Update on field propagation

J. Apostolakis

$\mathbf{x_0}$, $\mathbf{p_0}$
$\mathbf{x_1}$, $\mathbf{p_1}$, $\mathbf{\Delta x}$, $\mathbf{\Delta p}$
$\mathbf{x_2}$, $\mathbf{p_2}$, $\mathbf{\Delta x}$, $\mathbf{\Delta p}$

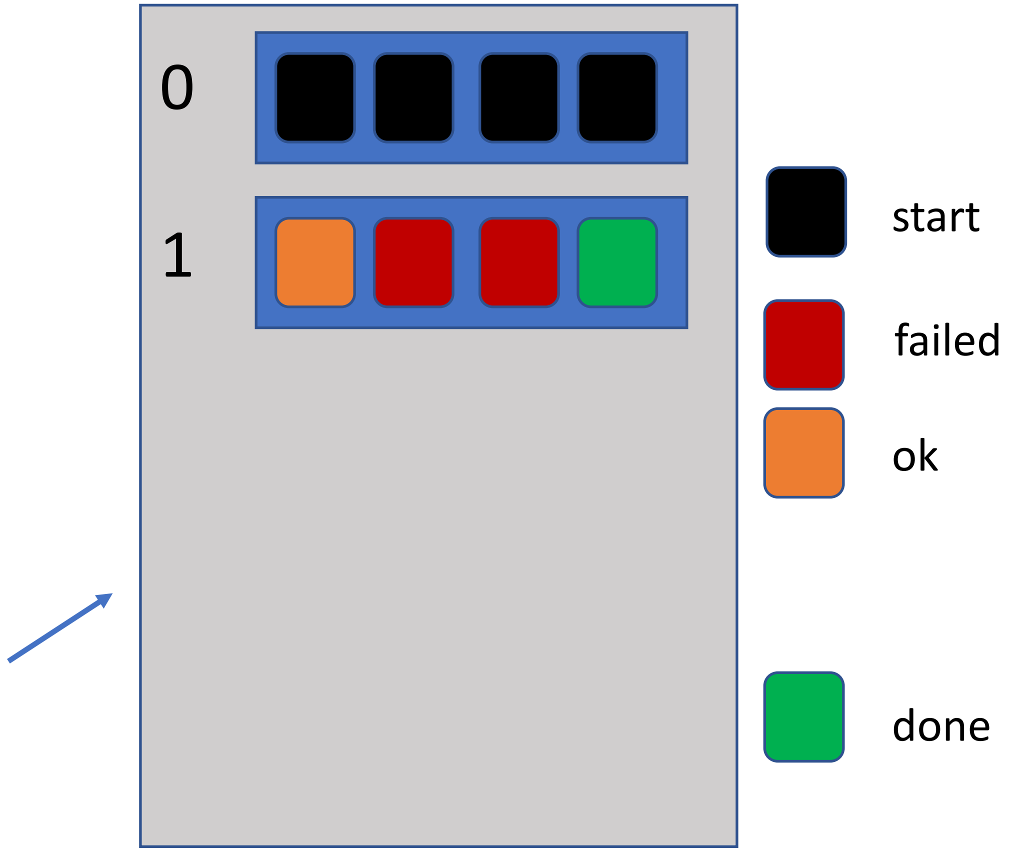
Field propagation overview

- Field propagation involves solution of Ordinary Differential Equation
  - Typically Runge-Kutta methods are used (as in Geant4)
- In GeantV created **vectorised Runge-Kutta** propagation
  - Charged tracks in a basket are sent to the FieldPropagation classes
  - Vectorised over tracks
- Challenge to ensure that all vector lanes are working & use mostly vector operations

- Motion in field requires solving ODE for endpoint **x**, **p** after length s
- Runge-Kutta step: evaluate B-field, estimate **x, p, Δx, Δp**
- Successful if $|\mathbf{\Delta x}| < \varepsilon\, s$ & $|\mathbf{\Delta p}| < \varepsilon\, |\mathbf{p}|$
- Each step of a Runge-Kutta algorithm is easy to vectorise
  - But different tracks (vector lanes) can take different number of iterations to finish integration
  - The ‘driver’ class which calls the RK ‘stepper’ must play coordinate the work

Simple Vector propagation

- First version “SimpleIntegrationDriver”
  - sought a good step in each lane
- A step can either
  - Fail,
  - Succeed but not get to the end (“ok”)
  - Finish the integration (“done”)



⬛ start

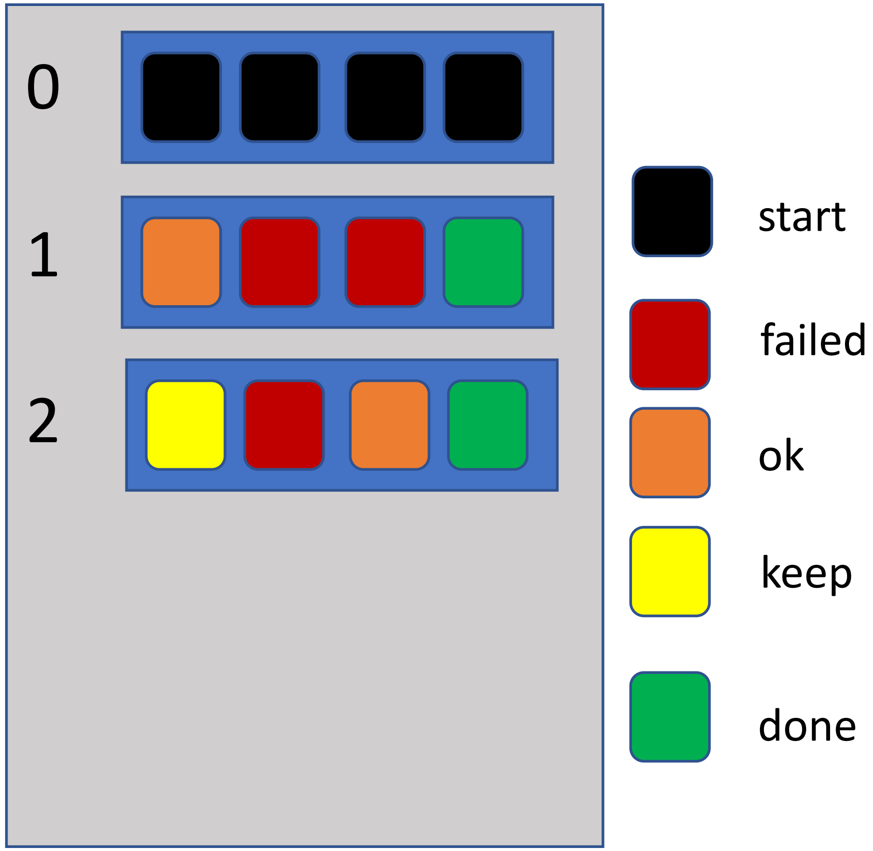
🟥 failed

🟧 ok

🟩 done

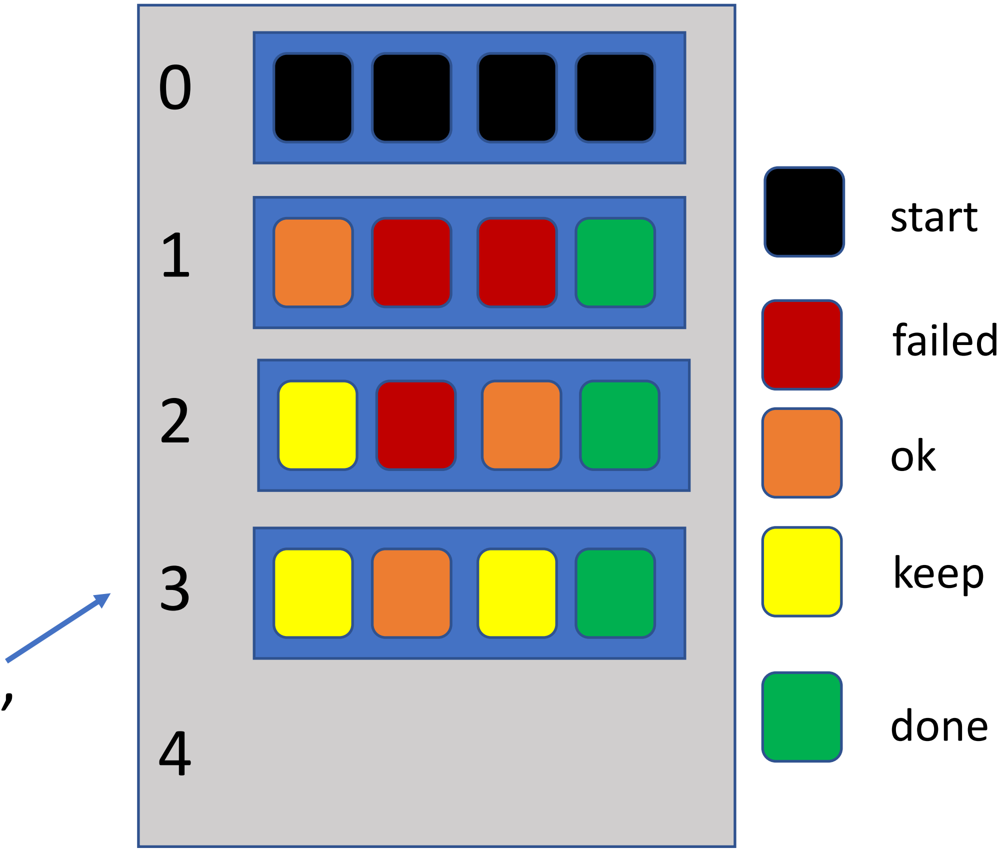
Simple Vector propagation

- First version sought a good step in each lane
- A step can either
  - Fail
  - Succeed but not get to the end (“ok”)
  - Finish the integration (“done”)
- If a lane is “done”, in ‘SimpleIntegrationDriver”
  - it must maintain its result (“keep”)



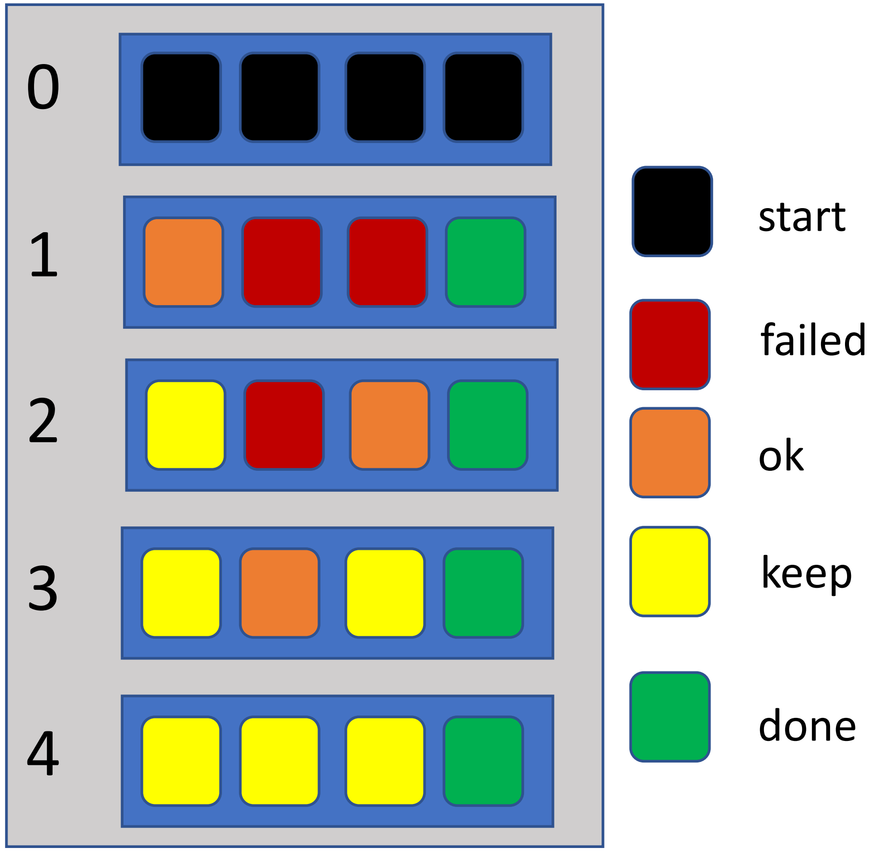
Simple Vector propagation

- First version sought a good step in each lane
- A step can either
  - Fail
  - Succeed but not get to the end (“ok”)
  - Finish the integration (“done”)
- If a lane is “done”, in ‘SimpleIntegrationDriver”
  - it must maintain its result (“keep”)



Simple Vector propagation

- First version sought a good step in each lane
- A step can either
  - Fail
  - Succeed but not get to the end (“ok”)
  - Finish the integration (“done”)
- If a lane is “done”, in ‘SimpleIntegrationDriver”
  - it must maintain its result (“keep”)



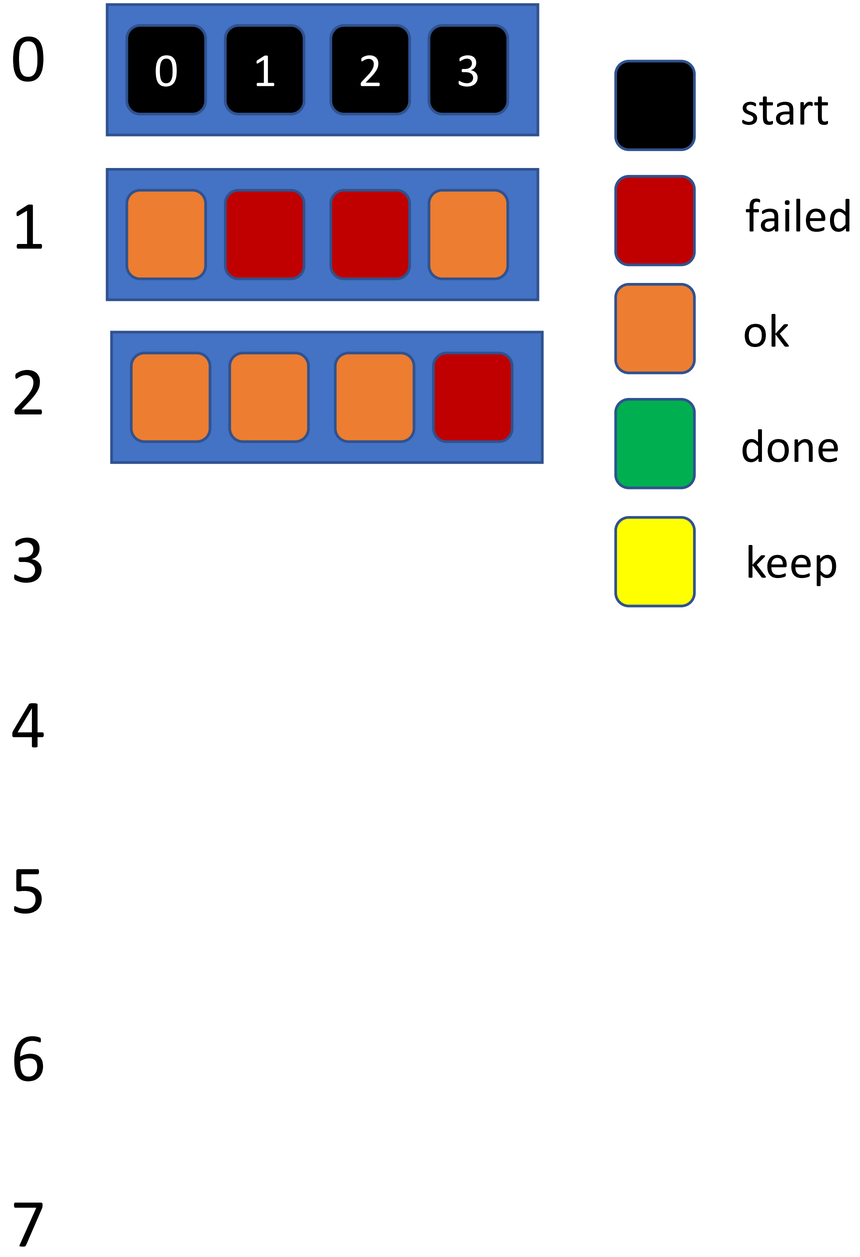
- Cross checked against the ‘scalar’ integration
- Adequate only if the ***success rate*** of steps is very ***high***
  - Good speedup for constant field
  - ***Poor results*** for non-uniform field CMS setup

Observations

- Simplest vector algorithm, but implementation is still complex
    - ‘OneGoodStep’ 109 lines ‘cleaned’(no prints, comments, blanks, ‘bare’ {})
    - ‘AccurateAdvance’ 121 lines ’cleaned’
    - The full class is over 2000 lines (with prints, comments, load & test methods)
- This initial version worked in Feb/Mar 2018
    - The applications were updated later to work with non-uniform fields
- In the past 2.5 months
    - Restored changes from March-May 2018
    - With improved checks, made 2 small fixes to fully agree with scalar runs
        - (exponent step growth, error normalization)

Vector propagation v 2.0

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval



⬛ start

🟥 failed

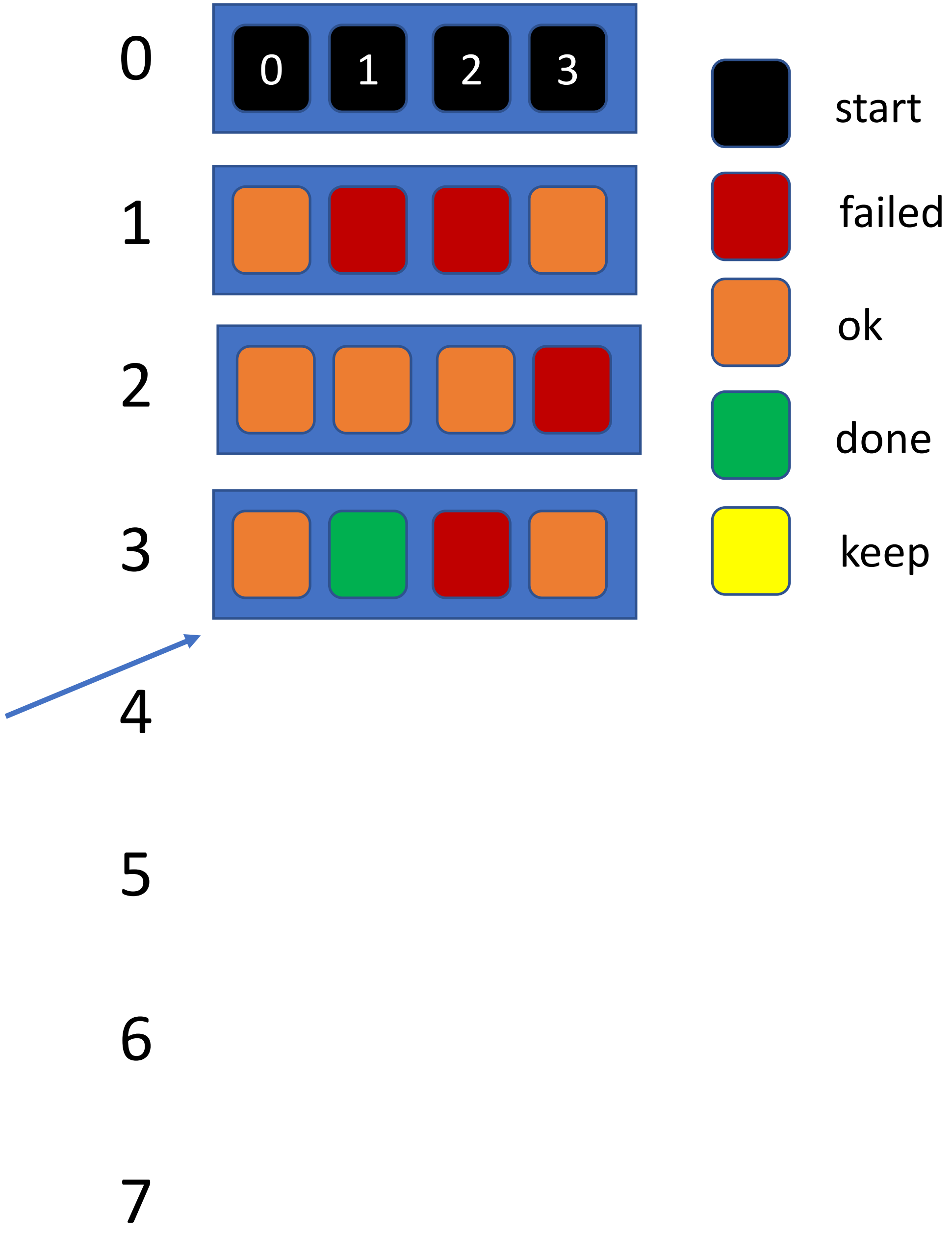
🟧 ok

🟩 done

🟨 keep

Vector propagation v 2.0

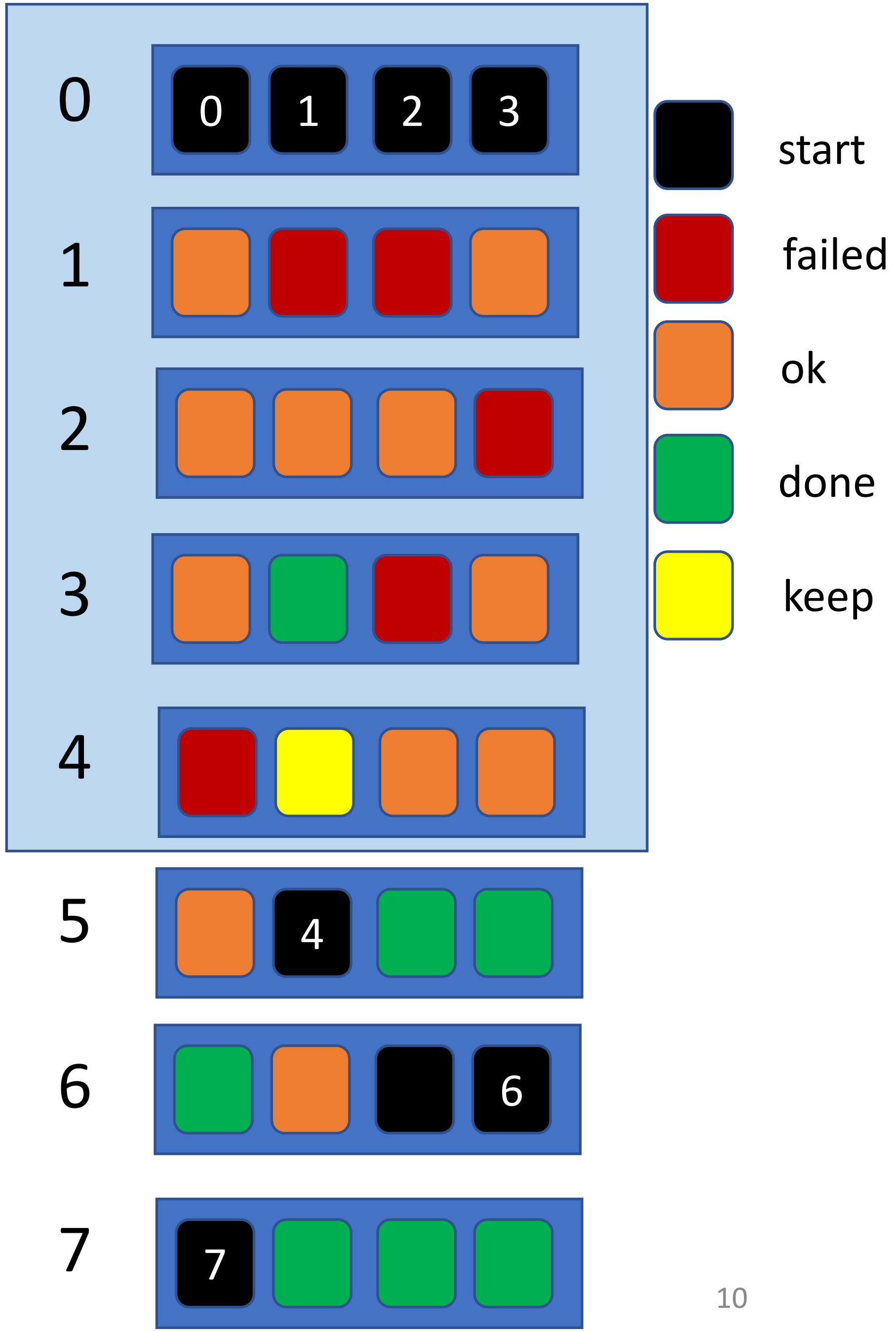
- ‘Keep Stepping’ in lanes with work
    - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval



Could finish here.
At least one lane is done

Vector propagation v 2.0

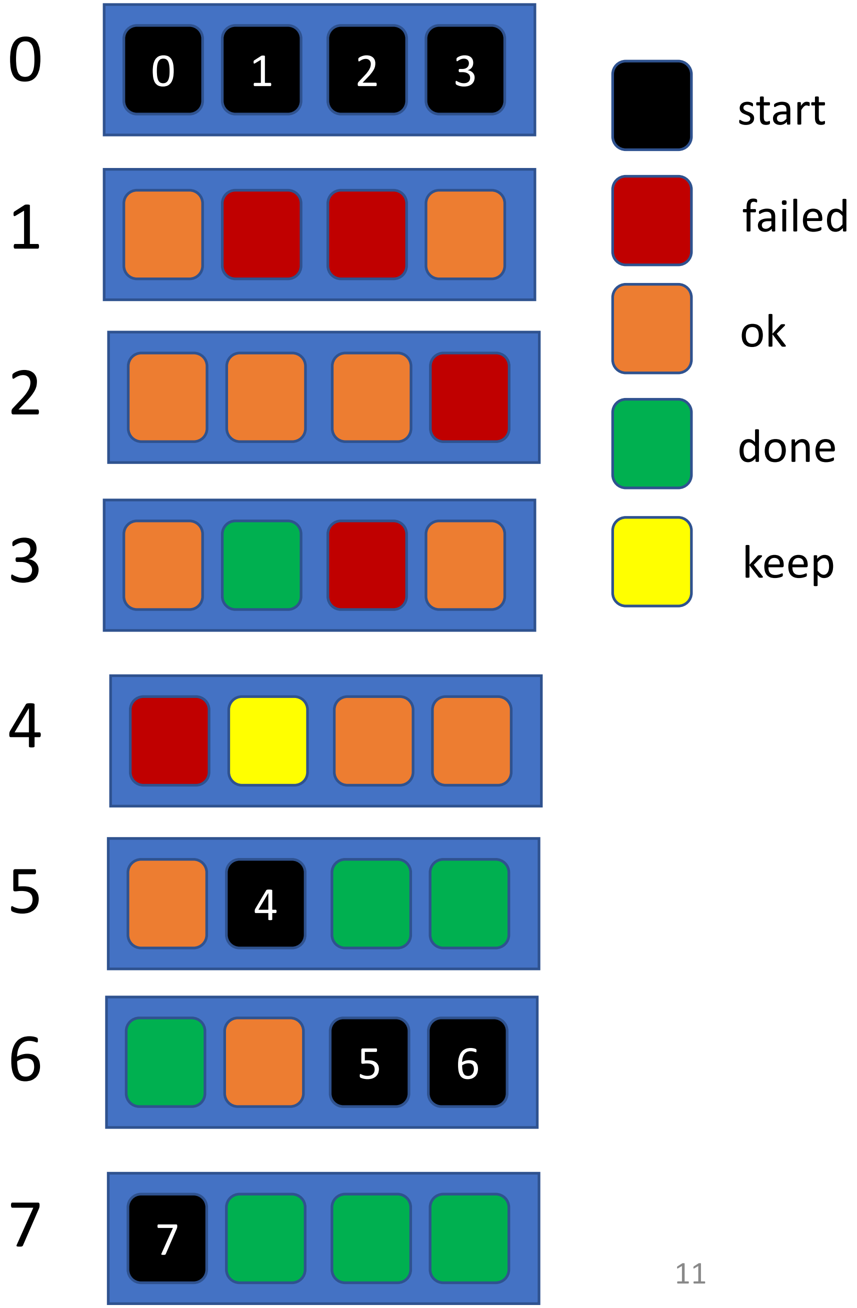
- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval



Decided to do one more.
Now return for ‘refill’

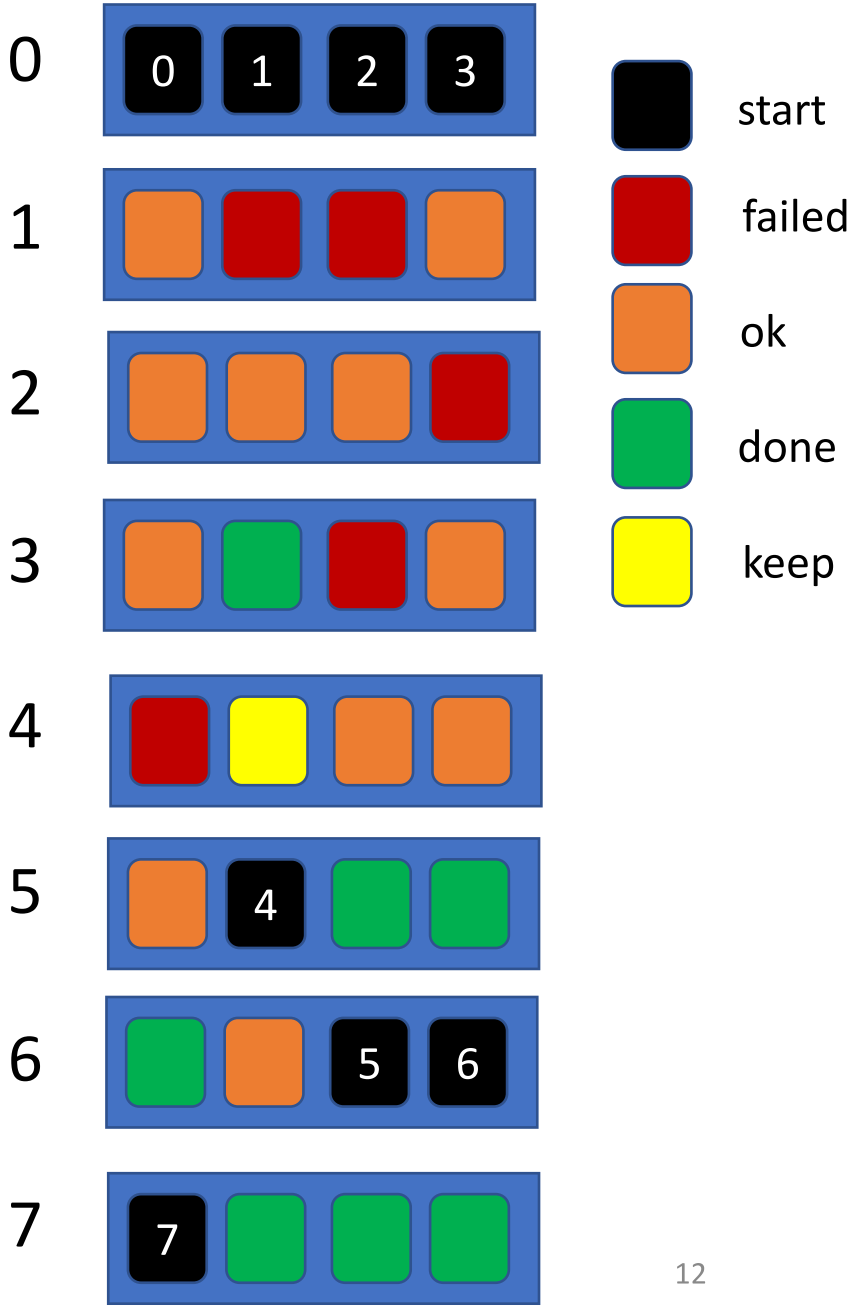
Vector propagation v 2.0

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval



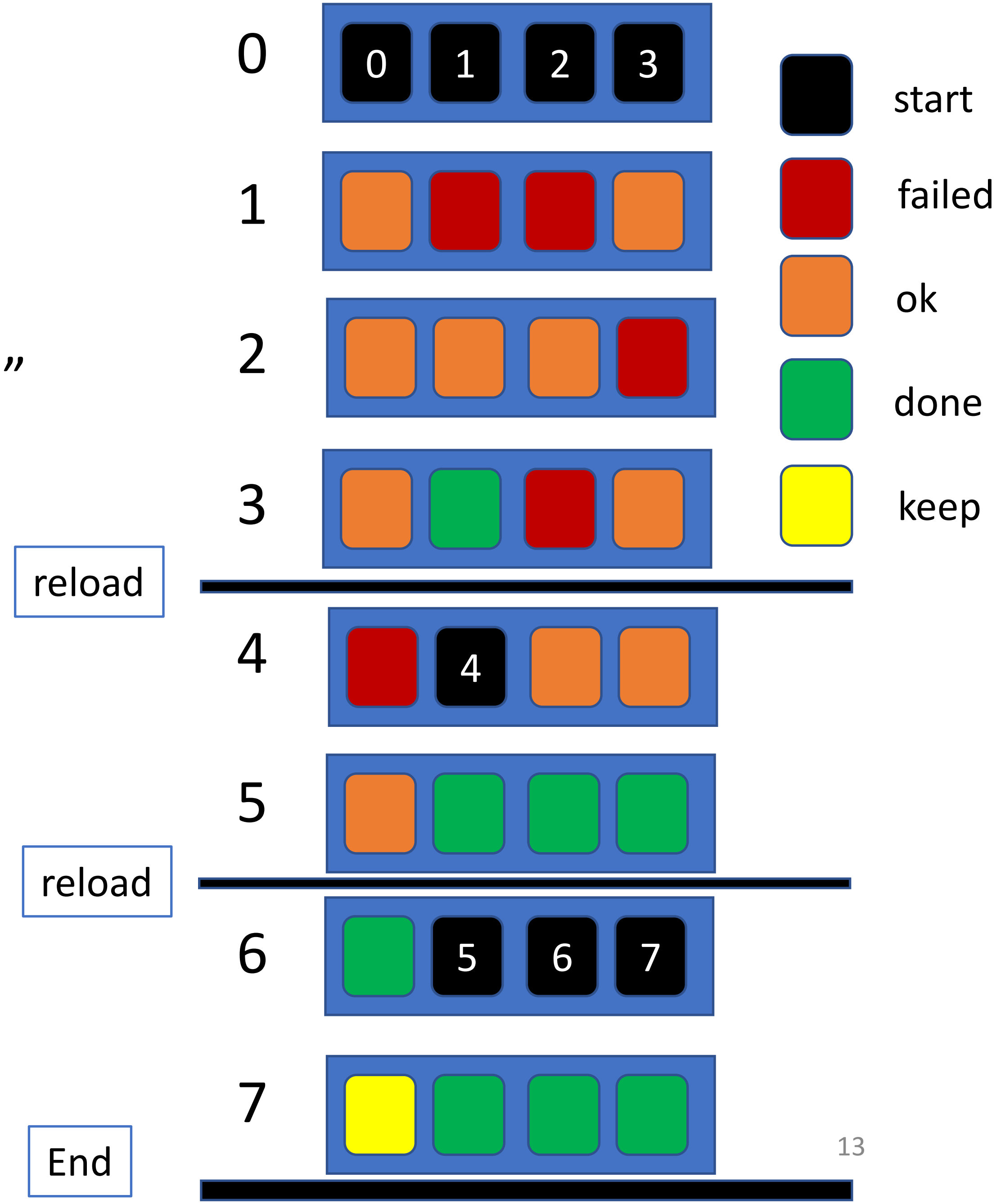
Vector propagation v 2.0

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval



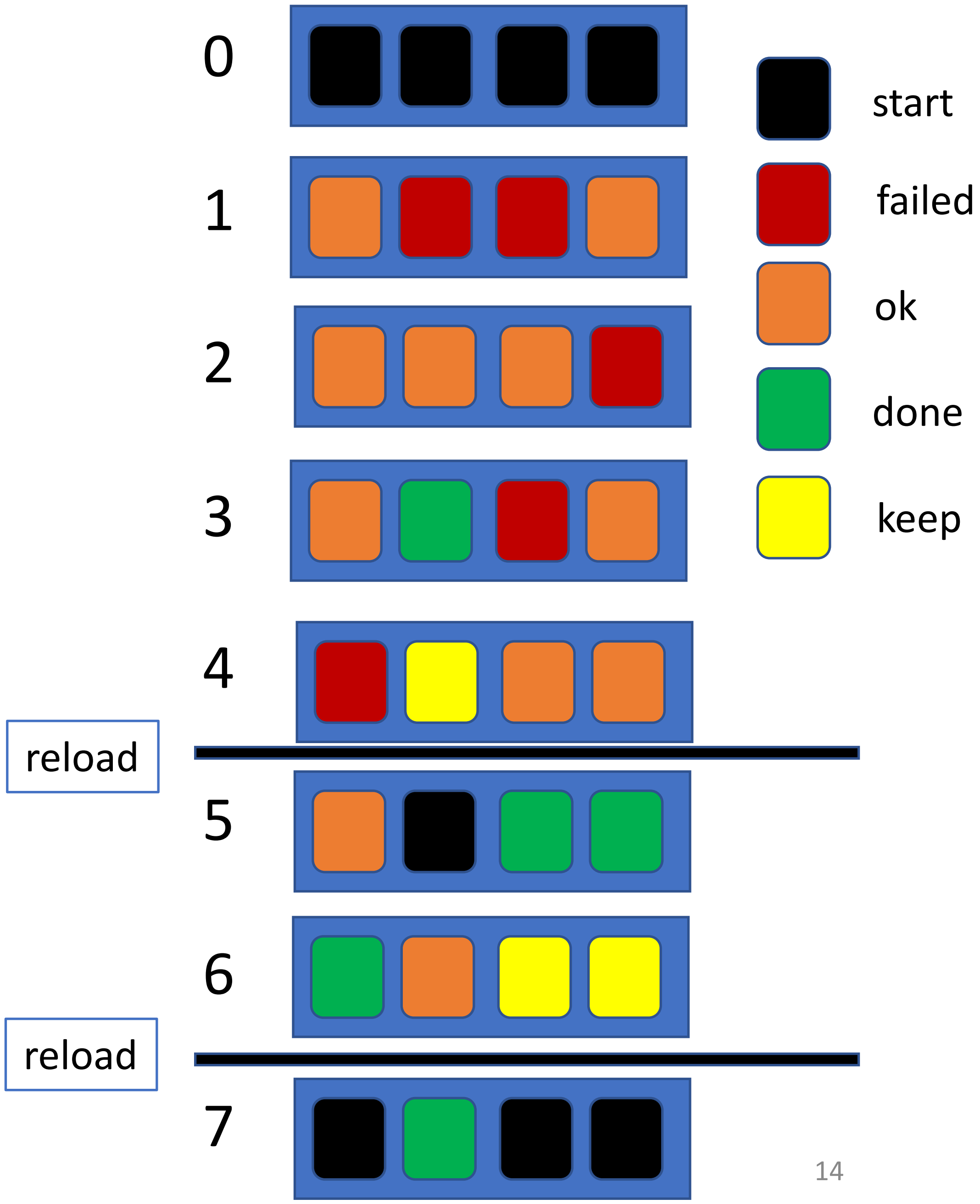
Vector propagation v 2.0

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval
- Potential Criterion for ‘reload’:
  - At least one finishes



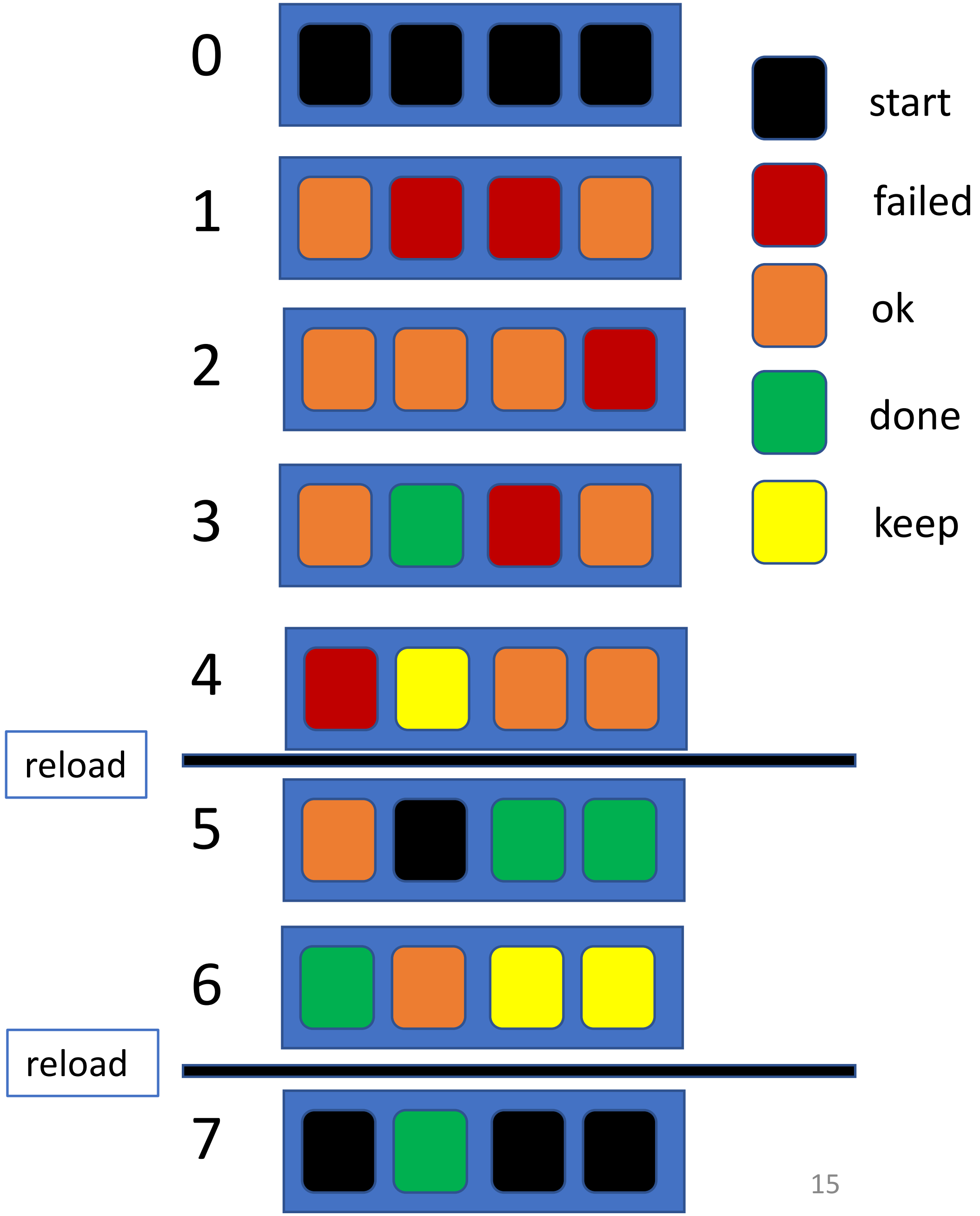
‘Rolling’ Integration Driver

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval
- Potential Criterion for ‘reload’:
  - At least one finishes
  - One+ finish, plus X steps (**current**, X=1)



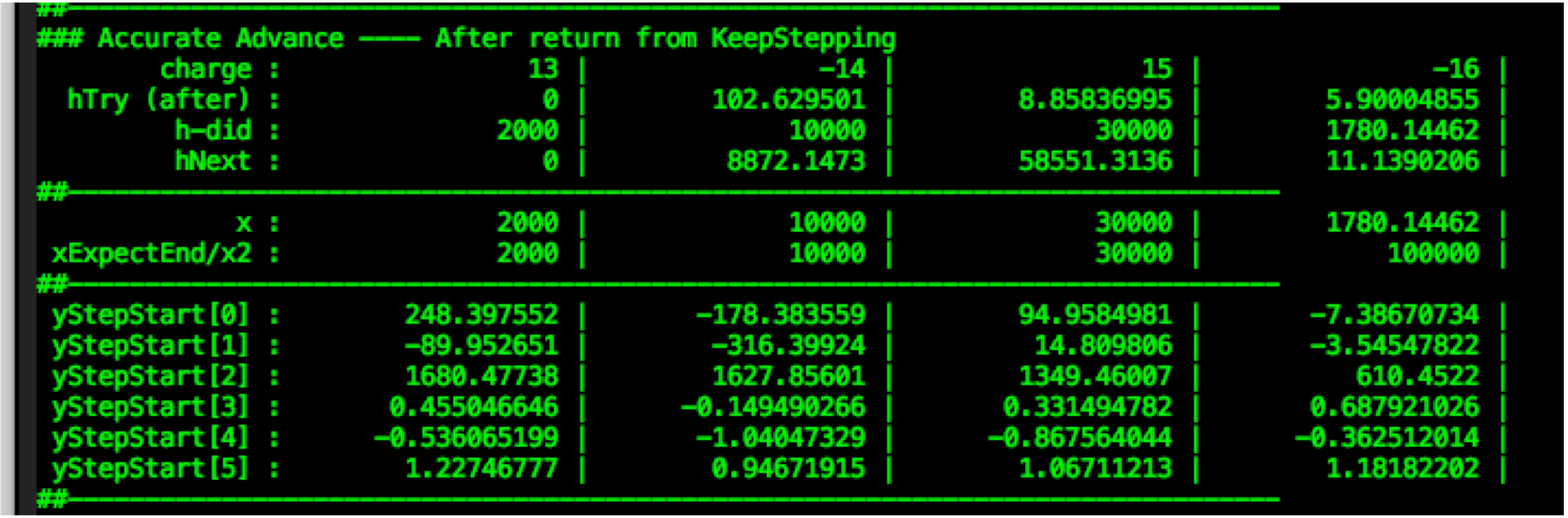
‘Rolling’ Integration Driver

- ‘Keep Stepping’ in lanes with work
  - Keep all/most lanes working “all the time”
- Reload when lanes reach the end of integration interval
- Potential Criterion for ‘reload’:
  - At least one finishes
  - One+ finish, plus X steps (**current**, X=1)
  - …
  - n>threshold finish
  - All finish (will test – but likely not the best.)



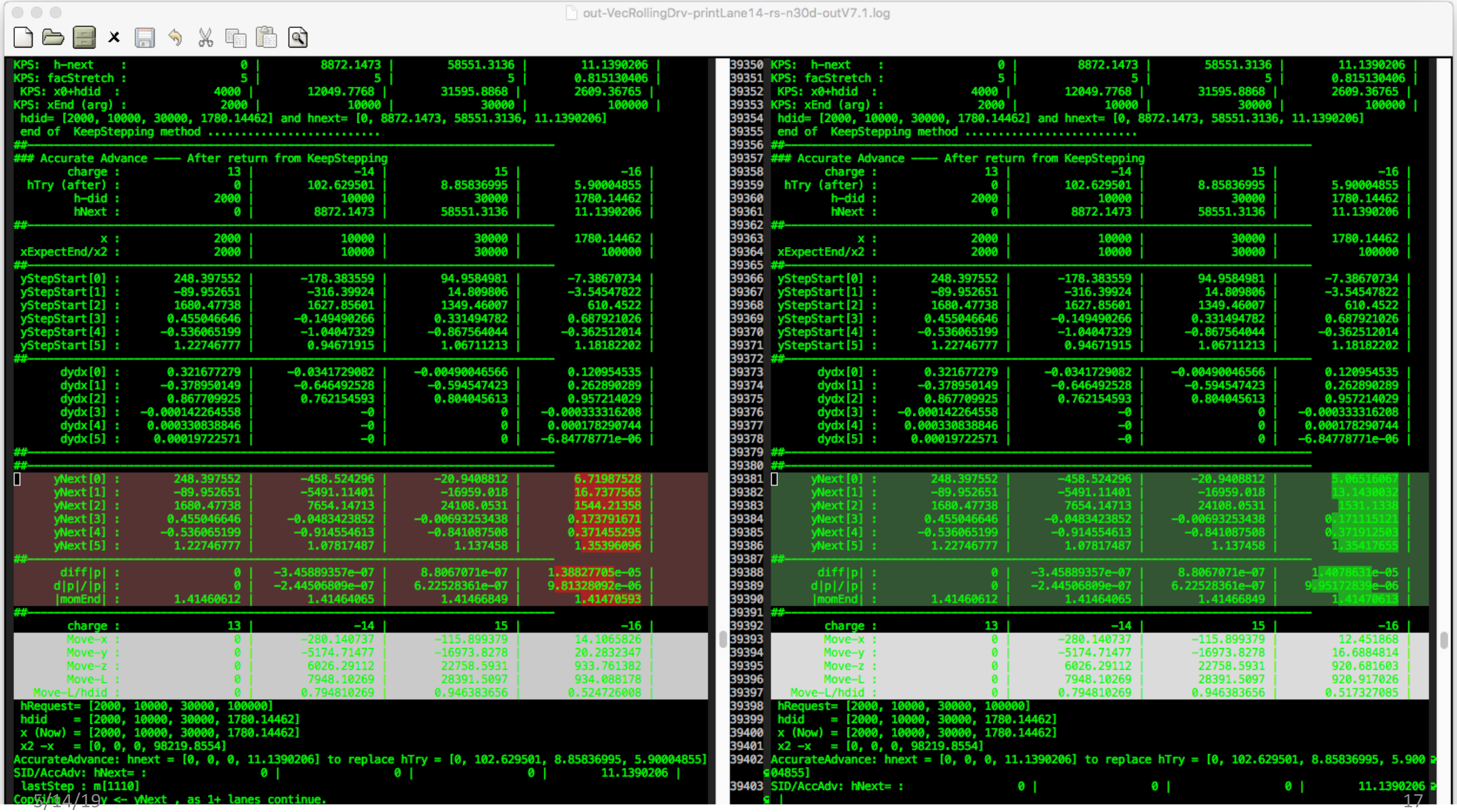
Tools to monitor / compare / diagnose

- Utilities to print values in columns: (Real_v, Real_v[], Index_v, Bool_v)



```
##----------------------------------------------------------------------------------
### Accurate Advance ---- After return from KeepStepping
        charge :            13 |         -14 |          15 |         -16 |
  hTry (after) :             0 |  102.629501 |  8.85836995 |  5.90004855 |
         h-did :          2000 |       10000 |       30000 |  1780.14462 |
         hNext :             0 |   8872.1473 |  58551.3136 |  11.1390206 |
##----------------------------------------------------------------------------------
             x :          2000 |       10000 |       30000 |  1780.14462 |
 xExpectEnd/x2 :          2000 |       10000 |       30000 |      100000 |
##----------------------------------------------------------------------------------
 yStepStart[0] :    248.397552 | -178.383559 |  94.9584981 | -7.38670734 |
 yStepStart[1] :    -89.952651 |  -316.39924 |   14.809806 | -3.54547822 |
 yStepStart[2] :    1680.47738 |  1627.85601 |  1349.46007 |    610.4522 |
 yStepStart[3] :   0.455046646 | -0.149490266 | 0.331494782 | 0.687921026 |
 yStepStart[4] :  -0.536065199 | -1.04047329 | -0.867564044 | -0.362512014 |
 yStepStart[5] :    1.22746777 |  0.94671915 |  1.06711213 |  1.18182202 |
##----------------------------------------------------------------------------------
```

- Utility to print the status of the lane for a particular track Id
  - Use this to create a 'trace' for comparisons
  - Comparing between Scalar, Simple and new 'Rolling' version



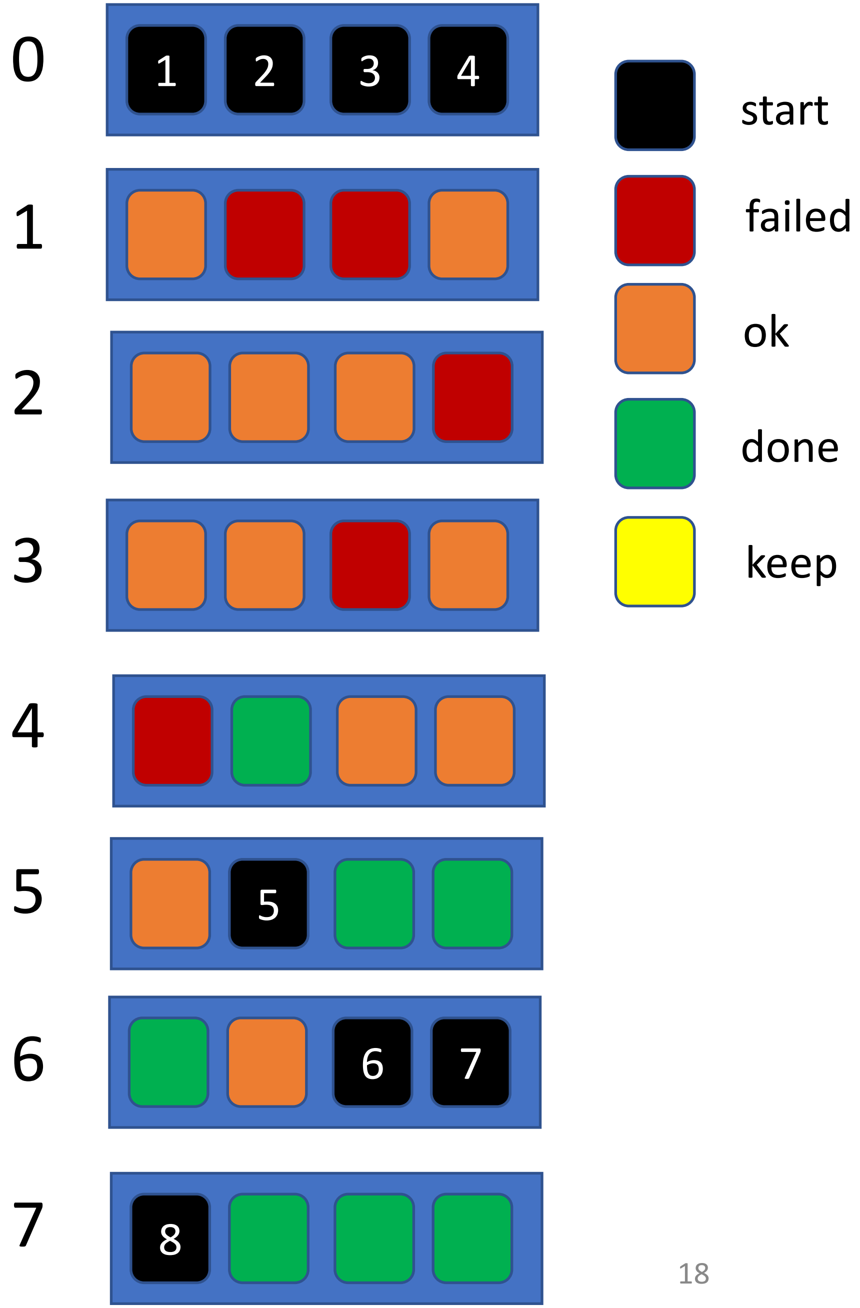
out-VecRollingDrv-printLane14-rs-n30d-outV7.1.log

```
KPS:  h-next    :               0 |       8872.1473 |      58551.3136 |      11.1390206 |
KPS: facStretch :               5 |               5 |               5 |     0.815130406 |
 KPS: x0+hdid   :            4000 |      12049.7768 |      31595.8868 |      2609.36765 |
KPS: xEnd (arg) :            2000 |           10000 |           30000 |          100000 |
 hdid= [2000, 10000, 30000, 1780.14462] and hnext= [0, 8872.1473, 58551.3136, 11.1390206]
 end of  KeepStepping method ..........................
##--------------------------------------------------------------------------------
### Accurate Advance ---- After return from KeepStepping
        charge :              13 |             -14 |              15 |             -16 |
  hTry (after) :               0 |      102.629501 |      8.85836995 |      5.90004855 |
         h-did :            2000 |           10000 |           30000 |      1780.14462 |
         hNext :               0 |       8872.1473 |      58551.3136 |      11.1390206 |
##--------------------------------------------------------------------------------
             x :            2000 |           10000 |           30000 |      1780.14462 |
 xExpectEnd/x2 :            2000 |           10000 |           30000 |          100000 |
##--------------------------------------------------------------------------------
 yStepStart[0] :      248.397552 |     -178.383559 |      94.9584981 |     -7.38670734 |
 yStepStart[1] :      -89.952651 |      -316.39924 |       14.809806 |     -3.54547822 |
 yStepStart[2] :      1680.47738 |      1627.85601 |      1349.46007 |        610.4522 |
 yStepStart[3] :     0.455046646 |    -0.149490266 |     0.331494782 |     0.687921026 |
 yStepStart[4] :    -0.536065199 |     -1.04047329 |    -0.867564044 |    -0.362512014 |
 yStepStart[5] :      1.22746777 |      0.94671915 |      1.06711213 |      1.18182202 |
##--------------------------------------------------------------------------------
       dydx[0] :     0.321677279 |   -0.0341729082 |  -0.00490046566 |     0.120954535 |
       dydx[1] :    -0.378950149 |    -0.646492528 |    -0.594547423 |     0.262890289 |
       dydx[2] :     0.867709925 |     0.762154593 |     0.804045613 |     0.957214029 |
       dydx[3] :  -0.000142264558 |             -0 |               0 | -0.000333316208 |
       dydx[4] :  0.000330838846 |              -0 |               0 |  0.000178290744 |
       dydx[5] :   0.00019722571 |              -0 |               0 | -6.84778771e-06 |
##--------------------------------------------------------------------------------
##--------------------------------------------------------------------------------
      yNext[0] :      248.397552 |     -458.524296 |     -20.9408812 |      6.71987528 |
      yNext[1] :      -89.952651 |     -5491.11401 |      -16959.018 |      16.7377565 |
      yNext[2] :      1680.47738 |      7654.14713 |      24108.0531 |      1544.21358 |
      yNext[3] :     0.455046646 |   -0.0483423852 |  -0.00693253438 |     0.173791671 |
      yNext[4] :    -0.536065199 |    -0.914554613 |    -0.841087508 |     0.371455295 |
      yNext[5] :      1.22746777 |      1.07817487 |        1.137458 |      1.35396096 |
##--------------------------------------------------------------------------------
       diff|p| :               0 | -3.45889357e-07 |   8.8067071e-07 |  1.38827705e-05 |
      d|p|/|p| :               0 | -2.44506809e-07 |  6.22528361e-07 |  9.81328092e-06 |
      |momEnd| :      1.41460612 |      1.41464065 |      1.41466849 |      1.41470593 |
##--------------------------------------------------------------------------------
        charge :              13 |             -14 |              15 |             -16 |
        Move-x :               0 |     -280.140737 |     -115.899379 |      14.1065826 |
        Move-y :               0 |     -5174.71477 |     -16973.8278 |      20.2832347 |
        Move-z :               0 |      6026.29112 |      22758.5931 |      933.761382 |
        Move-L :               0 |      7948.10269 |      28391.5097 |      934.088178 |
   Move-L/hdid :               0 |     0.794810269 |     0.946383656 |     0.524726008 |
 hRequest= [2000, 10000, 30000, 100000]
 hdid     = [2000, 10000, 30000, 1780.14462]
 x (Now) = [2000, 10000, 30000, 1780.14462]
 x2 -x    = [0, 0, 0, 98219.8554]
AccurateAdvance: hnext = [0, 0, 0, 11.1390206] to replace hTry = [0, 102.629501, 8.85836995, 5.90004855]
SID/AccAdv: hNext= :                0 |               0 |               0 |      11.1390206 |
 lastStep : m[1110]
Copying y <- yNext , as 1+ lanes continue.
```

```
KPS:  h-next    :               0 |       8872.1473 |      58551.3136 |      11.1390206 |
KPS: facStretch :               5 |               5 |               5 |     0.815130406 |
 KPS: x0+hdid   :            4000 |      12049.7768 |      31595.8868 |      2609.36765 |
KPS: xEnd (arg) :            2000 |           10000 |           30000 |          100000 |
 hdid= [2000, 10000, 30000, 1780.14462] and hnext= [0, 8872.1473, 58551.3136, 11.1390206]
 end of  KeepStepping method ..........................
##--------------------------------------------------------------------------------
### Accurate Advance ---- After return from KeepStepping
        charge :              13 |             -14 |              15 |             -16 |
  hTry (after) :               0 |      102.629501 |      8.85836995 |      5.90004855 |
         h-did :            2000 |           10000 |           30000 |      1780.14462 |
         hNext :               0 |       8872.1473 |      58551.3136 |      11.1390206 |
##--------------------------------------------------------------------------------
             x :            2000 |           10000 |           30000 |      1780.14462 |
 xExpectEnd/x2 :            2000 |           10000 |           30000 |          100000 |
##--------------------------------------------------------------------------------
 yStepStart[0] :      248.397552 |     -178.383559 |      94.9584981 |     -7.38670734 |
 yStepStart[1] :      -89.952651 |      -316.39924 |       14.809806 |     -3.54547822 |
 yStepStart[2] :      1680.47738 |      1627.85601 |      1349.46007 |        610.4522 |
 yStepStart[3] :     0.455046646 |    -0.149490266 |     0.331494782 |     0.687921026 |
 yStepStart[4] :    -0.536065199 |     -1.04047329 |    -0.867564044 |    -0.362512014 |
 yStepStart[5] :      1.22746777 |      0.94671915 |      1.06711213 |      1.18182202 |
##--------------------------------------------------------------------------------
       dydx[0] :     0.321677279 |   -0.0341729082 |  -0.00490046566 |     0.120954535 |
       dydx[1] :    -0.378950149 |    -0.646492528 |    -0.594547423 |     0.262890289 |
       dydx[2] :     0.867709925 |     0.762154593 |     0.804045613 |     0.957214029 |
       dydx[3] :  -0.000142264558 |             -0 |               0 | -0.000333316208 |
       dydx[4] :  0.000330838846 |              -0 |               0 |  0.000178290744 |
       dydx[5] :   0.00019722571 |              -0 |               0 | -6.84778771e-06 |
##--------------------------------------------------------------------------------
##--------------------------------------------------------------------------------
      yNext[0] :      248.397552 |     -458.524296 |     -20.9408812 |      5.06516067 |
      yNext[1] :      -89.952651 |     -5491.11401 |      -16959.018 |      13.1430032 |
      yNext[2] :      1680.47738 |      7654.14713 |      24108.0531 |       1531.1338 |
      yNext[3] :     0.455046646 |   -0.0483423852 |  -0.00693253438 |     0.171115121 |
      yNext[4] :    -0.536065199 |    -0.914554613 |    -0.841087508 |     0.371912503 |
      yNext[5] :      1.22746777 |      1.07817487 |        1.137458 |      1.35417655 |
##--------------------------------------------------------------------------------
       diff|p| :               0 | -3.45889357e-07 |   8.8067071e-07 |   1.4078631e-05 |
      d|p|/|p| :               0 | -2.44506809e-07 |  6.22528361e-07 |  9.95172839e-06 |
      |momEnd| :      1.41460612 |      1.41464065 |      1.41466849 |      1.41470613 |
##--------------------------------------------------------------------------------
        charge :              13 |             -14 |              15 |             -16 |
        Move-x :               0 |     -280.140737 |     -115.899379 |       12.451868 |
        Move-y :               0 |     -5174.71477 |     -16973.8278 |      16.6884814 |
        Move-z :               0 |      6026.29112 |      22758.5931 |      920.681603 |
        Move-L :               0 |      7948.10269 |      28391.5097 |      920.917026 |
   Move-L/hdid :               0 |     0.794810269 |     0.946383656 |     0.517327085 |
 hRequest= [2000, 10000, 30000, 100000]
 hdid     = [2000, 10000, 30000, 1780.14462]
 x (Now) = [2000, 10000, 30000, 1780.14462]
 x2 -x    = [0, 0, 0, 98219.8554]
AccurateAdvance: hnext = [0, 0, 0, 11.1390206] to replace hTry = [0, 102.629501, 8.85836995, 5.900
04855]
SID/AccAdv: hNext= :                0 |               0 |               0 |      11.1390206
 |
```

Updated status

- Working with Unit test 'testVectorIntegrationDriver.cxx'
  - Artificial value to 'stretch' code, test limits
- Several fixes in the last days
  - Now agreement for over 200 steps
  - Small disagreements in momentum
- Revisions
  - use 'final' momentum for relative error
- ToDos
  - Add per-track counter of steps (to enforce same maximum # steps as scalar.)



Next steps

- Finalise comparison with scalar case (in unit test)
- Run in GeantV tracking with CMS-like field
- Tune parameters and explore tradeoffs
    - Add ‘observables’ to monitor
    - vector loads vs full work
- Benchmark, profile/optimise with CMS-like semi-realistic fields
    - Identify cost of ‘loading’ vs extra iteration (for tuning)
- Later: ‘quick’ exploration of simple improvements
    - Reduce copying of data
    - More use of vector ‘Load’