Chem. Eur. J. **2003**

Supporting Information

for

Tracking Chemical Kinetics in High-Throughput Systems

H. F. M. Boelens, D. Iron, J. A. Westerhuis and G. Rothenberg

This is an annotated MATLAB 6.1 m-file routine for quick evaluation of high-throughput sampling strategies.

```
function [o_xinf,o_times] = Compstrat(k,nmeas,Tmeas,Tinf)
% function [o_xinf,o_times] = Compstrat(k,nmeas,Tmeas,Tinf)
%
% Purpose:
% ========
% Compare the effect of the various sampling strategies
% for estimating the kinetic constant of a first order reaction
% using the results of Appendix of the publication:
% "TRACKING CHEMICAL KINETICS IN HIGH-THROUGHPUT SYSTEMS"
% Boelens, et al.
%
% Input:
% ======
% k        -   scalar.
%              Kinetic constant of first order reaction [min^-1].
%
% nmeas    -   scalar or vector.
%              Number of measurements to perform. If "nmeas" is
%              a vector the relative performance of the three
%              different sampling strategies will
%              be plotted as a function of the number of
%              measurements.
%
% Tmeas    -   Time needed to do a measurement [min].
%
% Tinf     -   Maximum time to consider [min].
%
%
% Output:
% =======
% o_xinf   -   vector or matrix
%              Each row (i) contains the squared sum of the
```

```
%             information vector for the sampling
%             strategies S1,S2,S3 and S4. The values
%             of the i-th row belong to the number of
%             measurements contained in: nmeas(i).
%             Example: When nmeas = [2,3], then first
%             row, o_xinf(1,:), contains results for
%             doing 2 measurements, and the second row
%             o_xinf(2,:) results for doing 3
%             measurements.
%
% o_times -   matrix
%             Times at which measurements are done for the
%             various sampling strategies:
%             o_times(:,1) - Strategy S1.
%             o_times(:,2) - Strategy S2.
%             o_times(:,3) - Strategy S3.
%             Only returned when "nmeas" is a scalar.
%
% Description:
% ============
% The next sampling strategies are considered (see publication):
% S1 Equidistant sampling along time axis.
% S2 Pack samples at start of reaction.
% S3 Equidistant sampling along concentration axis.
% S4 Theoretically optimal sampling strategy.
%
% Examples of use:
% ================
% Compstrat(0.20,3,1)
% Compare strategies for:
% + k = 0.20 min^-1
% + 3 measurements
% + Time for measurement is 1 min.
%
% Compstrat(0.20,2:1:20,2)
% Compare strategies for:
% + k = 0.20 min^-1
% + Test strategies for different number of measurements.
% + take as 2 minutes as the time needed to do a measurement.
close all % Close all figure windows.
xinf=[];stimes=[];c3=[];
if ( nargin < 1 )
    k = input('Kinetic constant [min^-1] (default: 0.23 min^-1) : ');
end
if ( isempty(k) )
    k = 0.23;
    fprintf('Compstrat: Kinetic constant %8.2f [min^-1]\n',k);
end
if ( ~all(size(k) == 1) )
    error('Compstrat: "k" must be a scalar')
end

if ( nargin < 2 )
    nmeas = input('Number of measurements (default: 3) : ');
end
if ( isempty(nmeas) )
    nmeas = 3;
end

if ( nargin < 3 )
    Tmeas = input('Time for a measurement [min^-1] (default: 10 sec) : ');
end
if ( isempty(Tmeas) )
```

```
   Tmeas = 10/60; % Tmeas = 10 sec
end

if ( nargin < 4 )
   % No Tinf specified. set to 6/k
   Tinf = 6/k;
end

Tdel      = Tmeas / 10;
Taxis     = 0:Tdel:Tinf;
c_curve   = exp(-k .* Taxis);
Dc_curve  = (Taxis .* exp(-k .* Taxis)).^2;

if ( isempty(nmeas) )
   nmeas = input('# measurements: ');
end

if ( size(nmeas) == [1,1] )

   % xinf(1) is squared sum of information vector for strategy 1.
   [xinf(1),F1,ts1]          = strat1(nmeas,k,Tmeas,Tinf);

   % xinf(2) is squared sum of information vector for strategy 2.
   [xinf(2),F2,ts2]          = strat2(nmeas,k,Tmeas);

   % xinf(3) is squared sum of information vector for strategy 3.
   [xinf(3),F3,ts3,cs3]      = strat3(nmeas,k,Tmeas);

   % xinf(4) is squared sum of information vector for best strategy.
   [xinf(4),F4,ts4]          = strat4(nmeas,k);

   if ( nargout == 0 )

      % Plot compares various strategies.
      figure(1)
      subplot(2,1,1)
      plot(Taxis,c_curve,'k-')
      hold on
      plot(ts1,zeros(size(ts1)),'ko','linewidth',1.5,'markersize',6,'markerface','r');
      plot(ts2,zeros(size(ts2)),'ks','linewidth',1.5,'markersize',6,'markerface','b');
      plot(zeros(size(cs3)),cs3,'kd','linewidth',1.5,'markersize',6,'markerface','g');
      legend('conc','strat: S1','strat: S2','strat: S3');
      axis([-0.1/k,Tinf,ylim]);
      lims = axis;
      ylabel('\it{a_{ t}}');

      subplot(2,1,2)
      plot(Taxis,Dc_curve,'k-');hold on
      plot(ts1,F1.^2,'ko','linewidth',1.5,'markersize',6,'markerfacecolor','r');
      plot(ts2,F2.^2,'ks','linewidth',1.5,'markersize',6,'markerfacecolor','b');
      plot(ts3,F3.^2,'kd','linewidth',1.5,'markersize',6,'markerfacecolor','g');
      plot([1/k,1/k],ylim,'r:')
      axis([lims(1:2),ylim]);
      xlabel('\it{Time / Min}');
      ylabel('\it{f^{ 2}}');

   else
      o_xinf    = xinf;
      o_times   = [ts1',ts1',ts3'];
   end

elseif ( min(size(nmeas)) == 1 )

```

```
   % Calculate xinf ( total information value ) for each
   % separate strategy. Loop runs over the number
   % measurements for which the calculations
   % are made.
   for ( imeas = 1:length(nmeas) )
      xinf(imeas,1) = strat1(nmeas(imeas),k,Tmeas,Tinf);
      xinf(imeas,2) = strat2(nmeas(imeas),k,Tmeas);
      xinf(imeas,3) = strat3(nmeas(imeas),k,Tmeas);
      xinf(imeas,4) = strat4(nmeas(imeas),k);
   end

   % Calculate relative preformace [percent] of strategies
   % S1,S2,S3 with respect to the best strategy (S4).
   xinf(:,1) = xinf(:,1) ./ xinf(:,4) * 100;
   xinf(:,2) = xinf(:,2) ./ xinf(:,4) * 100;
   xinf(:,3) = xinf(:,3) ./ xinf(:,4) * 100;

   if ( nargout == 0 )

      % Plot when variable "nmeas" is vector.
      figure(1)
      plot(nmeas,xinf(:,1),'ko','linewidth',1.5,'markersize',6,'markerfacecolor','r');
      hold on
      plot(nmeas,xinf(:,2),'ks','linewidth',1.5,'markersize',6,'markerfacecolor','b');
      plot(nmeas,xinf(:,3),'kd','linewidth',1.5,'markersize',6,'markerfacecolor','g');
      xlabel('# measurements');
      ylabel('Relative performance[%]');
      legend('S1','S2','S3');
      title(sprintf('k = %5.2f [min]^{-1} Tmeas: %5.2f [min]',k,Tmeas));

   else

      o_xinf = xinf;
      o_times = [];

   end

end

function [xinf1,F1,ts1] = strat1(nmeas,k,Tmeas,Tinf)
% +++++> Sampling strategy 1: Equidistant sampling along time axis.
   Tdel   = Tinf / (nmeas+1);
   if ( Tdel < Tmeas )
         Tdel = Tmeas;
   end
   ts1    = Tdel:Tdel: Tinf - Tdel;
   F1     = ts1 .* exp(-k .* ts1);
   xinf1  = sum(F1 .* F1);

function [xinf2,F2,ts2] = strat2(nmeas,k,Tmeas)
% +++++> Sampling strategy 2: Pack samples at start of reaction.
   ts2    = Tmeas:Tmeas:(nmeas) * Tmeas;
   F2     = ts2 .* exp(-k .* ts2);
   xinf2  = sum(F2 .* F2);

function [xinf3,F3,ts3,Cs3] = strat3(nmeas,k,Tmeas)
% +++++> Sampling strategy 3: Equidistant sampling concentration axis.
   Cdel   = 1 / (nmeas+1);
   Cs3    = 1-Cdel:-Cdel:0+Cdel;
   ts3    = -log(Cs3)/k;
   ts3_0  = [0,ts3];
   index  = find ( diff(ts3_0) < Tmeas );
   done   = 0;
```

```
  while ( ~isempty(index) )
        for ( i = 1:length(index) )
              ts3(index(i)) = ts3_0(index(i)) + Tmeas + 100*eps;
              ts3_0 = [0,ts3];
        end
        index = find ( diff(ts3_0) < Tmeas );
        if ( done )
              fprintf('%d ',nmeas);
              done = 1;
        end
  end
  fprintf('\n')
  F3 = ts3 .* exp(-k .* ts3);
  xinf3 = sum(F3 .* F3);

function [xinf4,F4,ts4] = strat4(nmeas,k)
% +++++> Sampling strategy 4: Optimal sampling.
  ts4    = 1/k * ones(nmeas,1);
  F4     = ts4 .* exp(-k .* ts4);
  xinf4  = sum(F4 .* F4);
```