CLAS-NOTE 2003-006
CVS: docs/clas_notes/dc_inefficiency_procedure
Original: Apr 04, 2003
Version: 1.0

Procedure for Drift Chamber inefficiencies

Maurizio Ungaro, Ji Li

1 Introduction

We present a procedure to take into account DC dead, inefficient, normal and hot wires for acceptance calculation purposes. There are currently three methods to identify such wires:

- **Pdu internal algorithm:** for a given wire w_i, it first calculate the sum of the change in occupancy of a wire from its neighbors

  $$C_i = |o_i - o_{i-1}| + |o_i - o_{i+1}|$$

  where o_i is the occupancy of w_i. The weight $W_i = 1/(C_i + 1)$ is used to calculate the nominal occupancy O in each superlayer

  $$O = \frac{\sum o_i W_i}{\sum W_i}$$

  If a wire has less than $1/10$ of the nominal occupancy, i.e. if $o_i < \frac{1}{10} O_i$ then it is flagged "dead" (status = 1).

- **Todor - Niculescu algorithm:** for each wire, a sample of the equivalent wires in all six sectors is considered for a total of six wires. The hot wires, being identified by a certain number pf rms deviation from the mean, are removed from the sample, then the new mean M is calculated. Wires with occupancy below 80% of M are considered dead. See CLAS NOTE 02-017 for details.

- **Ungaro - Li algorithm:** it assigns a fractional number to a wire representing its efficiency and correlations. This note describes this method in details (see sections 4-6).

After this identification, the user can choose to perform one or more of the following actions:

- remove the flagged dead wires from the montecarlo simulation.
- remove the flagged dead wires from the real data (before cooking).
- reproduce real wires inefficiencies and correlations in the montecarlo simulation.

2 Background and phenomenology

Each CLAS sector has the same drift chamber configuration: 3 separate regions containing a total of 34 layers of sense wires. Region 1 has four layers, region 2 and 3 have six layers each. To simplify the labeling scheme, we'll count 36 layers in all regions, hence adding the two phantom layers 5 and 6 to region 1.

Layer	1	2	3	4	5	6	7	8	9	10	11	12
Region 1	130	130	130	130	0	0	142	142	142	126	121	120
Region 2	184	185	186	187	188	189	189	189	190	191	192	192
Region 3	192	192	192	192	192	192	192	192	192	192	192	192

Table 1: *Number of wires in each layer. Region 1 has only 4 layers, so layers 5 and 6 are phantom.*

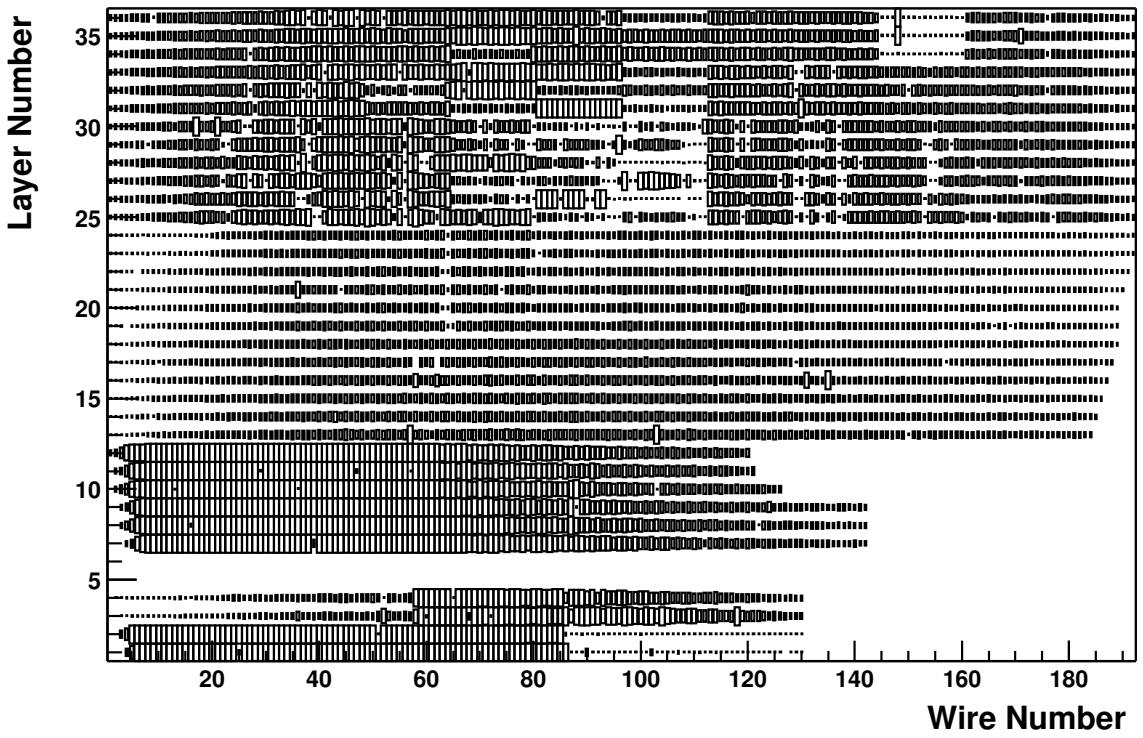

Figure 1: *Drift chamber occupancy distribution, data is from e1-6 run period*

There are nearly no counts in layers 34-35-36 and wire number ≈ 150. This is an example of a "hole". During tracking, a hole could affect track reconstruction because a minimum number of wires are required to define a track. Wires that count significantly more than neighboring ones are "hot". Track reconstruction is basically undisturbed by them[1]. A third pathology is represented by wires that count less than neighboring ones

[1]This is an empirical statement.

but not *substantially less*. For example, a wire can count an average of 70% relative to its neighbours. Such wires can be considered "warm" wires. Warm wires could be correlated due to common electronics. For example they could be attached to the same (defective) ADB board [2], so that all wires in that board have *the same efficiency at the same time*. Correlated wires affect tracking in that a group of wires might miss at the same moment, preventing the creation of a track segment. Treating warm wires systematically as holes results in the loss of particle tracks.

Another pathology is represented by wires that are alive during part of the run and dead during another part of the run. Running pdu on files that cover the whole time period could result in zones that seem warm: as an example, if a wire is alive for one month and dead the following, it's overall efficiency could come out as 50%. Holes and warm wires must both be considered in a correct acceptance calculation.//*[0.5cm]

The complete procedure for drift chamber inefficiencies involves:

1. **Find the pathologies.**
2. **Write the pathologies to database (MAP).**
3. **Incorporate inefficiencies in the analysis.**

3 Ungaro Li: motivations for a new algorithm

The motivations for developing a new method to identify dead and inefficient wires are the following:

- A better way of identifying dead wires was required. At first we also tried a sample of 6 wires but it didn't work because the limited sample space. Specifically, in some special cases when there are two or more hot wires in the group, it is impossible to remove them by judging from mean and sigma. Actually this was our main motivation.

- The same algorithm identifies warm wires and assign them a number between 0 and 1, dubbed the "efficiency". A wire with 65% efficiency is never knocked out with previous methods but it can affect the acceptance calculation. The new method will knock it out 35% of the time randomly.

- In principle, one has to identify the exact time a dead zone appears. N dead zones appearing at different times produce N run sub-periods (and even more sub-periods if they come alive again), each with its own DC status configuration. The simulation must then weight the number of events accordingly for each run sub-period. Having an *efficiency* instead of a *dead/alive* flag solves this nightmare scenario. For example 50% efficiency could mean that the wire is 50% efficient or that it was alive only half the time, say the first half of run period. In calculating a global acceptance for the run period acceptance, both scenarios lead to the same result (as long as the efficiencies are treated as correlated).

[2] An ADB board with a $60Hz$ varying gain of threshold might give a correlated efficiency.

- Having an efficiency means that one doesn't need to drop low occupancy wires during cooking as in the previous methods.
- Correlation between wires is also partially implemented[3]. Wires efficiencies that are correlated are knocked out *together* according to their efficiency (see section 7) .

4 Ungaro Li algorithm

For each wire $w(i,S)$ in sector S and wire index i, its next neighbors in the same sector $w(i-1,S)$ and $w(i+1,S)$ are taken, along with the corresponding wires in all the other sectors $w(i,S')$, $w(i-1,S')$, $w(i+1,S')$ to form a sample of 18 wires. This set of wires w_j, $j=1...18$ is used to calculate an expectation value for $w(i,S)$ in the following way. For each w_j we count how many of the other wires have occupancies within a certain percentage of w_j. Let's call this percentage variable **BUDDY**. Let's say BUDDY is 8 and consider an example coming from real CLAS data.

Layer	Sector 1	Sector 2	Sector 3	Sector 4	Sector 5	Sector 6
i_o	670080	674517	681877	678828	676214	2207
i_o-1	736412	734450	738558	746698	739865	5281
i_o+1	678419	665103	685710	105299	410887	677456

Table 2: *Wire occupancies taken from real CLAS data. The sample comes from the first* $50,000$ *events of each run of g6c. For each wire* w_i *a similar sample is taken.*

We can draw the histogram in Fig. 2. On the $x-axis$ is the wire index j, $j=1...18$ within the above sample of 18 wires, while the $y-axis$ denotes how many wires among this set have occupancy within 8% of the occupancy of w_j.

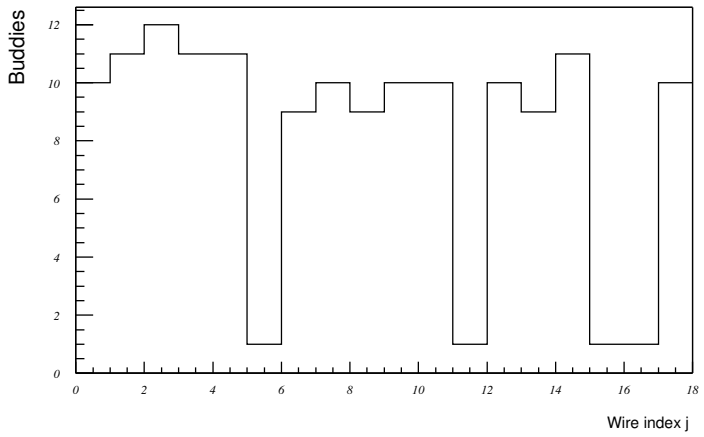

Figure 2: *The "buddies" histogram. Wire 3 has the biggest number of buddies.*

The first bin correspond to w_1, the first wire of the sample (occupancy 670080). Ten other wires in the sample have occupancy within 8% of w_1. Eleven wires have occupancies

[3]Due to the complexity of the correlations, only the simplest forms of correlations are considered.

within 8% of w_2 (second bin) and so on. Notice that only one wire has occupancy within 8% of w_6, which is w_6 itself (occupancy 2207). Same for w_{12}, w_{16} and w_{17} (occupancies 5281, 105299, 410887).

The algorithm identifies the largest buddies group by simply getting the histogram bin with the highest content, in this case group number 3 (which has bin content 12). Wires 6, 12, 16, 17, are not in this group because their occupancies are too low. Wires 10 and 11 are not in this group because their efficiencies are too high (746698 and 739865).

The expectation value A for this group is calculated averaging the occupancies of the wires within this group of "buddies". In this particular case, the average of the 12 wires is $A = 695191$. Wire 1 will be therefore assigned the efficiency

$$E = Occupancy/Expectation = 670080/695191 = 0.96387$$

Wire 6 will be assigned the efficiency

$$E = 2207/695191 = 0.0032$$

This algorithm uses the output of the program `pdu`, and utilities have been created which act on this output. Therefore we will briefly describe the use of `pdu`.

5 Running pdu

`Pdu` is a program that counts the number of raw hits for each wire. The cue version in $CLAS_BIN could be used. If you want to compile your own version of pdu, check it out with cvs:

```
cvs co utilities/pdu ; cd utilities/pdu ; make pdu [MAP=1 if you want
                      to use MAP instead of DB]
```

Running pdu is very easy:

`pdu` *filename*

produces an hbook file containing the occupancy histograms, a log file and a summary file. For example

```
pdu /cache/mss/clas/e1-6a/data/clas_030921.A00
```

will produce the log file *pdu_30921.out*, the summary file *pdu_wire_30921* and the hbook file *pdu_30921.hbk*.

5.1 Using the internal pdu algorithm

While running, pdu also flags wires according to its internal algorithm discussed in the introduction. The summary file contains the list the pathologies (dead and hot); each line contains sector - layer - wire - status .

hot wires - status = 100
dead wires - status = 1

One could use this file to put the information into the database (or map),

system = **DC_STATUS** subsystem = **sectorN** item = **status**

However, pdu comes with the option **-W**$<$ *runno* $>$ that does this automatically, so that:

```
pdu -W30921 /cache/mss/clas/e1-6a/data/clas_030921.A00
```

will put the calculated set of 0's and 1's in the database (or map) for run 30921, overwriting the entry if one exists already.

The hbook file contains the important six histograms h2X0 (X denoting the sector) of the occupancy, like the one in Fig. 1. We recommend to run pdu as part of the *pass-0* data processing, so that the first file in each run is diagnosed.

6 How to use Ungaro Li algorithm

To take advantage of the new method, some utilities had been implemented in the pdu directory. `dc_eff , groups, adbboard.pl, seclaywir.pl` to calculate efficiencies, find correlations and assign a *group id* to each group of correlated wires.

6.1 How to find wires efficiencies

The utility dc_eff is used to calculate the efficiency according to the above algorithm. This program is in the pdu directory and the input file is a root file. If you have processed, as suggested, the first files of each run for your run period, then you would have N hbook files. You have to convert them to root [4] and then you can sum them all together with the command `hadd sumfile.root <list of root files>`.
The syntax of *dc_eff* is

```
dc_eff <options> <root file>
```

The user can choose the BUDDY variable as well as the low and high efficiency limit. For example, to mark dead all wires with less than 40% efficiency, use *-l0.4*. The complete list of options is:

```
dc_eff  <options> <root file>
        options are:
        -h              Print this message.
        -o<filename>    Output ROOT file name.
        -a<filename>    Output ASCII file name.
        -m[#]           Min. layer number to be analyzed (default = 1).
        -M[#]           Max. layer number to be analyzed (default = 36).
        -l[#]           if efficiency less than this,    it will be written as 0 (default = 1e-3).
        -H[#]           if efficiency greater than this, it will be written as 1 (default = 0.9).
        -B[#]           wires with occupancy within this percentage are considered buddies (default = 16).
        -v              Verbose mode.
```

For example, choosing $l = H = p$ will flag wires dead or alive, i.e. 1 or 0, if their efficiency is less/greater than p, thus producing a table map compatible with the old method. The command

[4] to convert file.hbook to root format you can type `h2root file.hbook`.

```
dc_eff -oeff.root -amap.txt -l0.2 -H2 -B10 occupancy.root
```

- will calculate efficiencies. Only wires with occupancy within 10% of each other inside each sample will be considered to calculate the expectation value.
- will produce the eff.root file with two sets of histograms, one with calculated efficiencies and one with the efficiencies with the cuts l,H: all wires with $E < 0.2$ will shows as zeros, and all wires with $E > 2$ will shows as ones[5].
- will produce a map.txt file containing the efficiency values with the cuts l,H.

The verbose mode is used to display single wire set; example:

```
Wire:  1  -  Occ:    7.19292e+06  -  buddy YES   -  wire exist = 1
Wire:  2  -  Occ:    6.36085e+06  -  buddy YES   -  wire exist = 1  <--- wire under current analysis
Wire:  3  -  Occ:    6.56806e+06  -  buddy YES   -  wire exist = 1
Wire:  4  -  Occ:    6.54793e+06  -  buddy YES   -  wire exist = 1
Wire:  5  -  Occ:              1  -  buddy NO    -  wire exist = 1
Wire:  6  -  Occ:         957698  -  buddy NO    -  wire exist = 1
Wire:  7  -  Occ:    6.85542e+06  -  buddy YES   -  wire exist = 1
Wire:  8  -  Occ:    6.29898e+06  -  buddy YES   -  wire exist = 1
Wire:  9  -  Occ:       6.49e+06  -  buddy YES   -  wire exist = 1
Wire: 10  -  Occ:    6.50276e+06  -  buddy YES   -  wire exist = 1
Wire: 11  -  Occ:    6.88094e+06  -  buddy YES   -  wire exist = 1
Wire: 12  -  Occ:    6.67128e+06  -  buddy YES   -  wire exist = 1
Wire: 13  -  Occ:    7.75228e+06  -  buddy YES   -  wire exist = 1
Wire: 14  -  Occ:    6.96606e+06  -  buddy YES   -  wire exist = 1
Wire: 15  -  Occ:    7.21386e+06  -  buddy YES   -  wire exist = 1
Wire: 16  -  Occ:    7.31219e+06  -  buddy YES   -  wire exist = 1
Wire: 17  -  Occ:    7.49755e+06  -  buddy YES   -  wire exist = 1
Wire: 18  -  Occ:     7.3879e+06  -  buddy YES   -  wire exist = 1

Max bin = 1  which has 15 elements
Mean = 6.90619e+06  Efficiency = 0.921036
```

[5] a hot wire can have efficiency higher then 1.

The root file contains the efficiency plots. Here's an example for *g6c* run period:

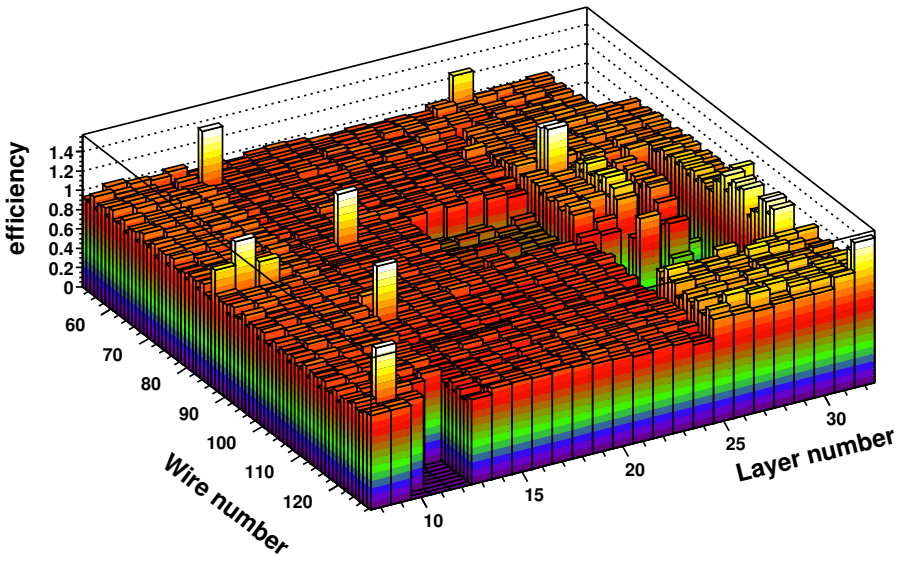

Figure 3: *The efficiency plot for g6c run period, sector 6, zoomed in. Dead, hot and warm wires are visible. Wires in the depleted region (layers* 18 *to* 23*, wires* $\approx$ 90*) belong to the same HV board and all have efficiency* $\approx 60\%$*.*

A typical summary of dc_eff:

```
Results:
Efficiency range     Number of wires       Percentage

    < 0.001                 565           0.0160584
 0.001 % 0.1                898             0.025523
  0.1  % 0.2                138           0.00392224
  0.2  % 0.2                 95           0.00270009
  0.3  % 0.4                171           0.00486016
  0.4  % 0.5                212           0.00602547
  0.5  % 0.6                199           0.00565598
  0.6  % 0.7                216           0.00613915
  0.7  % 0.8                355            0.0100898
  0.8  % 0.9               1717            0.0488006
  0.9  % 1.0              13219             0.375711
  1.0  % 1.5              16688             0.474307
  1.5  % 2.0                302           0.00858345
  2.0  % 10.0               330           0.00937926
    > 10                    79           0.00224534

Total number of wires analyzed: 35184
```

The ASCII file is needed for putting the constants into the database. Before doing that, the user should find correlations.

6.2 Wires correlations

At the moment there is no quantitative way to identify correlations. Plots of occupancy and/or efficiency versus time of different wires do not show strong evidence of correlations on a time scale of several minutes (around 10 minutes, the time of writing one file within a run during acquisition). Correlations do show up in DC **reconstructed** hits as shadows next to suspicious regions, as explained in the following example. Suppose we have a square warm region which contains $N \times N$ wires. If the efficiency of each wire is p (the same for all wires in that region) and they are independent, then the probability P of four of them not firing at the same time (this is the probability of losing a track passing through this region) is p^4, which is only $P = p^4 = 6.2\%$. However, if the wires are correlated, that probability is $P = p = 0.5$ (9 times higher) because they are all dead at the same time. Tracks traversing a correlated warm region are thereby affected, and the acceptance is distorted. So one possibility of identifing correlations is to plot the DC reconstructed hits on tracks. There's no utility to do it yet so one has to do it "by hand".

There's a more direct approach. If all the wires in a depleted region belong to the same ADB board, and no other wires belong to that board, we could safely assume that there is correlation. This is exactly the case as shown in Fig 3. All the wires are attached to the same ADB crate r2s6st (see appendix A). The easiest and fastest way to identify boards is the following:

Inside dc_eff directory, there's a directory "plot" with macro "plot.C". First, copy the dc_eff root output in this directory with name "eff.root". Then run `root plot.C`. This will produce 12 efficiency postscript files, 2 per sector, as the one in Fig 4. Suspicious warm regions are fairly visible in these plots.

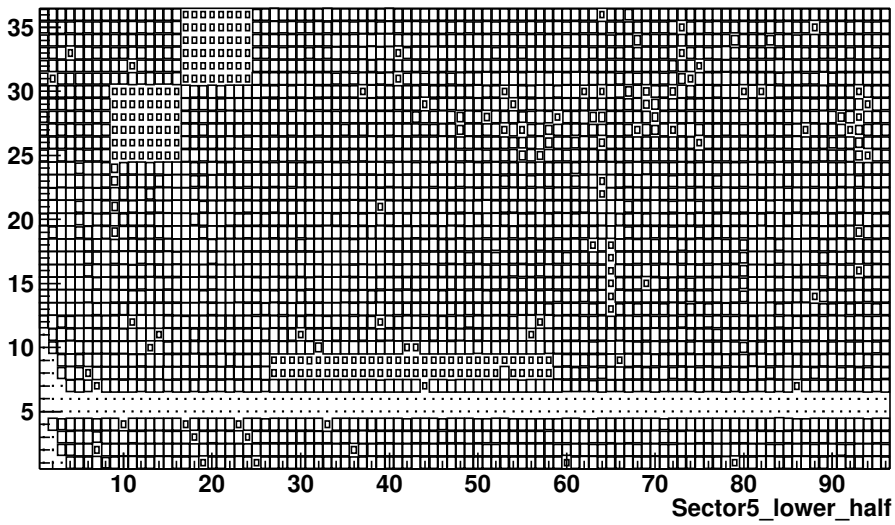

Figure 4: *Sector 5, wires up to 100. Warm regions are visibile in upper left corner.*

The perl scripts *adbboard.pl* and *seclaywir.pl* (usage is explained in appendix A) allows us to determine which wire belongs to which ADB board.

6.3 Incorporating correlations

One can use the utility `groups` (located in the same directory dc_eff) to

- assign a GROUP number to a region. GPP will treat all the wires in the same group as correlated.
- assign efficiency 0 to that region. GPP will kill all the wires in that region.

Add the line:

Sector - min wire - max wire - min layer - max layer - Group id

to a file (e.g. group.txt), then just run the program *groups*

```
groups -oOUTPUTFILE.map -ggroup.txt INPUTFILE.map
```

where INPUTFILE.map is the ASCII map created with dc_eff. OUTPUTFILE.map will contain the same entries as OUTPUTFILE.map except for the groups selected. For those wires, the MEAN is calculated and the number $1000*GROUPID+MEAN$ replaces the efficiency, so all wires are labeled with the same efficiency and as belonging to the same group. In the case special $GROUP$ is set to 0, then the efficiency for all wires in this group will be set to zero.

6.3.1 Example 1

Fig. 3 shows a depleted region in sector 6 whose wires are attached to the same ADB board, as previously mentioned (see appendix A).

wires 71 to 90 in layers 21 to 25, efficiency $\approx 70\%$

Suppose that we also find out that

wires 23 to 32 in layers 30 to 36 in sector 3, efficiency $\approx 54\%$

are correlated too. To assign GROUP ID # 1 to the wires in sector 6 and GROUP ID # 2 to the ones in sector 3 add the following numbers to group.txt

Sector	min wire	max wire	min layer	max layer	Group id
6	71	90	21	25	1
3	23	32	30	36	2

then run *group* as discussed before[6]. The program will calculate the mean for the two groups of wires, let's say $M_1 = 0.701$ and $M_2 = 0.5398$. The program will also write in the ASCII map the number 1000.701 for all wires in the first group and 2000.5398 for all wires in the second group.

[6] do not write the titles in group.txt, just the numbers.

6.3.2 Example 2

Suppose we found out that the two regions in Fig.4 (upper left) have an efficiency $\approx 77\%$ that actually comes from the fact that the wires died at the same moment sometime during the last part of the run but were alive during the first (and longer) part. Suppose also that the warm region in layer 8 and 9 is really warm all the time. We want to account for this in the simulation, and the way to do it is add the lines

```
5   9   17  24  31  3
5   17  24  32  36  3
5   26  59  8   9   4
```

to the file groups.txt. Notice the first two GROUP IDs are the same! *group* will produce three numbers, i.e. $M_1 = 0.765$ and $M_2 = 0.745$ $M_3 = 0.66$ for the three diffent regions but it will assign the SAME group number 3 to the first two sets and group number 4 to the last set. The wires in the first two sets are correlated not only within their group but also with the wires in the other group because a single LV fuse feeds both square region in Fig. 4. Wires in the third group are independent.

6.3.3 Example 3

Suppose we found out that in sector two there is a depleted region (wires 93 to 102, layers 12 to 14) with an efficiency of 50%. But these are **noisy** wires that by chance happen to have an occupancy comparable with the good wires. We want to kill those in the simulation with the following entry in groups.txt

```
2 93 102 12 14 0
```

The special 0 assignment to the group id will put all 0 in the map for these wires for run 30921, so they will be knocked out by GPP.

6.4 Writing wire efficiencies to the database (map)

The map created with `dc_eff` and `groups` has to be placed in the database (or map) for use with GPP. This is done with the program `groups` using the option `-W#`. The command

```
groups -onewmap.txt -ggroups.txt -W30921 map.txt
```

will read map.txt and produce newmap.txt according to data given in groups.txt. It will furthermore put the newmap.txt constants in the database, system = GPP, subsystem = DC_WIRE.

One can also fill the system DC_STATUS with numbers from map.txt. In this case, make sure that only 0 and 1 are present in the map (see section 6.1) because DC_STATUS accepts only integer numbers. For instruction on how to fill the database from an ASCII file, consult the offline software page at

```
http://clasweb.jlab.org/offline/offline_soft.html
```

7 Incorporating inefficiencies

After identifying inefficiencies with any of the algorithms described above, the user can adopt different strategies. The user still has the option of running GPP and knocking out dead wires according to DC_STATUS map as in the past, but the program GPP has been modified to operate with efficiencies as well and to use the new entry in the GPP system that contains the efficiency. In this case, the efficiency is used as a *probability* to keep the wire in the simulation. A random number is thrown for each wire and compared with this probability to decide wether to keep that wire or knock it out. In case of correlations, only one random number is thrown **per group** so for each event all the wires in that group will be kept or knocked out simultaneously.

The default version in $CLAS_BIN can be used. Also if you want to compile your own version of gpp, check it out with cvs:

```
cvs co gpp ; cd gpp ; make [MAP=1 if you want to use MAP instead of
                                    DB]
```

7.1 How to knock out wires in the montecarlo simulation

7.1.1 using DC_STATUS system

The system DC_STATUS contains the map of flagged dead wires determined from pdu's internal algorithm, the Todor/Niculescu algorithm, or the Ungaro/Ji algorithm (as described in section 6.4), whichever the user chooses. To drop all these flagged dead wires in the file SIMULATION.bos type:

```
gpp -P0x1 -R30921 -oNEW_SIMULATION_DC30921.bos SIMULATION.bos
```

where an output name is suggested and run number 30921 is used to retrieve DC status (type gpp -h for the complete list of options).

7.1.2 using GPP system

As mentioned in section 6.1 and 6.4, the Ungaro/Ji algorithm can also produce a dead/alive flag, which is saved in GPP system. To drop all these flagged dead wires in the file SIMULATION.bos type:

```
gpp -Y -R30921 -oNEW_SIMULATION_DC30921.bos SIMULATION.bos
```

7.2 How to knock out wires in real data

To remove the dead wires in the data which are flagged by the DC_STATUS system, the program `a1c` can be used with the process flag `-P0x8000` (as in during *g6c* cooking). See CLAS NOTE 02-017 for details.

7.3 How to reproduce inefficiencies in the simulation

After producing the correct entries in the GPP system with the programs `dc_eff` and `groups`, the command

```
gpp -Y -R30921 -oNEW_SIMULATION_DC30921.bos SIMULATION.bos
```

will reproduce inefficiencies and found correlations.

The following picture shows a comparison of the depleted region in sector 6 for data and GSIM simulation. The wires in this region are not treated as correlated in this case.

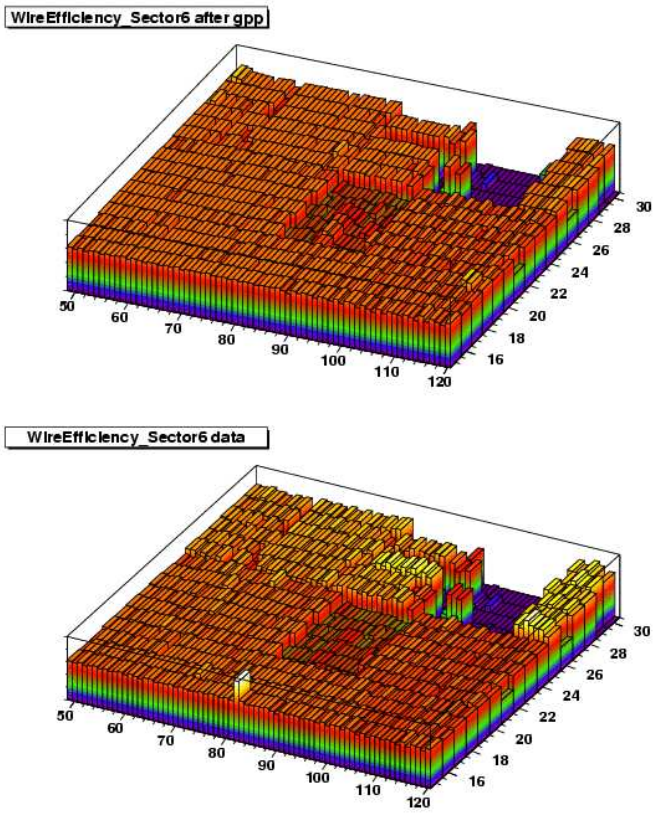

Figure 5: *Comparison of GSIM simulation (top) with real data (bottom) for DC efficiency after gpp. The data is taken from g6c running period. The depleted region of* 60% *efficiency is nicely reproduced in the simulation. Notice also other similarities, such as dead regions and individual warm wires.*

A Wires and ADB utilities

We'll illustrate how to find out if the wires in a depleted region belong to the same ADB board with the example in Fig.3 already mentioned. We'll make use of the perl scripts *seclaywir.pl* and *adbboard.pl*. First of all, let's pick one wire in that region, i.e. layer 22, wire 85 (sector is 6). Run *seclaywir.pl* and answer the question properly: sector, layer, wire. The output is

```
signal cable name = st6.1
adb crate name = r2s6st, located on spaceframe - Level3 - south
adb crate number = 27
adb slot = 6
adb connector = 3
adb pin = 5
mux connector = 1
mux pin = 13
signal length = long
tdc crate number = 10, located on spaceframe - Level3 - south
tdc slot = 20
tdc connector = 4
tdc pin = 13
```

The next step is to run *adbboard.pl* giving it the information we just obtained: ADB board = r2s6st , slot = 6, signal length = long. The output will show the list of wires attached to that board

```
 Sec Lay Wire    Sec Lay Wire
  6  20  79       6  22  79
  6  23  80       6  19  80
  6  24  80       6  20  80
  6  21  80       6  22  80
  6  23  82       6  24  82
  6  20  82       6  22  82
  6  19  83       6  23  83
  6  24  83       6  21  83
  6  19  85       6  23  85
  6  20  85       6  24  85
  6  21  85       6  22  85
  6  19  86       6  21  86
  6  20  87       6  22  87
  6  23  88       6  19  88
  6  24  88       6  20  88
  6  21  88       6  22  88
  6  22  90       6  23  90
  6  24  90       6  20  90
  6  19  91       6  23  91
  6  24  91       6  21  91
  6  19  93       6  23  93
  6  20  93       6  24  93
  6  21  93       6  22  93
  6  19  94       6  21  94
```

As you can see, all the wires belong to the depleted region.