Program Guide to Market 8.0

Ralph Abraham

January 9, 2007

This is the PDF of market080.03guide.tex, revised 09 January 2007. It is a program guide for market080.03.nlogo, our basic NetLogo model for a market of portfolio managers. In the first two sections, Managers and Steps, we follow the math notation conventions of the papers posted earlier on our website, http://www.vismath.org/research/landscapedyn/.

In the next two sections, Constants and Variables, we give a translation guide between the math notations (in *italics*) and those of the NetLogo code (in `typewriter style text`). One exception is this: we use u (in the program) in place of x (in the papers) to denote the risk strategy of managers, right from the start. Finally, we describe the main Procedures of the code.

This model is based on our paper, *Bubbles and Crashes: Escape Dynamics in Financial Markets*, of January, 2007, which we refer to as *B+C*. This model, 8.0, is a simplified starting point for a sequence of evolving models, all for our NSF grant research. See our website on Landscape Dynamics for the background.

We have a positive population of M portfolio managers, $Z(n), n = 1, ..., M$. The $n-th$ manager has a strategy $u(n)$ depending on time, t, which increments in timesteps of size, `stepsize`, and an investment portfolio of size $z(n)$. Usually we suppress the index, n, writing u in place of $u(n)$ where n may be inferred from context.

Sprites (*turtles* in NetLogo-speak) that represent these managers move about in the graphics window of NetLogo, with coordinates `(xcor, ycor)` indicating the managers variables, `(u, z)`. The graphics window is discretized in pixels (*patches* in NetLogo-speak) with coordinates `(pxcor, pycor)`.

1 Managers

There is a population of M managers (M = `totalpop`). Each manager has a portfolio comprising two types of assets, riskfree and risky. And each manager has two managers-own variables, in addition to NetLogo's usual `xcor, ycor, color, size`, and `heading`:

- u, current strategy choice, between 0 (all riskless) and *umax* (usually 4). Value 0 means all riskfree (such as bonds), 1 means all risky (such as stocks), and $u > 1$ means leveraged, that is, risk increased by borrowing.

- z, current portfolio worth, between `zmin` (usually 0) and `zmax` (settable by slider, default 4).

The managers-own variables `(u, z)` are mapped by an affine isomorphism into the managers-own coords `(xcor, ycor)` of the graphics window, where the managers are displayed as triangular turtles of random colors.

2 Simulation Steps

Here we explain the adjustment protocol, the stepsize, and the adjustment procedures for the two variables, u and z.

2.1 Adjustment protocol

The simulation proceeds in discrete time steps. Each step iterates a number, `u-steps`, chosen by a chooser, of substeps. With each substep, each manager adjusts her strategy u by trading risky for riskless assets or vice versa. This is done by an Euler step in the gradient dynamics of the net utility function, $\phi =$ `phi`,

$$\phi = R(u) = (R_1 - R_0)u - c_2u^2/2.$$

At the end of each step, each manager receives a payoff for her current holdings, and her portfolio worth, z, is adjusted accordingly, using the *gross payoff* function,

$$G = \phi_{gross} = (R_1 - R_0)u + R_0.$$

In principle, the times of these two adjustments need not coincide, but in this model, we assume both are done at the same regular timesteps. The equations for computing these two adjustments are given below.

2.2 Stepsize

In this model, time is measured in units of *years*. Rates of return or whatever are normally annualized. Timesteps may be chosen as $stepsize = 1/N$ where N is the number of timesteps per year. The model interface provides a chooser (drop-down menu) for:

$N = 1$ (annual)
$N = 4$ (quarterly)
$N = 12$ (monthly)
$N = 52$ (weekly)
$N = 365$ (daily)

There could be more steps per year in case of daytrading, but we will stop here for now. In addition, there is a smaller stepsize, `stepsize-u`, for the gradient dynamics. This is just `stepsize` divided by `u-steps`.

2.3 Adjustment of u

If the gradient vector is large, and the stepsize is also large, then numerical artifacts may be objectional. The most likely artifact is erratic large jumping of the managers. This situation may be improved by increasing the number, `u-steps`, of substeps in a step.

Then,

$$\texttt{u\_next} = \texttt{u\_now} + \texttt{stepsize\_u} * \texttt{phisubx}$$

where `phisubx` is our code for the gradient of ϕ. In this adjustment, the increment `u-jump = stepsize-u * phisubx` is clipped to a maximum jump (up or down) set by a slider, `max-u-jump`. Following this adjustment we further clip `u-next` to the width of the graphics window, `[umin, umax]`, and finally, `u-next` replaces `u-now`.

2.4 Adjustment of z, cutting down for stepsize

Fix one or the other of the two assets, and let its annual-yield be denoted by G (usually between -1.0 and $+1.0$ or so). This G could be greater than $+1.0$ (100 per cent) but may not be lower than -1.0 (which means wipeout) and we clip this at -0.999. Then with each of n steps per year there will be a step-yield, g, that compounds to an annual-yield of G, so,

$$(1+g)^N = 1+G$$

or equivalently,

$$g = (1+G)^{stepsize} - 1$$

We define a function (*reporter* in NetLogo-speak),

$$g = cut(G)$$

but apply clipping, so G cannot be less than -0.999. That is, if $stepsize = 1$,

$$asset\_value\_next\_year = asset\_value\_now * (1+G)$$

or rather, if $stepsize = 1/N$,

$$asset\_value\_next\_step = asset\_value\_now * (1+g)$$

where $g = cut(G)$, and if G does not change, then

$$(1+g)^N = 1+G$$

after N steps.

2.5 Step formulas for z adjustment

With each step of the simulation, each manager adjusts her u and z. The adjustment of u is determined by the gradient of ϕ, while for the adjustment of z, we use $G = \phi_{gross}$ itself, which depends on R_0, R_1, and u only. So let us begin with these. Recall,

$$G = \phi_{gross} = (R_1 - R_0)u + R_0 \tag{1}$$

$$R_1 = (R_s - g_s)\bar{u}^{-\alpha} + g_s + \alpha\dot{\bar{u}}/\bar{u} \tag{2}$$

$$(R_s - g_s) = dR \ (set\ by\ a\ slider),\ default\ 0.03 \tag{3}$$

$$\alpha = 2 \tag{4}$$

$$g_s \ (set\ by\ a\ slider),\ default\ 0.00 \tag{5}$$

$$R_0 \ (set\ by\ a\ slider),\ default\ 0.03 \tag{6}$$

Now the sensitivity of each manager to the choices of other managers is through the inclusion of the mean u, $\bar{u}$, and its rate of change, in the R_1.

Finally, the z adjustment is,

$$z(t + \Delta t) = z(t)[1 + cut(G)] \tag{7}$$

where $\Delta t = 1/n = stepsize$, and cut adjusts the annual gross yield to the stepsize by the formula shown above.

2.6 Step formulas for u adjustment

This is the gradient rule, so we just need the derivative of ϕ with respect to u. We ignore the derivative of $\bar{u}$ with respect to u, as we assume that the number of managers is large.

$$\phi_u = R_1 - R_0 - c_2 u \tag{8}$$

$$u(t + \Delta t) = u(t) + \phi_u \Delta t(\Delta u/\Delta t) \tag{9}$$

and we have set $(\Delta u/\Delta t) = 1$.

3 Constants

Global constants may be initialized with initial values, or by equations, or by sliders.

3.1 Global constants for model with initial values

$Math$,	`NetLogo`,.....	Initial,	Description .....
N,	`totalsteps`,	0,	increase by one with each "step" of many with "go"
T,	`totaltime`,	0,	totalsteps * stepsize
$\bar{x}$,	`mean-u`,	0.5,	defined by discrete integration over patches
$\bar{x}_{n-1}$,	`mean-u-old`,	0.5,	to hold previous value
$\dot{\bar{x}}$,	`mean-u-dot`,	0,	time derivative of $mean - u$

3.2 Global constants for equations with initial values

$Math$,	NetLogo,	Initial,	Description .....
d_2,	d2,	1.0,	current dividend per share
α,	alpha,	2,	exp in P,
x_{min},	umin,	0.0 ,	min u for managers, (0 means all riskless)
x_{max},	umax,	4.0,	max u
z_{min},	zmin,	0,	min portfolio size
z_{lim},	zlim,	0.1,	limit to size below which losses accrue

Note: c_2 is constant in model 6.0, but will vary in model 6.1.

3.3 Global constants and variables, set by an equation

$Math$,.....	NetLogo,.....	Description .....
$\cdots$,	width-z = zmax - zmin,	max range of z
b_2,	b2 = - alpha,	exponent in expression for phi
$\cdots$,	width-u = umax - umin,	total width of interval for manager variable u
$\cdots$,	delta-u = width-u / screen-size-x,	increment for patch dummy variable pu
x_{min},	xmin = (- screen-edge-x - 0.5),	min of xcor
x_{max},	xmax = (screen-edge-x + 0.5),	max of xcor
$\cdots$,	width-x = xmax - xmin,	same as screen-size-x but float, screen-size-x
y_{min},	ymin = - screen-edge-y - 0.5,	min of ycor, for conversion rule
y_{max},	ymax = screen-edge-y + 0.5,	max of ycor
$\cdots$,	width-y = ymax - ymin,	same as screen-size-y but float
$V = \frac{e^{g_s t}}{R_s - g_s}$,	V,	value of the risky asset
$P = V\bar{u}^{\alpha}$,	PM,	price of risky asset
R_s,	Rs = R0 + dR,	annual discount rate
$\bar{z}$,	mean-z,	mean after z is incremented, no normalization
R_1,	R1,	rate of return on risky investment

Note: $\cdots$ indicates nonexistence in the mathematical model.

3.4 Global constants, set by a slider

ΔM, population from 0 to 200 step 1, default 50 (total size of herd)
$\cdots$, center (of initial turtle herd, as percentage of screen width)
$\cdots$, width (of initial turtle herd, as percentage of screen width)
$\cdots$, altitude (vertical center of initial turtle herd, as percentage of screen height)
$\cdots$, height (vertical span of initial turtle herd, as percentage of screen height)
Δt, stepsize (for euler method)

$\cdots$, climbrate (for euler method)
z_{max}, zmax, max portfolio size
c_2, c2, coefficient of quadratic term in risk premium function, risk cost
$\cdots$, max-u-jump, maximum increment in u
R_0, R0, rate of return of riskless asset
g_s, gs, annualized growth rate for the risky asset

3.5 Global constants, set by interface

totalpop, total number of managers
frequency, number of steps in a year
u-steps, number of substeps in a step

4 Variables

We have patch variables (for patches only) and turtle variables (for managers only).

4.1 Patch variables

$\cdots$, patch-u, varies from umin to umax as pxcor from -(screen-size-x) to +(screen-size-x)
$\cdots$, patch-z, similar
$\cdots$, Fraw, raw distribution (number of turtles per patch)
$\cdots$, F, normed Fraw, F/totalpop
ϕ, patch-phi, the landscape, from B+C
ϕ_x, patch-phisubx, gradient of phi

4.2 Turtle variables for managers

z, z, portfolio size, display this as positive ycor (scaled)
u, u, choice of riskiness, this is x in *B+C*.

5 Procedures

These are the primary procedures from the NetLogo code.

5.1 To step

This is the primary procedure in the model. We need do-math for everything:

```
to do-inner-math    ;;; three computes needed for substeps
  compute-mean-u    ;;; needed for phi and slope, average patch-u on row 0
  compute-mean-u-dot  ;;; ditto
  compute-R1
end

to do-math          ;;; compute all funcs
  count-managers    ;;; count all managers, record Fraw on row 0
  do-inner-math     ;;;compute mean-u. mean-u-dot, and R1
  compute-stepsize  ;;; in case frequency or u-steps has been changed
  compute-slope     ;;; compute phisubx on row 0
  color-slope       ;;; color pycor = -6 row red, yellow, green acc to slope value
  compute-patch-phi       ;;; not used to adjust z
  compute-PM        ;;; for display only, market price ticker-tape
  compute-DPM       ;;; also for display as ticker-tape
  compute-mean-z    ;;; so new value will be reported in the monitor
  set Rs ( R0 + dR )   ;;; in case sliders have been changed
end
```

The most important procedure is step.

```
to step
  compute-stepsize ;;; if choosers changed
  do-math          ;;; make sure all up-to-date (substep only does do-inner-math)
  foreach n-values u-steps [?] ;;; exm: list = 0,..., u-steps-1
    [
    substep ;;; of size substep-u
    ]
  ;;; managers receive pay from prior value of phi
  update-managers-z ;;; includes clipping
  adjust-mean-queen
  adjust-yellow-king
  ;;; update globals and plots
  set totalsteps (totalsteps + 1)
  set totaltime (totaltime + stepsize)
  if totalsteps mod 10 = 0   ; every 10th time step, redraw the two graphs
    [
    do-sprint-1
    do-sprint-2
    ]
  do-plot-3
  track-managers
```

```
end

to substep ;;; move one step up slope (all managers)
;;; 1. managers decide to move their strategy u, thus also xcor
  move-managers-u
  update-managers-xcor
;;; 3. do-math (compute mean-u, R1, slope, and phi)
;;;    to prepare for the next substep
  ;;; do-math
  set localtime ( localtime + stepsize-u )
end
```

Note: we could equally have put substep (3) before (1) and (2).

Part 1 is accomplished with this action taken by all managers:

```
to move-managers-u ;;; move one substep up slope (all managers)
  ask managers [ ;; update u and clip if necc
    ;;;;; recompute slope, each manager
    do-inner-math
    ;;; compute-mean-u    ;;; needed for phi and slope, average patch-u on row 0
    ;;; compute-mean-u-dot  ;;; ditto
    ;;; compute-R1
    let phisubx ( R1 - R0 - c2 * u )
    let u-jump ( stepsize-u * phisubx )
    ;;; now clipping
    if (u-jump > max-u-jump) [ set u-jump max-u-jump ] ;;; clip u-jump right
    if (u-jump < ( - max-u-jump)) [ set u-jump ( - max-u-jump ) ] ;;; clip u-jump left
    ;;; now move u
    set u ( u + u-jump )
    ;;; more clipping
    if (u < umin ) [ set u umin ] ;;; clipping left edge
    if (u > umax ) [ set u umax ] ;;; clipping right edge
  ]
end
```

Part 2 is accomplished with this action taken by all managers:

```
to update-managers-z ;;; update turtle variable z and move its ycor
  ask managers [
    ;;; let annual-pay ( u * ( R1 - R0 ) - c2 * u * u / 2 ) ;;; this is net return
    let annual-gross ( u * ( R1 - R0 ) + R0 ) ;;; this is gross return
```

```
    let pay cut annual-gross ;;; this is the annual yield reduced to interval of stepsize
    set z ( z * ( 1 + pay ) ) ;;; increment size of portfolio for immediate timestep
    if (z < zlim) [ set z zlim ]
    if (z > zmax) [ set z zmax ]
    let ytemp ( z - zmin ) * ( width-y - 1 ) / ( 2 * width-z ) ;;; convert z to ycor
    set ycor ytemp ;;; keep as float
  ]
end
```

Substep 2 also implements the cluster point, u*, described in Proposition 1 of B+C. The yellow king is moved to a position xcor corresponding to u^*. From B+C we have the definition of x^* as a positive root of the equation,

$$R_1 - R_0 = (R_s - g_s)\bar{x}^{-\alpha} + g_s - R_0 - c_2 x = 0$$

Recall x of B+C is u and $u^* = \bar{u}$ here, so,

$$= (R_s - g_s)\bar{u}^{-2} + g_s - R_o - c_2 u^* = 0$$

We now approximate the solution numerically, using Newton's method. Here is the code.

```
;;; to clip number outside of interval (-0.1, +0.1)
to-report clip-away [ number ]
    let tempmag abs number
    let tempsign 1
    let tempresult 0
    if not ( tempmag = 0 ) [ set tempsign ( number / tempmag ) ]
    ifelse ( tempmag < 0.1 ) [ set tempresult ( tempsign * 0.1 ) ]
      [set tempresult number ]
    report tempresult
end

to adjust-yellow-king ;;; move him to crit position, new approach
  ask turtle 2 [
    let a ( Rs - gs1 ) ;;; coeff of u ^ (-alpha) term in func, aka dR
    let b ( gs1 - R0 ) ;;; constant term in func
    let u0 u ;;; to begin newton method
    let func0 a / ( u0 ^ 2 ) + b  - ( c2 * u0 )
    let funcdot0 -2 * a / ( u0 ^ 3 ) - c2
    set funcdot0 clip-away funcdot0 ;;; not to divide by zero
    let u1 u0 - func0 / funcdot0 ;;; first step of newton
    set u1 clip-away u1 ;;; not to divide by zero
```

```
    let func1 a / ( u1 ^ 2 ) + b  - ( c2 * u1 )
    let funcdot1 -2 * a / ( u1 ^ 3 ) - c2
    set funcdot1 clip-away funcdot1 ;;; not to divide by zero
    let u2 u1 - func1 / funcdot1 ;;; 2nd step of newton
    ;;; clip u2
    if ( u2 > umax ) [ set u2 umax ]
    if ( u2 <  umin ) [ set u2 umin ]
    set u u2  ;;; record current u2 as u0 for next step
    ;;; now move the king
    set xcor ( (u2 - umin) * width-x / width-u + xmin )
    set ycor -1
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ]
end
```

All this is summarized in the following diagrams.

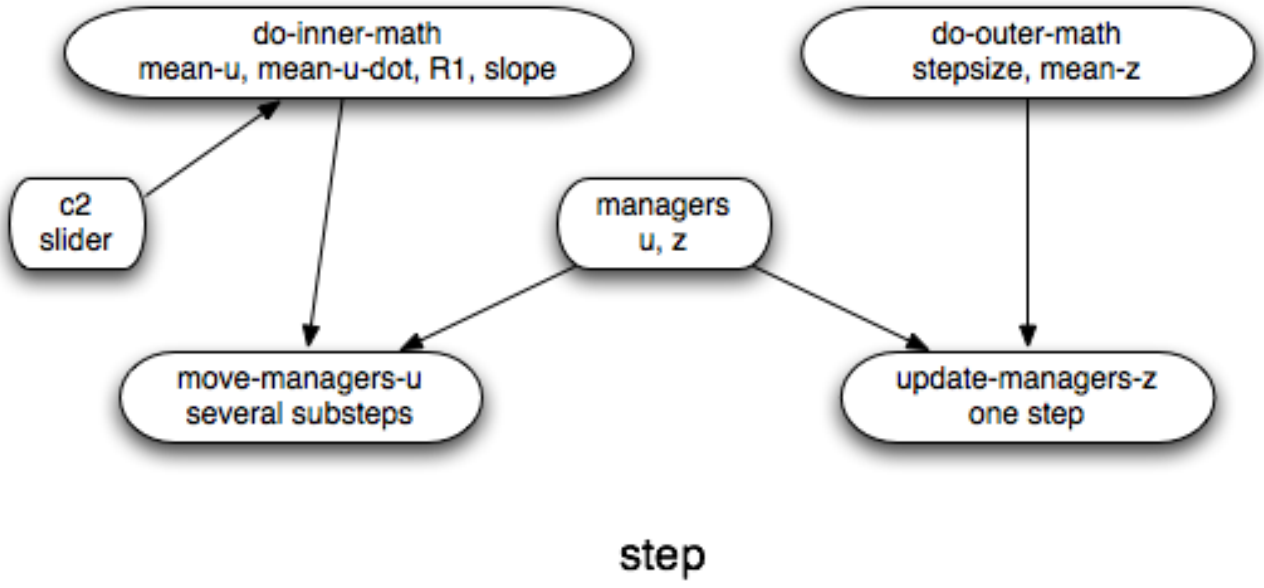

Figure 1: The Big Picture: Outline of the Step Procedure.

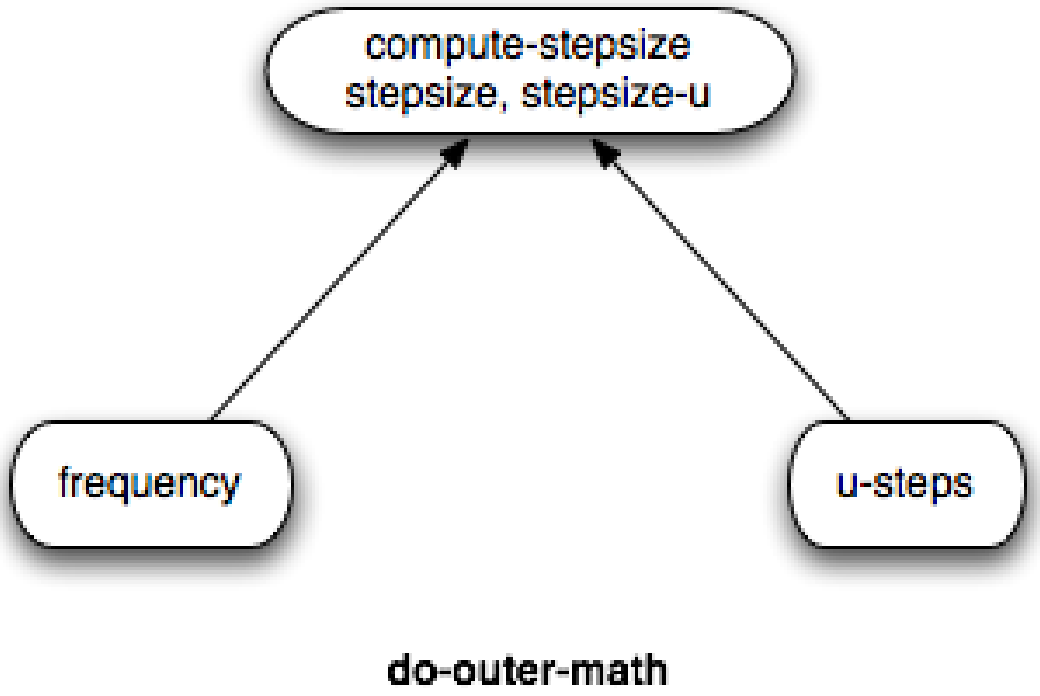

Figure 2: The "do-math" procedures: outer loop.

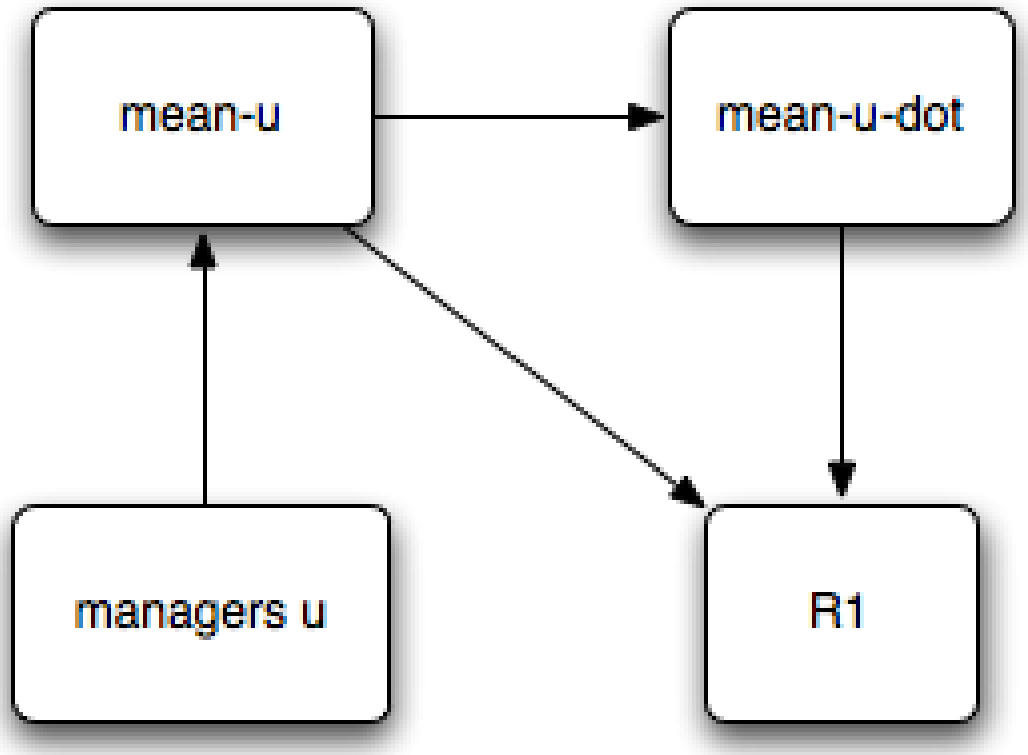

Figure 3: The "do-math" procedures: inner loop.

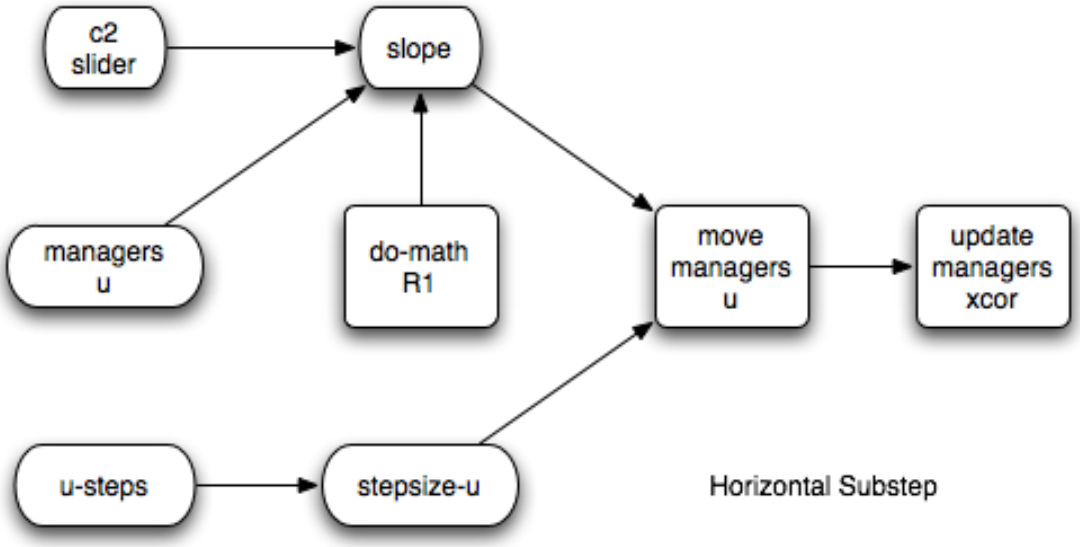

Figure 4: Outline of the horizontal substep procedure.

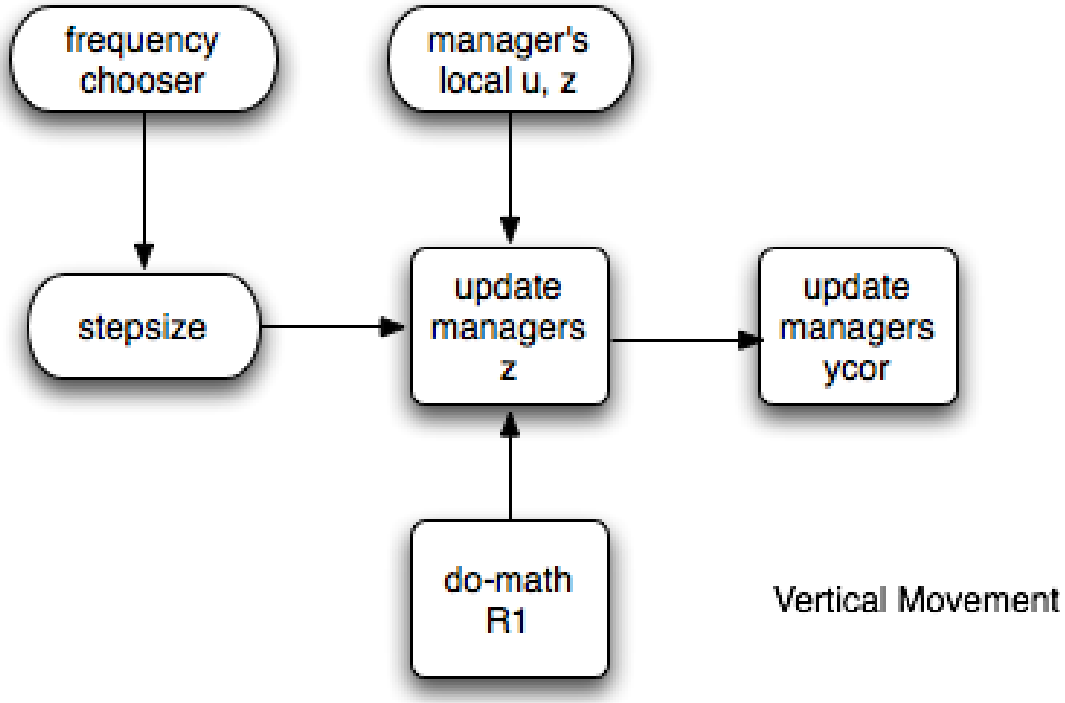

Figure 5: Outline of the vertical step procedure.