Numerical methods of the fractional calculus

(continued II)

Computation of binomial coefficients

The following coefficients appear frequently (e.g., in G1):

$$w_k^{(\alpha)} = (-1)^k \binom{\alpha}{k}, \qquad k = 0, 1, 2, \ldots,$$

It is possible to use the following recurrence:

$$w_0^{(\alpha)} = 1; \quad w_k^{(\alpha)} = \left(1 - \frac{\alpha+1}{k}\right) w_{k-1}^{(\alpha)}, \quad k = 1, 2, 3, \ldots$$

Possible MATLAB code:

```
function y=bcrecur(a, n)
y=cumprod([1, 1 - ((a+1) ./ [1:n])]);
```


Computation of binomial coefficients

```
>> bcrecur(1.8,10)
ans =
  Columns 1 through 8
    1.0000   -1.8000    0.7200    0.0480    0.0144    0.0063    0.0034    0.0020
  Columns 9 through 11
    0.0013    0.0009    0.0007
>> bcrecur(2,10)
ans =
     1    -2     1     0     0     0     0     0     0     0     0
>> bcrecur(3,10)
ans =
  Columns 1 through 8
    1.0000   -3.0000    3.0000   -1.0000         0         0         0         0
  Columns 9 through 11
         0         0         0
```

for each routine make a qualitative check

and check the backward compatibility for integer orders

Computation of the Mittag-Leffler function

(1) Directly using the definition:

```
function y=mitlef(alpha,beta,z,N)

%   Evaluation of the Mittag-Leffler function in two parameters:
%   E_{alpha,beta}(z), using its series expansion.
%
% PARAMETERS:
%   alpha   - first index (scalar), beta   - second index (scalar)
%   z    - row vector of values of the function argument
%   N    - number of terms in the power expansion (scalar)
%          OPTIONAL, default N=100.

if nargin<4, N=100; end

m1=max(size(z));
m2=min(size(z));
if m2>1, z=z(1,:); end

k=repmat((1:N)',1,m1);
t=repmat(z,N,1);
t=t.^(k-1);
a=repmat(gamma(alpha*((1:N)'-1)+beta), 1, m1);
y=sum(t./a);
```


Computation of the Mittag-Leffler function

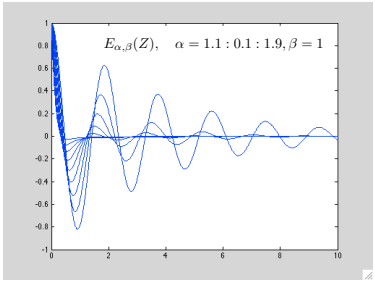
(2) There is a routine at MATLAB File Exchange:

http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8738&objectType=FILE

`MLF(alpha,beta,Z,P)`
is the Mittag-Leffler function $E_{alpha,beta}(Z)$ evaluated with accuracy 10^{-P} for each element of `Z`.

Computation of the Mittag-Leffler function

(2) There is a routine at MATLAB File Exchange:



Computation of fractional derivatives

Hany Farid:

```
%%% FRACTIONAL DERIVATIVES (12.14.00)
%%% Hany Farid (farid@cs.dartmouth.edu | www.cs.dartmouth.edu/~farid)
%%%

clear;
set( gcf, 'Renderer', 'zbuffer' );

dim    = 256;
ramp   = [-dim:dim-1];
ramp   = pi * ramp/dim;
f      = exp( -(ramp.^2)/(0.5) );          % GAUSSIAN
f      = f - mean(f);                      % ZERO-MEAN:AVOID DISCONTINUITY
F      = fftshift( fft( f ) );             % FOURIER TRANSFORM

%%% CYCLE THROUGH FRACTIONAL DERIVATIVES
for n = 0 : 0.1 : 4
        Fn     = (j * ramp).^n .* F;       % MULIPLY BY RAMP^n
        fn     = ifft( ifftshift(Fn) );    % INVERSE FOURIER TRANSFORM
        if( mean( abs(imag(fn)) ) > 1e-5 )  % FOR REAL SIGNALS
          fprintf( 1, 'Non-zero imaginary component (%f)\n',mean(abs(imag(fn))) );
          return;
        end
        fn     = fn / max(abs(fn));        % NORMALIZE
        t = sprintf( '%0.2f', n );         % PLOT...
        plot( real(fn) );
        text( 20, 1.0, sprintf( '%0.2f', n ) );
        axis( [0 2*dim -1.2 1.2] ); axis square;
        h = line( [dim dim], [-1.2 1.2] );
        set( h, 'LineStyle', ':' );
        drawnow;
end
```


Digital fractional differentiators

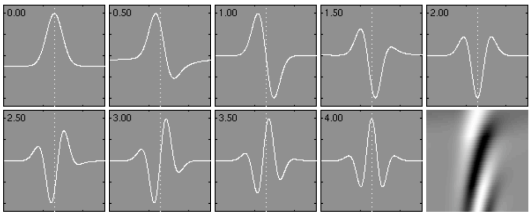
Ivo Petras -- IIR version:

```
function sysdfod=dfod1(n,T,a,r)
% sysdfod=dfod1(n,T,a,r): new digital fractional order differentiator
%                         and integrator
%
% Output: =>
% Discrete system in the form of the rational approximation - IIR filter
% obtained by continued fraction expansion of a new generating function.
%
% Inputs: <=
% n: order of truncation n:=[1 2 3 4 5]
% T: sampling period in [sec]
% a: weighting factor between Euler and Tustin rules ( 0 <= a <= 1 )
%    0 - Euler rule, 1 - Tustin rule, 1/2 - Al-Alaoui rule, etc.
% r: approximated fractional order (s^r), r is generally real number
%
% Author: Dr. Ivo Petras (ivo.petras@mail.com)
% URL: http://ivopetras.tripod.com/
%
% Note: This approach is based on the original Al-Alaoui's work.
%
% Copyright (c), 2003.
%
k = ((1+a)/T)^r;
if (n>5)
    n=5;
    disp('Maximal term is clipped to 5!');
end
if (n==1)
    b0=2/(a*r+a*r-1);
    b1=-(r+a*r+a+1)/(a*r+a*r-1);
    a0=2/(a*r+a*r-1);
    a1=1;
    sys1=tf(k*[b0 b1],[a0 a1],T,'Variable', 'z^-1');
    sysdfod=sys1;
elseif (n==2)
    b0=12/(a^2*r^2+3*a^2*r+2*a*r^2+2*a*r^2-2*a-3*r+2);
    b1=(12*a-6*r-6*a*r-12)/(a^2*r^2+3*a^2*r+2*a*r^2+2*a*r^2-2*a-3*r+2);
    b2=(a^2*r^2-3*a^2*r+2*a*r^2+2*a*r^2-2*a+3*r+2*a^2+2*a^2*r^2+3*a^2*r+2*a*r^2+2*a*r^2-2*a-3*r+2);
    a0=12/(a^2*r^2+3*a^2*r+2*a*r^2+2*a*r^2-2*a-3*r+2);
    a1=(12*a+6*r+6*a*r-12)/(a^2*r^2+3*a^2*r+2*a*r^2+2*a*r^2-2*a-3*r+2)
    a2=1;
    sys2=tf(k*[b0 b1 b2],[a0 a1 a2],T,'Variable', 'z^-1');
    sysdfod=sys2;
elseif (n==3)
    b0=120/(a^3*r^3+6*a^3*r^2+3*a^2*r^3+11*a^3*r+6*a^2*r^2+3*a*r^3+3*a^3+6*a^3-27*a^2*r-6*a*r^2+r^3-54*a^2-27*a*r-6*r^2+54*a+11*r-6);
```


Computation of fractional derivatives

Hany Farid (www.cs.dartmouth.edu/~farid):

Zeroth through fourth-order normalized derivatives of a Gaussian in steps of 0.5.



Digital fractional differentiators

YangQuan Chen -- IIR version:

```
% New DFOD
function sysdfodnewdfod(n,T,a,s)
% n order of truncation (1 2 3 ...)
% T sampling period in sec.
% a weighting factor between Simpson and Tustin ( 0 <= a <= 1 )
z2=(3*a-2*sqrt(3*a))/(3-a);b=sqrt(3*(1-a))*z;
if (n==1)
    b1=-(2*b/2*s+b*2-1);
    b0=2*b/(2*b/2*s+b*2-1);
    a0=1;
    a1=1;
    a0=2*b/(2*b/2*s+b*2-1);
    sys1=tf([b0 b1],[a0 a1],T,'Variable', 'z^-1');
    sysdfod=sys1;
elseif (n==2)
    den2=(8*b^4*s^4+12*b^4*s^3+24*b^4*s^2-3*b^4*s-12*b^4*s^3-b^6-36*b^4*s^2+3*b^4+3*b^4+18*b^2*s^2-9*b^2*s+9*b^2+9*s-9);
    b2=(18*b^2*s^2+3*b^4*s^2+6*b^2*s^2+3*b^4*s+2*b^6*s^2-12*b^4*s^3-3*b^4+12*b^4*s^3-3*b^4*s+9*b^2*s-9*s+9*b^4+9*b^2-9*s)/den2;
    b1=(-24*s^2*b^3-24*s^3*b^5-12*b^3*s^3+18*b+36*s*b^3+18*s^6-6*s*b^5+24*s^2*b^5-6*b^5)/den2;
    b0=(18+24*b^4*s^2-6*b^4-36*b^2)/den2;
    a
    a2=1;
    a1=(-24*s^2*b^3+24*s^3*b^5-12*b^3*s^3+18*b-36*s*b^3+18*s^6-6*s*b^5+24*s^2*b^5-6*b^5)/den2;
    a0=(18+24*b^4*s^2-6*b^4-36*b^2)/den2;
    sys2=tf([b0 b1 b2],[a0 a1 a2],T,'Variable', 'z^-1');
    sysdfod=sys2;
elseif (n==3)
den3=(64*b^12*s^7+192*b^12*s^6+144*b^12*s^5-48*b^12*s^4-192*b^10*s^6-64
  -1440*b^10*s^5-1296*b^12*s^2-162*b^10*s^4+11*b^12*s+720*b^10*s^3+720
  -90*b^10*s+2520*b^8*s^3-1080*b^10-3240*b^8*s^2-480*b^6*s^6-675*b^8*s
  -1440*b^6*s^5+1440*b^8*s+1080*b^6*s^4+1440*b^4*s^6+1350*b^6*s^3-1350
b3=-(675*s^2*b^8+3*b^8*s^4-240*b^6*s^4+540*b^6*s+108*b^10*s^3+360*b^2*s+2700*s
  -8100*b^6*s^2-2475*b^4*s-720*b^4*s^3+9480*b^4*s^2+180*b^6*s^4+480*b^6
  -324*b^8*s^2-675*b^8*s-720*b^4*s^4+675*s^6+90*s^4+180*b^4*s^2-720*
  1440*b^10*s^5+192*b^10*s^6+11*b^12*s+12*b^12*s^2-864*b^12*s^3+48*b^12
  +64*b^12*s^7-36*b^12/den3;
b2=-(-5400*b+6300*b^3-1440*b^5-3240*b^7-2700*s*b-360*b^9+2700*s*b^3-1800
  -1800*s*b^7-1224*b^7*s^2+720*s^3*b^3-288*s^4*b^7+1280*s*b-3280
  -480*b^11*s^3+960*b^11*s^5-384*b^11*s^6+264*b^11*s^2-384*b^11*s^4-36
b1= (-6300*b^4-8460*b^6+3*s1080*b^4*s-8100*b^2*s+3420*b^6+360*b^4*s^2-
  -1440*b^8*s^4-1440*b^10*s^4+960*b^10*s^5-13680*b^6*s^2+6750*b^2-480
  +1350-720*b^10*s^2)/den3;
b0= (-10800*b^3+10800*b^5+8100*b-2160*b^7-960*b^9*s^4+480*b^9*s^2-60*b
% denominator
a3=1;
a2=(5400*b-6300*b^3+1440*b^5-3240*b^7-2700*s*b+360*b^9+2700*s*b^3-1800
  -1800*s*b^7+1224*b^7*s^2+720*s^3*b^3-288*s^4*b^7+1280*s*b^3-3280
  -480*b^11*s^3+960*b^11*s^5-384*b^11*s^6+264*b^11*s^2-384*b^11*s^4-36
a1=(6300*b^4-8460*b^6+3*s1080*b^4*s-8100*b^2*s-3420*b^6-360*b^4*s^2-
  -1440*b^8*s^4+1440*b^10*s^4+960*b^10*s^5+13680*b^6*s^2-6750*b^2-480*b^10*s^3+2430*b^8-8360*b^8*s^2+480*b^10*s^2
  +1350-720*b^10*s^2)/den3;
```


Digital fractional differentiators

Ivo Petras -- Finite Impulse Response (FIR) version:

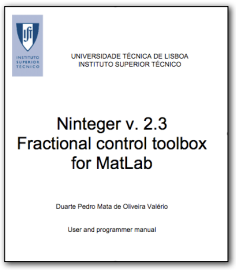
```
function sysdfod=dfod2(n,T,r)
% sysdfod=dfod2(n,T,r): digital fractional order differentiator
%                       and integrator
%
% Output: =>
% Discrete system in the form of the FIR filter of the order n
% obtained by power series expansion of the backward difference.
%
% Inputs: <=
% n: order of truncation (min n=100 is recommended)
% T: sampling period in [sec]
% r: approximated fractional order (s^r), r is generally real number
%
% Author: Dr. Ivo Petras (ivo.petras@mail.com)
% URL: http://ivopetras.tripod.com/
%
% Note: differentiator  -> nonrecusrive approximation
%       integrator      -> recursive approximation
%
% Copyright (c), 2003.
%
if r>0
   bo=cumprod([1,1-((r+1)./[1:n])]);
   sysdfod=filt(bo,[T^r],T);
end
if r<0
   bo=cumprod([1,1-((-r+1)./[1:n])]);
   sysdfod=filt([T^-r],bo,T);
end
%
```


Digital fractional differentiators

YangQuan Chen --

Oustaloup's recursive approximation for fractional order differentiators:

NINTEGER Toolbox by Duarte Valerio



NINTEGER Toolbox by Duarte Valerio

Table of contents

NINTEGER Toolbox by Duarte Valerio

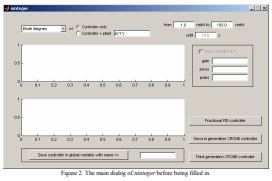


Figure 2. The main dialog of ninteger before being filled in.

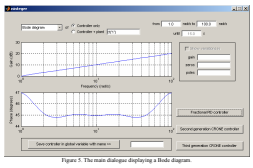


Figure 5. The main dialogue displaying a Bode diagram.

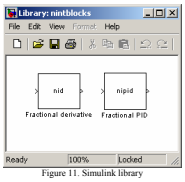


Figure 11. Simulink library

Sample applications

Fractional derivatives of order 1/2 can appear even within the context of the integer-order models!

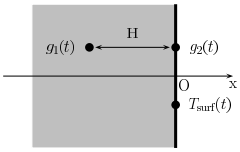
Heat flux in a blast furnace wall

$$\hat{c}\hat{\rho}\frac{\partial T}{\partial t} = \hat{\lambda}\frac{\partial^2 T}{\partial x^2}, \qquad (t>0, \quad -\infty < x < 0)$$

$$T(0,x) = T_0$$

$$T(t,0) = T_{\text{surf}}(t)$$

$$\left|\lim_{x\to-\infty} T(t,x)\right| < \infty$$



t - is time [s],
x - is the spatial coordinate in the direction of heat conduction [m],
$\hat{c}$ - is heat capacity $[J\,kg^{-1}\,K^{-1}]$,
$\hat{\rho}$ - is mass density $[kg\,m^{-3}]$,
$T(t,x)$ - is temperature $[K]$,
$\hat{\lambda}$ - is the coefficient of heat conductivity $[W\,m^{-1}\,K^{-1}]$

Heat flux in a blast furnace wall

Introduce an auxiliary function

$$u(t,x) = T(t,x) - T_0.$$

which must be a solution of the following problem:

$$\hat{c}\hat{\rho}\frac{\partial u}{\partial t} = \hat{\lambda}\frac{\partial^2 u}{\partial x^2}, \qquad (t>0, \quad -\infty < x < 0)$$

$$u(0,x) = 0$$

$$u(t,0) = T_{\text{surf}}(t) - T_0$$

$$\left|\lim_{x\to-\infty} u(t,x)\right| < \infty$$

Heat flux in a blast furnace wall

The Laplace transform gives:

$$s\,\hat{c}\hat{\rho}U(s,x) = \hat{\lambda}\frac{d^2U(s,x)}{dx^2}$$

The solution bounded at $x \to -\infty$ is:

$$U(s,x) = U(s,0)\exp\left(x\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\right)$$

Then we have

$$\frac{dU}{dx}(s,x) = U(s,0)\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\,\exp\left(x\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\right)$$

Heat flux in a blast furnace wall

$$U(s,x) = U(s,0)\exp\left(x\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\right) \qquad \frac{dU}{dx}(s,x) = U(s,0)\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\,\exp\left(x\sqrt{\frac{\hat{c}\hat{\rho}s}{\hat{\lambda}}}\right)$$

$$\frac{1}{\sqrt{s}}\frac{dU}{dx}(s,0) = \sqrt{\frac{\hat{c}\hat{\rho}}{\hat{\lambda}}}\,U(s,0)$$

The inverse Laplace transform gives:

$${}_0D_t^{-1/2}\frac{\partial u}{\partial x}(t,0) = \sqrt{\frac{\hat{c}\hat{\rho}}{\hat{\lambda}}}\,u(t,0)$$

After fractional integration of both sides:

$$\frac{\partial u}{\partial x}(t,0) = \sqrt{\frac{\hat{c}\hat{\rho}}{\hat{\lambda}}}\,{}_0D_t^{1/2}u(t,0)$$

Heat flux in a blast furnace wall

$$\frac{\partial u}{\partial x}(t,0) = \sqrt{\frac{\hat{c}\hat{\rho}}{\hat{\lambda}}}\,{}_0D_t^{1/2}u(t,0) \qquad u(t,x) = T(t,x) - T_0,$$

$$q_A(t) = \sqrt{\hat{c}\hat{\rho}\hat{\lambda}}\,{}_0D_t^{1/2}g(t), \quad g(t) = T_{\text{surf}}(t) - T_0$$

$$q_A(t) = \hat{\lambda}\frac{\partial T}{\partial x}(t,0)$$

the heat flux

Heat flux in a blast furnace wall

Numerical solution: short memory principle

$$q_A(t) \approx \tilde{q}_A(t) = \sqrt{\hat{c}\hat{\rho}\hat{\lambda}}\,{}_{(t-L)}D_t^{1/2}g(t)$$

Normalized error:

$$\delta_0 = \frac{|q_A(t) - \tilde{q}_A(t)|}{M} = \frac{1}{\sqrt{L}\,\Gamma(\frac{1}{2})}, \quad M = \max_{[0,\infty]}|g(t)|$$

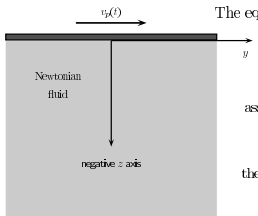
Condition on the memory length: $L \geq \dfrac{1}{\pi\delta_0^2}$

Calculate using:

$${}_{(t-L)}D_t^{1/2}g(t) = \tau^{-\alpha}\sum_{i=0}^{N(t)} c_i g(t - i\tau),$$

$$N(t) = \min\left\{\left[\frac{t}{\tau}\right], \left[\frac{L}{\tau}\right]\right\}, \quad c_i = (-1)^i\binom{1/2}{i}$$

A plate on a Newtonian fluid



The equation of motion of the fluid is

$$\rho\frac{\partial v}{\partial t} = \mu\frac{\partial^2 v}{\partial z^2}, \qquad (0 < t < \infty, \quad -\infty < z < 0)$$

assume that initially the fluid is in equilibrium state.

$$v(0,z) = 0, \qquad (-\infty < z < 0)$$

the influence of the plate's motion vanishes for $z \to \infty$:

$$v(t,-\infty) = 0, \qquad (0 < t < \infty)$$

The fluid's velocity at $z = 0$ is equal to the given velocity of the plate:

$$v(t,0) = v_p(t)$$

A plate on a Newtonian fluid

Applying the Laplace transform gives:

$$\rho s V(s,z) = \mu\frac{d^2V(s,z)}{dz^2}$$
$$V(s,0) = V_p(s),$$
$$V(s,-\infty) = 0,$$

The solution of this problem is:

$$V(s,z) = V_p(s)\exp\left(z\sqrt{\frac{\rho s}{\mu}}\right)$$

A plate on a Newtonian fluid

By differentiation we obtain:

$$\frac{dV(s,z)}{dz} = \sqrt{\frac{\rho s}{\mu}} V_p(s) \exp\left(z\sqrt{\frac{\rho s}{\mu}}\right) = \sqrt{\frac{\rho s}{\mu}} V(s,z)$$

Knowing the velocity, we can obtain the shear stress:

$$\sigma(t,z) = \mu \frac{\partial v(t,z)}{\partial z}$$

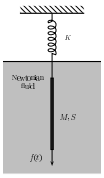
which in terms of the Laplace transforms reads:

$$\overline{\sigma}(s,z) = \mu \frac{dV(s,z)}{dz} = \sqrt{\mu\rho s}\, V(s,z).$$

and therefore, in time domain we have:

$$\sigma(t,z) = \sqrt{\mu\rho}\, {}_0D_t^{1/2} v(s,z)$$

Motion of an immersed plate



Considering the forces we have:

$$My''(t) = f(t) - Ky(t) - 2S\sigma(t,0)$$

Using the relationships

$$\sigma(t,z) = \sqrt{\mu\rho}\, {}_0D_t^{1/2} v(s,z) \qquad v_p(t,0) = y'(t)$$

we arrive at the following equation:

$$Ay''(t) + B\, {}_0D_t^{3/2} y(t) + Cy(t) = f(t) \qquad (t>0),$$

$$A = M, \quad B = 2S\sqrt{\mu\rho}, \quad C = K,$$

add initial conditions! $\quad y(0) = 0, \qquad y'(0) = 0$

Motion of an immersed plate

Sample numerical solution

```
% Script for solving Bagley-Torvik equation with zero
initial conditions:

clear all

alpha=1.5;

% Numerical solution:
h=0.075;
T=0:h:30;
N=30/h + 1;
B=zeros(N,N);

% (1) Make the main matrix B1_{N}^{alpha}:
bc=fliplr(bcrecur(alpha,N-1));
for k=1:N
   B1(k,1:k)=bc((N-k+1):N);
end
```


Motion of an immersed plate

Sample numerical solution

```
% (2) Make the main matrix B2_{N}^{2}:
alpha=2;
bc=fliplr(bcrecur(alpha,N-1));
for k=1:N
   B2(k,1:k)=bc((N-k+1):N);
end

% (3) Make the matrix for the entire equation:
B=h^(-2)*B2 + h^(-1.5)*B1 + eye(size(B1));

% Make right-hand side:
F=8*(T<=1)';
```


Motion of an immersed plate

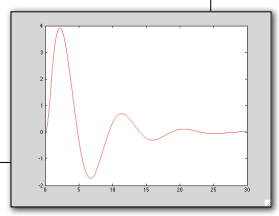
Sample numerical solution

```
% Make right-hand side:
F=8*(T<=1)';

% (3) Use zero initial conditions:
F(1)=0;
F(2)=0;
B(1,1)=1;
B(2,1)=-1/h; B(2,2)=1/h;

% (4) Solve the system BY=F:
Y=B\F;

% Plot the solution:
figure(1)
plot(T,Y,'r')
```



Short memory principle for FDEs

Consider the problem:

$${}_0D_t^{3/2} y(t) + y(t) = f(t), \qquad (t>0)$$

$$y(0) = y'(0) = 0$$

for the following RHS:

1. $f(t) \equiv 1$
2. $f(t) = te^{-t}$
3. $f(t) = t^{-1}e^{-1/t}$
4. $f(t) = e^{-t}\sin(0.2t)$

Short memory principle for FDEs

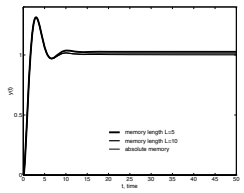


Figure 8.8: *Solution of the problem (8.56) for* $f(t) \equiv 1$

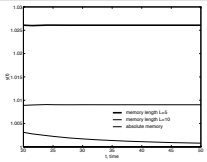


Figure 8.9: *Solution of the problem (8.56) for* $f(t) \equiv 1$ *(zoom)*

Short memory principle for FDEs

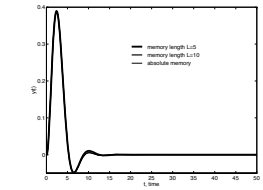


Figure 8.10: *Solution of the problem (8.56) for* $f(t) = te^{-t}$

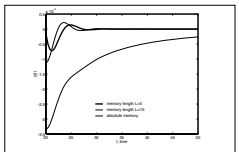


Figure 8.11: *Solution of the problem (8.56) for* $f(t) = te^{-t}$ *(zoom)*

Short memory principle for FDEs

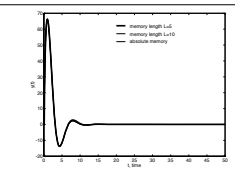


Figure 8.12: *Solution of the problem (8.56) for* $f(t) = t^{-1/2}e^{-1/t}$

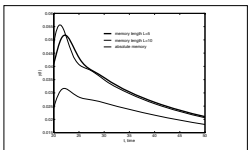


Figure 8.13: *Solution of the problem (8.56) for* $f(t) = t^{-1/2}e^{-1/t}$ *(zoom)*

Short memory principle for FDEs

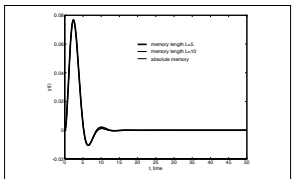


Figure 8.14: *Solution of the problem (8.56) for* $f(t) = e^{-t}\sin(0.2t)$

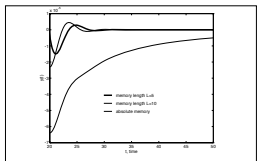


Figure 8.15: *Solution of (8.56) for* $f(t) = e^{-t}\sin(0.2t)$ *(zoom)*