Tutorials and worked examples for simulation, curve fitting, statistical analysis, and plotting.
http://www.simfit.org.uk

The evaluation of convolution integrals is frequently required in simulation or data analysis, e.g., where such integrals occur as output functions from the response of some device to an input function.

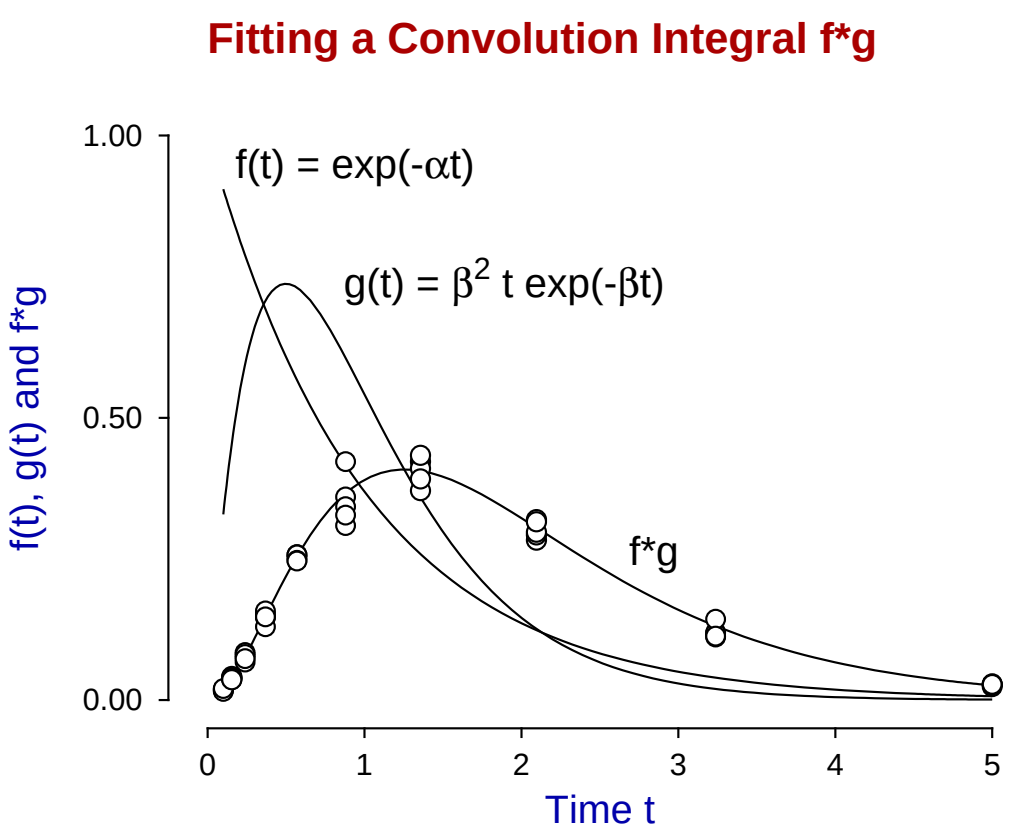
For instance, fitting convolution integrals involves parameter estimation in model functions $f(t)$, $g(t)$, and their convolution $(f * g)(t)$, where

$$(f * g)(t) = \int_0^t f(u)g(t-u)\,du.$$

Sometimes the input function can be controlled independently so that, from sampling the output function, parameters of the response function can be estimated, and frequently the functions may be normalized, e.g. the response function may be modeled as a function integrating to unity as a result of a unit impulse at zero time. However, any one, two or even all three of the functions may have to be fitted. The next figure illustrates fitting data in `convolv.tfl` using the model `convolv3.mod`, i.e.

$$f(t) = \exp(-\alpha t)$$
$$g(t) = \beta^2 t \exp(-\beta t)$$
$$1 = \int_0^\infty g(t)\,dt$$

with SIMFIT program **qnfit**.



In some cases a convolution integral can be performed explicitly, say by Laplace transforms, but SIMFIT provides a method to evaluate two arbitrary functions $f(\Theta, x), g(\Theta, x)$ and the convolution integral $f * g$ at the same time so that plotting and estimation of parameters Θ can be performed on any combination of the two functions and the convolution integral. The next example shows how the test file `convolv3.mod` is formatted. Note the following special commands.

- *putpar*
  This ensures that parameters are global, i.e., available to all models.
- *user*1(x, m)
  This reads m then x off the stack and adds submodel m evaluated at argument x.
- *begin model*(m), and *end model*(m)
  These define the start and completion of code to define model m.

In the results displayed above, the library file for the data was `convolv3.tfl` as follows.

```
Convolution data
%
%
convolv3.data
```

Percentage signs indicate that only output data was supplied in data file `convolv3.data` so only the convolution was fitted to the data, and plots for the input and transformation functions were for information.

```
 %
This demonstrates how to define 2 equations as sub-models, using
the command putpar to communicate parameters to the sub-models,
and the command convolute(1,2) to integrate sub-models 1 and 2
(by adaptive quadrature) from blim(1) = 0 to t = tlim(1) = x.
Precision of D01AJF quadrature is controlled by epsabs and epsrel
and blim(1) and tlim(1) must be used for the convolution limits
which, in this case are 0 to x, where x > 0 by assumption.
.............................................................
convolution integral: from 0 to x of f(u)*g(x - u) du, where
f1(t) = f(t) = exp(-p(1)*t)
f2(t) = g(t) = [p(2)^2]*t*exp(-p(2)*t)
f3(t) = f*g  = f1*f2
.............................................................
Note that usually extra parameters must be supplied if it wished
to normalise so that the integral of f or g or f*g is specified
(e.g.,equals 1) over the total range of possible integration.
This must often be done, e.g., if g(.) is a density function.
The gamma distribution normalising  factor p(2)**2 is stored in
this example to avoid unnecessary re-calculation.
 %
 3 equations
 1 variable
 2 parameters
 %
 putpar
 p(2)
 p(2)
 multiply
 put(1)
 1
 x
```

```
 user1(x,m)
 f(1)
 2
 x
 user1(x,m)
 f(2)
 0.0001
 epsabs
 0.001
 epsrel
 0
 blim(1)
 x
 tlim(1)
 convolute(1,2)
 f(3)
 %
begin{model(1)}
 %
Example: exponential decay, exp(-p(1)*x)
 %
 1 equation
 1 variable
 1 parameter
 %
 p(1)
 x
 multiply
 negative
 exponential
 f(1)
 %
end{model(1)}
begin{model(2)}
 %
Example: gamma density of order 2
 %
 1 equation
 1 variable
 2 parameters
 %
 p(2)
 x
 multiply
 negative
 exponential
 x
 multiply
 get(1)
 multiply
 f(1)
 %
end{model(2)}
```