The Particle Filter

David N. DeJong
University of Pittsburgh

Spring 2008

Introduction

The **Particle Filter** was developed by Gordon, Salmond and Smith (1993, *IEEE Proceedings F*) and Kitagawa (1996, *J. of Computational and Graphical Statistics*) as a means of achieving filtering and likelihood analysis in state-space representations featuring departures from linearity and/or normality.

Reboot State-Space Reps

State-transition equation:

$$s_t = \gamma(s_{t-1}, Y_{t-1}, \upsilon_t)$$

Associated density:

$$f(s_t | s_{t-1}, Y_{t-1})$$

Measurement equation:

$$y_t = \delta(s_t, Y_{t-1}, u_t)$$

Associated density:

$$f(y_t | s_t, Y_{t-1})$$

Initialization:

$$f(s_0)$$

Reboot State-Space Reps

- **Filtering objective:** construct $f\left(s_t \middle| Y_t\right)$, which can then be used to approximate $E_t\left(h(s_t) \middle| Y_t\right)$.

Reboot State-Space Reps

- **Filtering objective:** construct $f\left(s_t \middle| Y_t\right)$, which can then be used to approximate $E_t\left(h(s_t) \middle| Y_t\right)$.

- **Likelihood evaluation** obtains as a by-product of the filtering process.

Reboot State-Space Reps

- From Bayes' theorem, $f\left(s_t | Y_t\right)$ is given by

$$f\left(s_t | Y_t\right) = \frac{f\left(y_t, s_t | Y_{t-1}\right)}{f\left(y_t | Y_{t-1}\right)} = \frac{f\left(y_t | s_t, Y_{t-1}\right) f\left(s_t | Y_{t-1}\right)}{f\left(y_t | Y_{t-1}\right)},$$

Reboot State-Space Reps

- From Bayes' theorem, $f\left(s_t \middle| Y_t\right)$ is given by

$$f\left(s_t \middle| Y_t\right) = \frac{f\left(y_t, s_t \middle| Y_{t-1}\right)}{f\left(y_t \middle| Y_{t-1}\right)} = \frac{f\left(y_t \middle| s_t, Y_{t-1}\right) f\left(s_t \middle| Y_{t-1}\right)}{f\left(y_t \middle| Y_{t-1}\right)},$$

- where $f\left(s_t \middle| Y_{t-1}\right)$ is given by

$$f\left(s_t \middle| Y_{t-1}\right) = \int f\left(s_t \middle| s_{t-1}, Y_{t-1}\right) f\left(s_{t-1} \middle| Y_{t-1}\right) ds_{t-1},$$

Reboot State-Space Reps

- From Bayes' theorem, $f\left(s_t | Y_t\right)$ is given by

$$f\left(s_t | Y_t\right) = \frac{f\left(y_t, s_t | Y_{t-1}\right)}{f\left(y_t | Y_{t-1}\right)} = \frac{f\left(y_t | s_t, Y_{t-1}\right) f\left(s_t | Y_{t-1}\right)}{f\left(y_t | Y_{t-1}\right)},$$

- where $f\left(s_t | Y_{t-1}\right)$ is given by

$$f\left(s_t | Y_{t-1}\right) = \int f\left(s_t | s_{t-1}, Y_{t-1}\right) f\left(s_{t-1} | Y_{t-1}\right) ds_{t-1},$$

- and $f\left(y_t | Y_{t-1}\right)$ is given by

$$f\left(y_t | Y_{t-1}\right) = \int f\left(y_t | s_t, Y_{t-1}\right) f\left(s_t | Y_{t-1}\right) ds_t.$$

Reboot State-Space Reps

Taking $f\left(s_{t-1}\middle|Y_{t-1}\right)$ as given, initialized with $f\left(s_0\middle|Y_0\right) \equiv f(s_0)$, filtering and likelihood evaluation proceed recursively:

- Prediction: $f\left(s_{t-1}\middle|Y_{t-1}\right)$ combines with $f\left(s_t\middle|s_{t-1}, Y_{t-1}\right)$ to yield

$$f(s_t|Y_{t-1}) = \int f\left(s_t\middle|s_{t-1}, Y_{t-1}\right) f\left(s_{t-1}\middle|Y_{t-1}\right) ds_{t-1} \longrightarrow (4)$$

Reboot State-Space Reps

Taking $f\left(s_{t-1}|Y_{t-1}\right)$ as given, initialized with $f\left(s_0|Y_0\right) \equiv f(s_0)$, filtering and likelihood evaluation proceed recursively:

- Prediction: $f\left(s_{t-1}|Y_{t-1}\right)$ combines with $f\left(s_t|s_{t-1},Y_{t-1}\right)$ to yield

$$f(s_t|Y_{t-1}) = \int f\left(s_t|s_{t-1},Y_{t-1}\right) f\left(s_{t-1}|Y_{t-1}\right) ds_{t-1} \longrightarrow (4)$$

- Forecasting: $f(s_t|Y_{t-1})$ combines with $f\left(y_t|s_t,Y_{t-1}\right)$ to yield

$$f\left(y_t|Y_{t-1}\right) = \int f\left(y_t|s_t,Y_{t-1}\right) f\left(s_t|Y_{t-1}\right) ds_t. \longrightarrow (5)$$

Reboot State-Space Reps

Taking $f\left(s_{t-1}|Y_{t-1}\right)$ as given, initialized with $f\left(s_0|Y_0\right) \equiv f(s_0)$, filtering and likelihood evaluation proceed recursively:

- Prediction: $f\left(s_{t-1}|Y_{t-1}\right)$ combines with $f\left(s_t|s_{t-1}, Y_{t-1}\right)$ to yield

$$f(s_t|Y_{t-1}) = \int f\left(s_t|s_{t-1}, Y_{t-1}\right) f\left(s_{t-1}|Y_{t-1}\right) ds_{t-1} \longrightarrow (4)$$

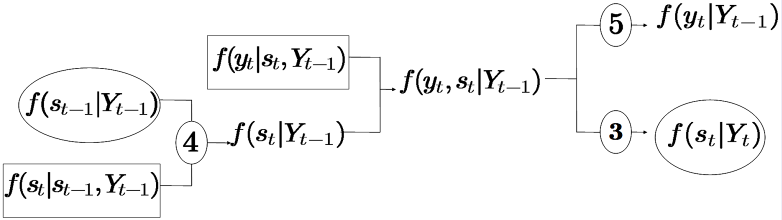
- Forecasting: $f(s_t|Y_{t-1})$ combines with $f\left(y_t|s_t, Y_{t-1}\right)$ to yield

$$f\left(y_t|Y_{t-1}\right) = \int f\left(y_t|s_t, Y_{t-1}\right) f\left(s_t|Y_{t-1}\right) ds_t. \longrightarrow (5)$$

- Updating: Bayes' Rule yields

$$f(s_t|Y_t) = \frac{f\left(y_t|s_t, Y_{t-1}\right) f\left(s_t|Y_{t-1}\right)}{f\left(y_t|Y_{t-1}\right)} \longrightarrow (3)$$

Reboot State-Space Reps



Notation and Terminology

- **Particle:** $s_t^{r,i}$ denotes the i^{th} draw of s_t obtained from the conditional density $f\left(s_t \middle| Y_{t-r}\right)$ for $r = 0, 1$.

Notation and Terminology

- **Particle:** $s_t^{r,i}$ denotes the i^{th} draw of s_t obtained from the conditional density $f\left(s_t \middle| Y_{t-r}\right)$ for $r = 0, 1$.
- **Particle Swarm:** $\{s_t^{r,i}\}_{i=1}^{N}$

Notation and Terminology

- **Particle:** $s_t^{r,i}$ denotes the i^{th} draw of s_t obtained from the conditional density $f\left(s_t \middle| Y_{t-r}\right)$ for $r = 0, 1$.
- **Particle Swarm:** $\{s_t^{r,i}\}_{i=1}^N$
- **Objective of Filtration:** transform a swarm $\{s_{t-1}^{0,i}\}_{i=1}^N$ to $\{s_t^{0,i}\}_{i=1}^N$

Notation and Terminology

- **Particle:** $s_t^{r,i}$ denotes the i^{th} draw of s_t obtained from the conditional density $f\left(s_t \middle| Y_{t-r}\right)$ for $r = 0, 1$.
- **Particle Swarm:** $\{s_t^{r,i}\}_{i=1}^N$
- **Objective of Filtration:** transform a swarm $\{s_{t-1}^{0,i}\}_{i=1}^N$ to $\{s_t^{0,i}\}_{i=1}^N$
- **Initialization of the filter:** $\{s_0^{0,i}\}_{i=1}^N$ drawn from $f\left(s_0 \middle| Y_0\right) \equiv f\left(s_0\right)$.

Period-t Filtration and Likelihood Evaluation

Period-t filtration and likelihood evaluation takes as input a swarm $\{s_{t-1}^{0,i}\}_{i=1}^{N}$. It consists of three steps.

- **Predictive step:** for each particle $s_{t-1}^{0,i}$, obtain a drawing $s_t^{1,i}$ from the conditional density $f\left(s_t \middle| s_{t-1}^{0,i}, Y_{t-1}\right)$.

Period-t Filtration and Likelihood Evaluation

Period-t filtration and likelihood evaluation takes as input a swarm $\{s_{t-1}^{0,i}\}_{i=1}^{N}$. It consists of three steps.

- **Predictive step:** for each particle $s_{t-1}^{0,i}$, obtain a drawing $s_t^{1,i}$ from the conditional density $f\left(s_t | s_{t-1}^{0,i}, Y_{t-1}\right)$.
- **Likelihood evaluation:** having obtained the swarm $\{s_t^{1,i}\}_{i=1}^{N}$, the MC estimate of the time-t likelihood $f\left(y_t | Y_{t-1}\right)$ is given by

$$\widehat{f}_N(y_t | Y_{t-1}) = \frac{1}{N} \sum_{i=1}^{N} f(y_t | s_t^{1,i}, Y_{t-1}).$$

This can be seen in light of (5):

$$f\left(y_t | Y_{t-1}\right) = \int f\left(y_t | s_t, Y_{t-1}\right) f\left(s_t | Y_{t-1}\right) ds_t.$$

Period-t Filtration and Likelihood Evaluation, cont.

- **Updating Step:** Updating involves the construction of an approximation to $f\left(s_t \middle| Y_t\right)$, which is achieved by re-weighting $\{s_t^{1,i}\}_{i=1}^N$ in accordance with

$$f\left(s_t \middle| Y_t\right) = \frac{f\left(y_t \middle| s_t, Y_{t-1}\right) f\left(s_t \middle| Y_{t-1}\right)}{f\left(y_t \middle| Y_{t-1}\right)}.$$

Since each particle $s_t^{1,i}$ represents a drawing from $f\left(s_t \middle| Y_{t-1}\right)$, its associated weight under $f\left(s_t \middle| Y_t\right)$ is given by

$$w_t^{0,i} = \frac{f\left(y_t \middle| s_t^{1,i}, Y_{t-1}\right)}{\widehat{f}_N\left(y_t \middle| Y_{t-1}\right)}.$$

Therefore, $\{s_t^{0,i}\}_{i=1}^N$ (the approximation to $f\left(s_t \middle| Y_t\right)$ we seek) is obtained by drawing with replacement from the swarm $\{s_t^{1,i}\}_{i=1}^N$ with probabilities $\{w_t^{0,i}\}_{i=1}^N$ (i.e., bootstrapping).

Example: Optimal Growth Model

State Transition Equations:

$$\begin{aligned}\left(1+\frac{g}{1-\alpha}\right)k'(\widetilde{k}_t,\widetilde{z}_t) &= i(\widetilde{k}_t,\widetilde{z}_t)+(1-\delta)k_t \\ \log z_t &= (1-\rho)\log(z_0)+\rho\log z_{t-1}+\varepsilon_t .\end{aligned}$$

Observation Equations:

$$\begin{aligned}X_t &= H'x_t+u_t, \qquad u_t\sim N(0,\Sigma_u), \\ H &= \begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 1\\ 0 & 0\\ 0 & 0\end{bmatrix}, \\ X_t &= \begin{pmatrix}\widehat{y}_t & \widehat{i}_t\end{pmatrix}',\end{aligned}$$

Σ_u diagonal.

Example, cont.

Algorithm for achieving likelihood evaluation
Step 1: For candidate model parameterization θ, obtain policy function $k'(\widetilde{k}_t, \widetilde{z}_t)$ using projection method.
Step 2 (Initialization): Obtain $\{s_0^{0,i}\}_{i=1}^N$ from the unconditional distribution $f(s_0)$, which is approximated using a log-linear model approximation.
Step 3 (Prediction): With $\{s_{t-1}^{0,i}\}_{i=1}^N$ now given, obtain $\{s_t^{1,i}\}_{i=1}^N$ from the conditional density $f\left(s_t | s_{t-1}^{0,i}, Y_{t-1}\right)$. For each particle $s_t^{1,i}$, obtain corresponding predictions of the observables $x_t^{1,i} = \left(\ln\left(\frac{y_t^{1,i}}{y*}\right) \quad \ln\left(\frac{i_t^{1,i}}{i*}\right)\right)'$.

Example, cont.

Step 4 (Likelihood Evaluation): The time-t value of the likelihood function is given by

$$f\left(y_t | s_t^{1,i}, Y_{t-1}\right) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_u|}} \exp\left(-\frac{\left(X_t - x_t^{1,i}\right)' \Sigma_u \left(X_t - x_t^{1,i}\right)}{2}\right).$$

Averaging over particles yields the likelihood estimate

$$\widehat{f}_N\left(y_t | Y_{t-1}\right) = \frac{1}{N}\sum_{i=1}^{N} f\left(y_t | s_t^{1,i}, Y_{t-1}\right).$$

The weight associated with a given particle is given by

$$w_t^{0,i} = \frac{f\left(y_t | s_t^{1,i}, Y_{t-1}\right)}{\widehat{f}_N\left(y_t | Y_{t-1}\right)}.$$

Example, cont.

Step 5 (Updating): Obtain $\{s_t^{0,i}\}_{i=1}^N$ (the approximation to $f\left(s_t \middle| Y_t\right)$ we seek) by drawing with replacement from $\{s_t^{1,i}\}_{i=1}^N$ with probabilities $\{w_t^{0,i}\}_{i=1}^N$ (i.e., bootstrapping).

With $\{s_t^{0,i}\}_{i=1}^N$ in hand, return to Step 3 and repeat until the end of the sample has been reached.

Example, cont.

Code. There are three main procedures that execute these steps.

- `qfct(praw)`: takes constrained parameters `praw`, maps to `p` via a logistic tansformation, establishes integrating constant for the likelihood function, removes means from the data, solves the model, and calls `partproc`
- `partproc`: executes Steps 2, 3, and 5 above, calls `lkeval` to perform Step 4.
- `lkeval`: performs Step 4.

Example, cont.

```
proc qfct(praw);
     // likelihood evaluation procedure
 local r,wmat,lnnormcons,relss,dmyi,lnlkwght;
      p = transform(praw);
      gss = 1+p[7]/(1-p[1]);
      xsi = gss^(-p[4]);
      r = zeros(2,2); // Euu' = r
      r[1,1] = p[8]^2;
      r[2,2] = p[9]^2;
      wmat = invpd(r);
      lnnormcons = ln( ((2*pi)^(-nobvars/2))*( (det(wmat))^0.5
) );
      xbar = ln(steady(p));
      ss = exp(xbar);
      onemrhoz = (1-p[5])*ln(ss[5]);
      relss = ss./ss[1];
      dmyi = yi - ln(relss[1 3]');
```

Example, cont.

```
    {TC,T1,T0,RC}=modelsol(p); // log-linear model approximation
    f = t1;
    vcvmat = sigmat(p); // vcv matrix of upsilon = [eps]
    q = (t0*vcvmat)*(t0'); // vcv matrix of e(t) = T0upsilon(t)
    vcvx = inv(eye(nvars^2)-f.*.f)*vec(q);
    vcvx = reshape(vcvx,nvars,nvars);
    stdx = sqrt(diag(vcvx));
    vcvstilde = vcvx[4 5,4 5];
// vcv matrix of ktilde, ztilde. Used in the particle filter to obtain unconditional draws of k, z.
    cvcvstilde = chol(vcvstilde)';
// Initialization for Chebyshev polynomial approximation here (suppressed).
    { gamopt,fopt,gopt,retcode } = nlsys(&feval,startval);
// Non-linear model approximation obtained.
    lnlkwght = partproc(dmyi,wmat,lnnormcons);

  retp(-lnlkwght);
endp;
```

Example, cont.

```
proc partproc(rawdat,wmat,lnnormcons);
 // Executes the particle filter. Returns likelihood function for the entire sample.
 // Inputs: raw data, information matrix for likelihood function, logged normalizing constant for
likelihood function
 // Output: log likelihood value for the entire sample
     local blahblahblah;
     newz = zeros(1,nparts);        // particles
     newk = zeros(1,nparts);
     lk = zeros(1,nparts);        // likelihood values associated with particles
     lts = zeros(nobs,1);    // date-t likelihood values (averaged over particles)
     lnlts = zeros(nobs,1);        // date-t log likelihood values
     rsnewz = zeros(1,nparts); // resampled particles (smoothed z's and k's) passed to
subsequent time period
     rsnewk = zeros(1,nparts);
```

Example, cont.

```
jjj = 1; do while jjj<=nobs;          // index over dates
 if jjj == 1;                         // draw initial state
      s0tildedraw = cvcvstilde*crn_rndn0;
     rsnewz = ss[5]*(1+s0tildedraw[2,.]);
      rsnewk = ss[4]*(1+s0tildedraw[1,.]);
endif;
iii = 1; do while iii<=nparts; // index over particles
     { newz[1,iii],newk[1,iii],lk[1,iii] } =
lkeval(rawdat[jjj,.]',crn_rndn1[jjj,iii]*p[6],rsnewz[1,iii]|rsnewk[1,iii],wmat,lnnormcons
// newz and newk are updated values: z' and k'
iii = iii+1; endo;
```

Example, cont.

```
lk = normal(lk);        // eliminates "Not a number" problem
probs = lk./(sumc(lk'));
csprobs = cumsumc(probs');
lts[jjj] = meanc(lk');
lnlts[jjj] = ln(lts[jjj])-lnlkadj;
// the term -lnlkadj eliminates the adjustement term employed by lkeval to prevent underflows
```

Example, cont.

```
    // resampling
    ix = 1;
    for iii (1,nparts,1);
    do while (udraw[iii,jjj] >= csprobs[ix]);
// udraw is sorted from smallest to largest uniform draws
            ix = ix + 1;
     endo;
                   rsnewz[1,iii] = newz[1,ix];
                   rsnewk[1,iii] = newk[1,ix];
     endfor;
```

Example, cont.

```
jjj = jjj+1; endo;
lnlk = sumc(lnlts);
retp(lnlk);
```

Example, cont.

```
proc (3) = lkeval(datt,veet,s_1,wmat,lnnormcons);
/* inputs: time-t data (y,i), time-t structural shocks (eps), time-(t-1) state (z,k)
information matrix of observation errors, logged normalizing constant */
/* output: zt, kt, uyt, uit, lkval (adjusted to prevent overflows/underflows) */
local blahblahblah;
    z_1 = s_1[1];
    k_1 = s_1[2];
    eps = veet;
    y = datt[1];
    i = datt[2];
```

Example, cont.

```
// convert yesterday's a and k into today's a and k
ktilde = (k_1 - ss[4])/omegak;
ztilde = (z_1 - ss[5])/omegaz;
yci = yci_of_kz(ktilde|ztilde,gamopt);
y_1 = yci[1];
c_1 = yci[2];
i_1 = yci[3];
k = (i_1+(1-p[3])*k_1)/gss;
lnz = onemrhoz + p[5]*ln(z_1) + eps;
z = exp(lnz);
```

Example, cont.

```
 /* construct predictions for todays y, c, i */
  ktilde = (k - ss[4])/omegak;
  ztilde = (z - ss[5])/omegaz;
 ycipred = yci_of_kz(ktilde|ztilde,gamopt);
  ypred = ycipred[1];
  cpred= ycipred[2];
ipred = ycipred[3];
```

Example, cont.

```
// construct implied errors
        uy = y - ln(ypred/ss[1]);
        ui = i - ln(ipred/ss[3]);
        u = uy|ui;
        lnkern = -0.5*(u'wmat*u);
        lnlkvaladj = lnnormcons+lnkern+lnlkadj;
// an adjustement term (lnlkadj) is added to prevent underflow; eliminated below
        lkvaladj = exp(lnlkvaladj);
        retp(z,k,lkvaladj);
endp;
```

Example, cont.

Now the Bad News

- Likelihood evaluation is expensive. Using artificial data generated from the example model, with T=239, N=60,000, a single evaluation of the likelihood function requires approximately 10 minitues of CPU time. (Fortran is much faster.)
- But: Speed can be greatly enhanced by avoiding the loop over particles (by a factor of roughly 6 in the present case). Reference: `yci_of_kz_swarm(s,gam)` (details below).

Example, cont.

- Numerical inaccuracy. Holding parameters fixed, and allowing random numbers to vary, log-likelihood values yielded by the particle filter:

1156.8176	1157.4325
1159.2276	1158.3206
1158.2024	1160.2855
1159.3893	1157.5191
1158.2917	1158.6697

Mean: 1158.4156, Std. Dev.: 1.03

Understanding Numerical Inefficiency

As the particle filter enters the **prediction** stage, the discrete approximation

$$f\left(s_{t-1} \middle| Y_{t-1}\right) \approx \{s_{t-1}^{0,i}\}_{i=1}^{N}$$

is set. To facilitate prediction, each particle $s_{t-1}^{0,i}$ is combined with $f\left(s_t \middle| s_{t-1}, Y_{t-1}\right)$ to generate the predictive swarm $\left\{s_t^{1,i}\right\}_{i=1}^{N}$.

By ignoring information contained in y_t in producing $\left\{s_t^{1,i}\right\}_{i=1}^{N}$, the particle filter is said to have produce ‘blind draws’ (Pitt and Shephard, 1999 *JASA*).

Numerical Inefficiency, cont.

The attainment of blind draws is a problem when the **measurement density**

$$f\left(y_t \middle| s_t, Y_{t-1}\right),$$

viewed as a function of y_t, is sharply peaked, and/or lies in the tails of $f\left(s_t \middle| Y_{t-1}\right)$.

In this case, $\left\{s_t^{1,i}\right\}_{i=1}^{N}$ will contain relatively few elements in the relevant range of $f\left(y_t \middle| s_t, Y_{t-1}\right)$.

This gives rise to a problem known as **sample impoverishment.**

Numerical Inefficiency, cont.

Why sample impoverishment? Recall that in the filtering stage, $\left\{s_t^{1,i}\right\}_{i=1}^N$ is converted to $\left\{s_t^{0,i}\right\}_{i=1}^N$ by sampling with replacement from $\left\{s_t^{1,i}\right\}_{i=1}^N$, with resampling probabilities given by the weights

$$w_t^{0,i} = \frac{f\left(y_t|s_t^{1,i}, Y_{t-1}\right)}{\widehat{f}_N\left(y_t|Y_{t-1}\right)}.$$

Thus if only a small portion of the particles $s_t^{1,i}$ are likely in light of $f\left(y_t|s_t, Y_{t-1}\right)$:

- only a small portion of those particles will be resampled, reducing the effective size of the swarm;
- and, the resampled swarm $\left\{s_t^{0,i}\right\}_{i=1}^N$ is likely to provide a poor approximation of $f\left(s_t|Y_t\right)$.

Avoiding the Loop Over Particles

GAUSS is notoriously slow in handling loops. When speed matters, loops are to be avoided whenever possible.

In the present context, the primary problem with time is the loop over particles. The key to avoiding this loop is to construct Chebyshev polynomials for vectors of state variables, rather than individual elements.

Avoiding the Loop Over Particles

```
proc yci_of_kz_swarm(s,gam);
 // calculates y,c,i as functions of state. here s is nparts x nstates
 local blahblahblah;
 iii=1; do while iii<=nstates;
     ordiii = ord[iii];
     ntees = ordiii;
     tees = zeros(nparts,ntees);
     tees[.,1] = ones(nparts,1);
     tees[.,2] = s[.,iii];
     if ordiii > 2;
         j=3; do while j<=ntees;
             tees[.,j] =
        2*s[.,iii].*tees[.,j-1]-tees[.,j-2];
        j=j+1; endo;
     endif;
```

Avoiding the Loop Over Particles, cont.

```
if iii==1;
    oldswarm = tees;
    newswarm = oldswarm;
else;
    kkk=1; do while kkk<=ord[iii];
        if kkk==1;
            newswarm =
      oldswarm.*tees[.,kkk];
    else;
            newswarm =
      newswarm~(oldswarm.*tees[.,kkk]);
    endif;
        kkk=kkk+1; endo;
    endif;
    oldswarm = newswarm;
iii=iii+1; endo;
```

Avoiding the Loop Over Particles, cont.

```
 clev = newswarm*gam;
 klev = ss[4] + omegak*s[.,1];
 zlev = ss[5] + omegaz*s[.,2];
 ylev=zlev.*(klev.^p[1]);
 ilev = ylev-clev;
 retp(ylev~clev~ilev);
endp;
```