{k} P(x|j,\theta_{n+1}).P(j|\theta_{n+1}).\frac{P(j|x,\theta_n)}{P(j|x,\theta_n)}\right)-\ln(P(x|\theta_n))$$

, which can be rearranged as

$$L(\theta_{n+1})-L(\theta_n)=\ln\left(\sum_{j=1}^{k} P(j|x,\theta_n).\frac{P(x|j,\theta_{n+1}).P(j|\theta_{n+1})}{P(j|x,\theta_n)}\right)-\ln(P(x|\theta_n)) \quad (5)$$

Using Jensen's inequality (4) and defining

$\lambda_j = P(j|x,\theta_n)$ and $z_j=\frac{P(x|j,\theta_{n+1}).P(j|\theta_{n+1})}{P(j|x,\theta_n)}$, equation (5) becomes

$$L(\theta_{n+1})-L(\theta_n)\geq\sum_{j=1}^{k} P(j|x,\theta_n).\ln\left(\frac{P(x|j,\theta_{n+1}).P(j|\theta_{n+1})}{P(j|x,\theta_n)}\right)-\ln(P(x|\theta_n))$$

$$L(\theta_{n+1})-L(\theta_n)\geq\sum_{j=1}^{k} P(j|x,\theta_n).\ln\left(\frac{P(x|j,\theta_{n+1}).P(j|\theta_{n+1})}{P(j|x,\theta_n)P(x|\theta_n)}\right) \qquad (6)$$

Let $\Delta(\theta_{n+1}|\theta_n)=\sum_{j=1}^{k} P(j|x,\theta_n).\ln\left(\frac{P(x|j,\theta_{n+1}).P(j|\theta_{n+1})}{P(j|x,\theta_n)P(x|\theta_n)}\right)$

Equation (6) tells us that $\Delta(\theta_{n+1}|\theta_n)$ is a lower bound of the log-likelihood function and that maximizing $\Delta(\theta_{n+1}|\theta_n)$ increases the log-likelihood. Figure 1 shows one iteration of the estimation process.

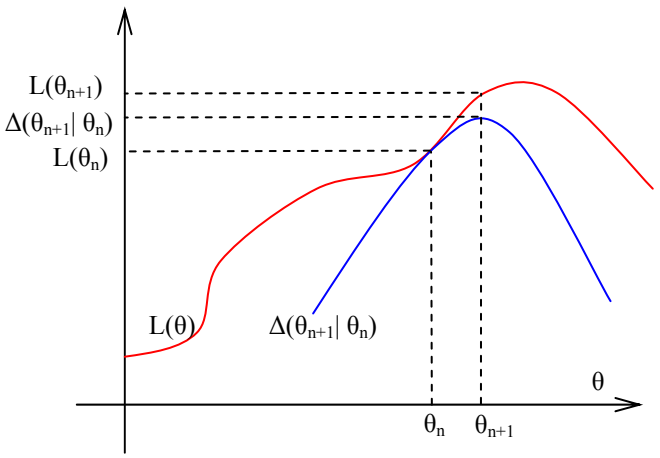

Figure 1: one step in the Maximum Likelihood estimation process

We can estimate the vector parameter θ calculating

$$\theta_{n+1} = \underset{\theta}{\mathbf{argmax}}\{L(\theta_n) + \Delta(\theta \mid \theta_n)\} \quad (7)$$

$$\theta_{n+1} = \underset{\theta}{\mathbf{argmax}}\left\{L(\theta_n) + \sum_{j=1}^{k} P(j \mid x,\theta_n).\ln\left(\frac{P(x \mid j,\theta).P(j \mid \theta)}{P(j \mid x,\theta_n)P(x \mid \theta_n)}\right)\right\}$$

Equation (7) can be simplified eliminating all the terms that are constant with respect to θ.

$$\theta_{n+1} = \arg\max_{\theta}\left\{\sum_{j=1}^{k} P(j \mid x,\theta_n).\ln(P(x,j \mid \theta))\right\} \quad (8)$$

Note that equation (8) maximizes the conditional expectation $E_{j|x,\theta}$ of the logarithm of P(x, j|θ) with respect to θ. That can be expressed as

$$\theta_{n+1} = \arg\max_{\theta}\{E_{j|x,\theta_n}(\ln(P(x,j \mid \theta)))\} \quad (9)$$

The two steps of the E-M algorithm are now evident in equation (9). The first step, E-step, calculates the conditional expectation $E_{j|x,\theta}$ using the current estimation of the vector parameter θ_n. The second step, M-step, maximizes the expectation with respect to θ.

The convergence properties of the EM algorithm are discussed in detail by McLachlan et al. [1]. Every iteration the new θ_{n+1} increases the log likelihood function until a local maximum is reached.

IV. APPLYING E-M TO THE VERTEXING PROBLEM

In the Gaussian mixture model (equation (1)) $p(x) = \sum_{j=1}^{k} \alpha_j G(x, \mu_j, C_j)$ the distances of maximum approach from tracks to each vertex is represented by a single Gaussian $G(x,\mu_j,C_j)$, where μ_j and C_j must be estimated. Please note that:

$P(j|x_i)$ is the probability that track I belongs to a vertex j.

p(j,x) is the probability density function of the "complete data". Also, p(j,x) = p(x|j) p(j)

$P(j) = \alpha_j$ is the cluster probability.

$p(x|j) = G(x,\mu_j,C_j)$ is the data error distribution in each cluster.

The KL divergence function measures the distance between the distribution of the "complete data" p(x,j) and the distribution of the "complete data" given the current estimate of the parameter vector $p(x,j|\ \theta_n)$. Unfortunately p(x,j) is unknown otherwise the problem would be already solved. The KL divergence is $KL(p(x,j|\ \theta_n) \parallel p(x,j))$.

$$KL(P(x,j \mid \theta_n) \| P(x,j)) = \sum_{j=1}^{k} \int \left[P(x,j \mid \theta_n) \ln\left(\frac{P(x,j \mid \theta_n)}{P(x,j)}\right)\right] dx \quad (10)$$

The parameter set θ concentrates the unknowns α_j,μ_j,C_j for i=1,..k.

We develop an estimation algorithm for the parameter set of equation (10) based on the E-M algorithm of equation (9)

$$\theta_{n+1} = \arg\max_{\theta}\{E_{j|x,\theta_n}(\ln(P(x,j \mid \theta)))\}$$

$$\theta_{n+1} = \arg\max_{\theta}\left\{\sum_{j=1}^{k} P(j \mid x,\theta_n).\ln(P(x,j \mid \theta))\right\}$$

Clearly, the E-step must find the expectation $E_{j|x,\theta}$ where

$$P(j \mid x,\theta_n) = \frac{\alpha_j^{(n)} G\left(x,\mu_j^{(n)},C_j^{(n)}\right)}{\sum_{j=1}^{k} \alpha_j^{(n)} G\left(x,\mu_j^{(n)},C_j^{(n)}\right)} \quad (11)$$

$\alpha_j^{(n+1)}$ is readily available from (11) using

$$\alpha_j^{(n+1)} = \frac{1}{N}\sum_{i=1}^{N} P(j \mid x_i,\theta_n) \quad (12)$$

, where i=1,..,N indexes over all the data set and x_i is a d-dimensional vector.

The maximization step finds a new parameter set estimate

$$\theta_{n+1} = \underset{\theta}{\mathbf{argmax}}\{E_{j|x,\theta_n} \ln(P(x,j \mid \theta))\} \quad (13)$$

$$\theta_{n+1} = \underset{\theta}{\mathbf{argmax}}\left\{\sum_{j=1}^{k} P(j \mid x,\theta_n).\ln P(j \mid x,\theta)\right\}$$

the maximization of (13) leads to

$$\mu_j^{(n+1)} = \frac{\sum_{i=1}^{N} P(j \mid x_i,\theta_n).x_i}{\sum_{i=1}^{N} P(j \mid x_i,\theta_n)}$$

or $$\mu_j^{(n+1)} = \frac{1}{\alpha_j^{(n)} N}\sum_{i=1}^{N} P(j \mid x_i,\theta_n).x_i \quad (14)$$

$$C_j^{(n+1)} = \frac{\sum_{i=1}^{N} P(j \mid x_i,\theta_n).(x_i - \mu_j^{(n)}).(x_i - \mu_j^{(n)})^T}{\sum_{i=1}^{N} P(j \mid x_i,\theta_n)}$$

or $$C_j^{(n+1)} = \frac{1}{\alpha_j^{(n)} N}\sum_{i=1}^{N} P(j \mid x_i,\theta_n).(x_i - \mu_j^{(n)}).(x_i - \mu_j^{(n)})^T \quad (15)$$

The Trigger algorithm iteratively calculates equations (11) to (15) until convergence. The convergence criteria is chosen,

as convenience, based on the rate of change of the parameters being estimated. We have chosen 1% for simulation.

V. NUMBER OF CLUSTERS

The derivation of the E-M algorithm of section 3 assumes that the number of clusters k is known. This is not the case for vertexing problem. If we define k as a new parameter in the model, k represents the order of the model, i.e. the number of Gaussians in the Gaussian mixture. We can include k in the definition of the KL divergence and make the divergence a function of F(θ,k). However, in order to use the E-M approach we must keep k constant, otherwise we do not know how many Gaussians are in the model and how many parameters we need to estimate. A way of finding k is to calculate $F(\theta^{opt},k)$ as in section 4, r times, running k from 1 to r, and using θ_k^{opt} the value that minimizes $F(\theta^{opt},k)$ for that value of k. Then k is the argmin_k $F(\theta^{opt},k)$. So $F(\theta^{opt},k)$ has a minimum when k is equal to the correct number of clusters in the distribution.

The bigger problem in finding k is that if we have no idea about the value of k the algorithm may become computer intensive. Lei Xu [2] demonstrated that the $F(\theta^{opt},k)$ equals to

$$F(k) = \sum_{j=1}^{k} \alpha_j^{(opt)} \cdot \ln\left(\frac{\sqrt{|C_j^{(opt)}|}}{\alpha_j^{(opt)}}\right) \quad for \quad k = 1,..,r \qquad (16)$$

A good heuristic in finding k is to use previous knowledge. If after analyzing a sufficient number of events we have a reasonable estimate of the μ and C in the distribution of number of vertices per event we can start with a value in excess and rapidly discard the number of vertices that have no tracks associated with them. We show that this approach works very well is the simulation.

VI. DISTANCE OF MAXIMUM APPROACH

In order to associate tracks to vertices we have used their distance of maximum approach. The L1 trigger simulations are based on simulations of the BTeV pixel detector. The first stage of the L1 trigger, described in [3], finds track segments corresponding to the inner most and the outer most section of each track. Each segment is represented by a triplet of points using (x,y,z) coordinates. For trigger level 1 we have used only the inner segment of the track. However, a better fitting can be used for other trigger levels or the off-line analysis. It is worth to mention that the method to find the distance of closest approach is independent of the EM algorithm although very important because directly influence the error of the measurements and the EM algorithm convergence rate.

Let each track segment be represented by 3 points with coordinates $P_1=(x_1,y_1,z_1)$, $P_2=(x_2,y_2,z_2)$, $P_3=(x_3,y_3,z_3)$, and let $P_0=(x_0,y_0,z_0)$ be the point of closest approach to the vertex $V=(x_v,y_v,z_v)$. Then the distance of closest approach between the track and the vertex is defined by

$x_0 = x_1 + m_x (z_0 - z_1)$
$y_0 = y_1 + m_y (z_0 - z_1)$
$z_0 = (z_v + m_x (x_v - x_1) + m_y (y_v - y_1) + z_1 (m_x + m_y)) / (1 + m_x^2 + m_y^2)$

where m_x and m_y are the segment slopes in the xz and yz planes respectively. The point of closest approach must be recalculated every time the vertex position changes, but this is done very fast because it only requires of sums and multiplications by constant factors (i.e track slopes are constant for all iterations).

VII. SIMULATIONS

We have run a simulation with min bias and b flavor events. The simulation starts with a guess for number of vertices. Once the estimation converges the function F(k) of equation (16) is calculated. If there are vertices with no tracks those vertices are removed. If all vertices have tracks we increase the order of the model and run again. After convergence a new value of F(k) is obtained. We chose the model that minimizes F(k).

Figure 2 a and b show the X-Z view of a particular event with 3 primary vertices. The algorithm is run with a model of order 4. The black asterisks represent the estimated position of the four the vertices. Since the event only has 3 primary vertices, one estimated vertex does not collect any tracks. Figure 2 b is a close up of the same event showing in detail the tracks associated to two close primary vertices.

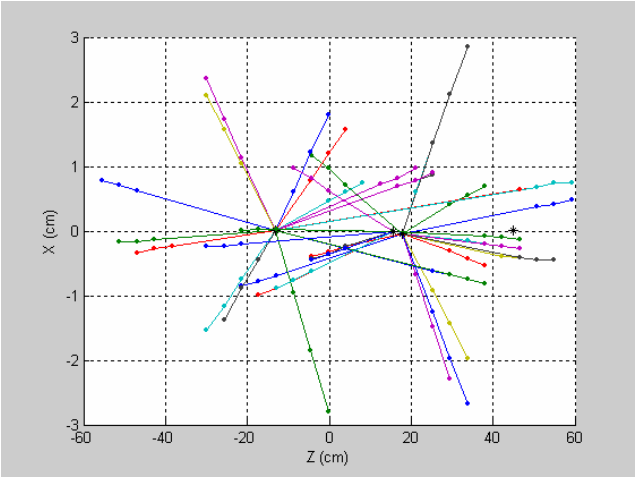

Figure 2a: Single event simulation for the primary vertex estimation algorithm.

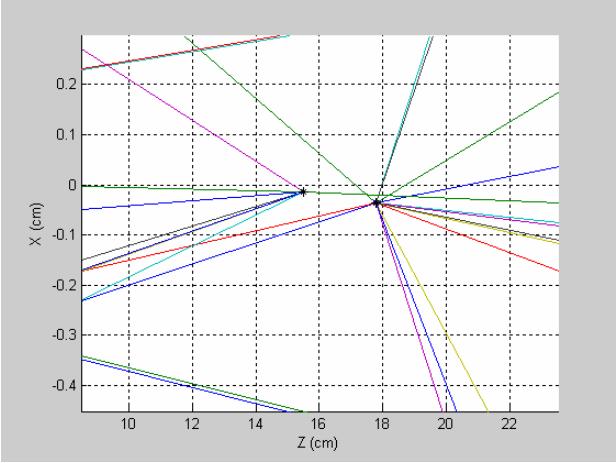

Figure 2b: Single event simulation for the primary vertex estimation algorithm.

The algorithm shows to converge and find over 99.5% of the primary vertices. The errors are listed in Table I.

TABLE I
SIMULATION ERRORS

	mean (μ)	sigma (μ)
x coordinate	275	65
y coordinate	1620	481
z coordinate	1107	280

VIII. DISCUSSION

The proposed algorithm applies an adaptive pattern recognition model to find the number and the estimated location of vertices, and clusters track-segments around those vertices. The track clustering and vertex finding is done in parallel. The pattern recognition model also generates the estimate of other important parameters such as the covariance matrix of the cluster vertices. The cancellation of the BTeV experiment left the simulation work unconcluded. It is expected that the number of vertices that the algorithm can find depends strongly on the data error model which depends on the way the distance of closest approach is determined. A follow up on this work should be aimed to reduce the measurement error in the data model.

REFERENCES

[1] Geoffrey McLachlan and Thriyambakam Krishnan. The EM Algorithm and its Extensions. John Willey & Sons, New York, 1996.

[2] Xu Lei, "Bayesian Ying-Yang machine clustering and number of clusters", Pattern Recognition Letters", vol. 18, pp 1167-1178, 1997.

[3] Cancelo, G.; Gottschalk, E.; Pavlicek, V.; Wang, M.; Wu, J**, "**Failure related dataflow dynamics in a highly parallel processor for L1 triggering" IEEE Transactions on Nuclear Science, Vol. 51, Issue 3, Part 3, June 2004 pp.1158-1162.

.