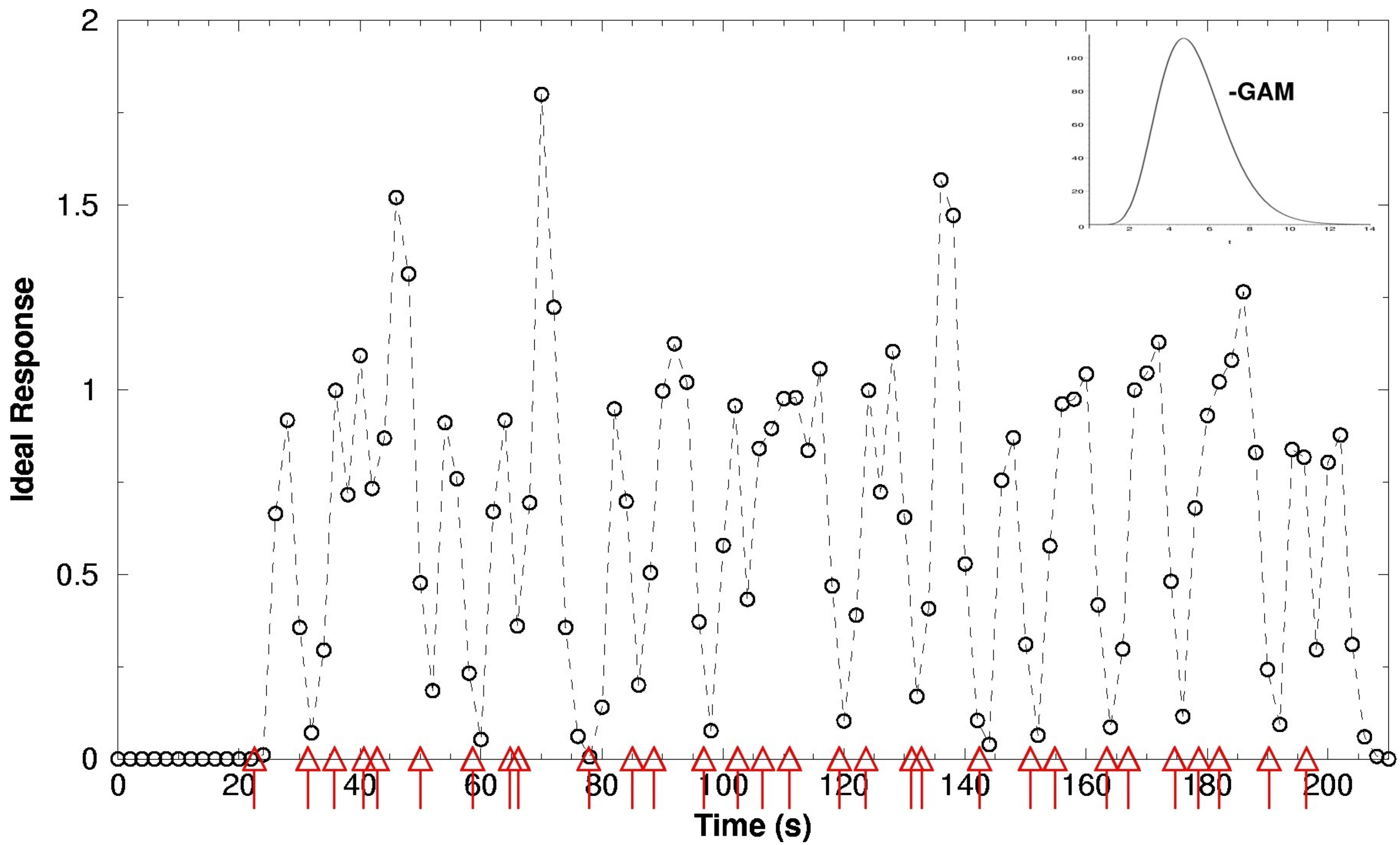
Irregular Stimulus Timing: Analysis with 3dDeconvolve

- 3dDeconvolve is set up to calculate response functions $h(t)$ when stimuli occur locked to image acquisition TR interval
  - ◇ The `-stim_nptr` option allows stimuli to occur at intervals of TR/p, where p is a small integer
  - ◇ But it seems like there is no way to do deconvolution for a completely irregular stimulus pattern
- The `waver -tstim` option does allow you to input irregular stimulus timing and generate a synthetic response
  - ◇ The result is the convolution of the hemodynamic response function $h(t)$ with a sequence of δ-functions at the stimulus times
  - ◇ $h(t)$ is chosen using the `-GAM`, `-WAV`, or `-EXPR` options to `waver`
- Simulating a time series:
  - ◇ `waver -dt 2.0 -GAM -peak 1 -tstim ‘cat tstim.1D‘ > ideal.1D`
  - ◇ File `tstim.1D` (30 stimulus times, averaging about 6 s apart):

```
22.6 31.4 35.8 40.6 42.8 50.0 58.6 66.2 64.8 77.8 85.0 88.6 96.8 102.4 106.6 111.0
119.2 123.6 131.2 132.8 142.4 150.8 154.8 163.4 167.0 174.6 178.6 182.0 190.2 196.4
```

◇ Graph of response `ideal.1D` (circles/dashed lines) with triangles △ at `tstim.1D` stimulus times, for 106 volumes to be simulated at TR=2 s:



◇ Despite appearances, this is <u>without noise</u>: fluctuations are just from different overlaps of hemodynamic response, since stimuli are not evenly spaced in time

◇ Goal of deconvolution is to retrieve hemodynamic response function (amplitude and shape) in each voxel, and test whether it is significantly different from zero

- Deconvolution, in the earlier presentations, models $h(t)$ itself as a sequence of evenly spaced δ-functions

$$h(t) = \sum_{a=\texttt{minlag}}^{\texttt{maxlag}} \lambda_a \cdot \delta(t - a \cdot \text{TR})$$

  ◇ Goal is to calculate the amplitude λ_a of each δ-function (in each voxel)

  ◇ Drawback to this description of $h(t)$ is that it says nothing about what the response is at times between the TR (image data) grid

  ↪ That's OK when stimuli are on the TR grid, but not when stimulus times are arbitrary

  ◇ Need to model $h(t)$ so can calculate response at an arbitrary time after the stimulus, not just at a fixed set of times

- Solution is to write response function $h(t)$ as the sum of a small number of basis functions:

$$h(t) = \sum_{a=1}^{N_a} \lambda_a \cdot \psi_a(t)$$

  ◇ Each $\psi_a(t)$ is a fixed function and the unknown amplitudes λ_a are to be determined from data (for each voxel)

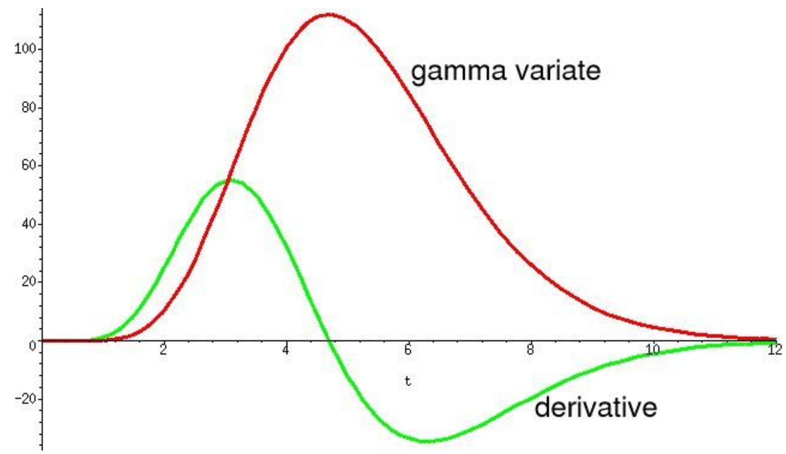
  ◇ Must choose the $\psi_a(t)$ basis functions based on the duration and shape of response you expect

- Basis functions:

  ◇ Gamma variate with derivative:

$$\psi_1(t) = t^b \exp(-t/\tau)$$
$$\psi_2(t) = \frac{d}{dt}\psi_1(t)$$

  for some fixed b, τ parameters
  [SPM approach]



  ↪ Basic idea: time shifted response $\psi_1(t+\Delta) \approx \psi_1(t) + \Delta \cdot \psi_1'(t)$ for small Δ

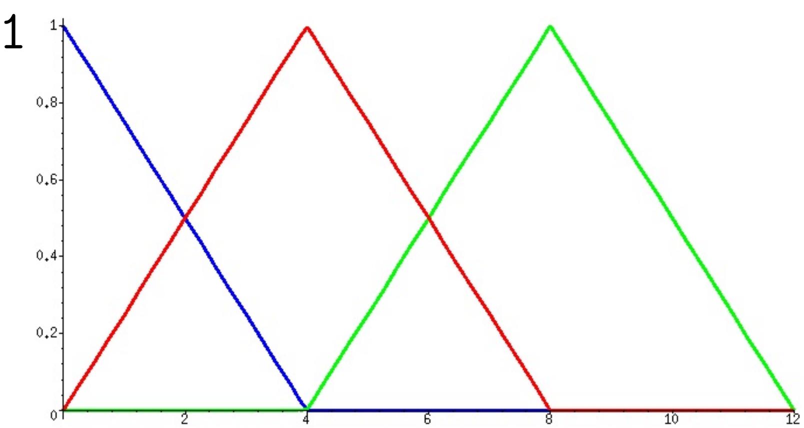
  ↪ Using both functions allows for small time shifts (0–3 s) in $h(t)$ — which using only $\psi_1(t)$ does not

  ◇ 'Tent' functions:

$$\texttt{tent(x)} = \begin{cases} 1-|\texttt{x}| & \text{if } -1 < \texttt{x} < 1 \\ 0 & \text{if } |\texttt{x}| \geq 1 \end{cases}$$

$$\psi_1(t) = \texttt{tent(t/4)}$$
$$\psi_2(t) = \texttt{tent((t-4)/4)}$$
$$\psi_3(t) = \texttt{tent((t-8)/4)}$$

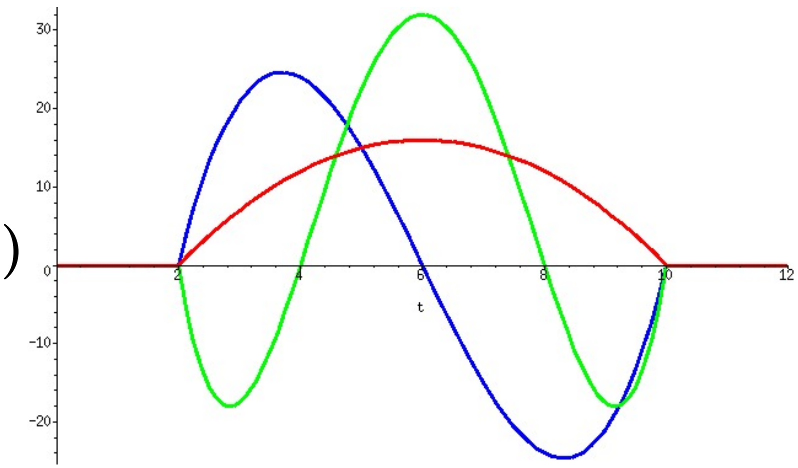
  [`tent()` is part of `waver`, `3dcalc`, etc.]

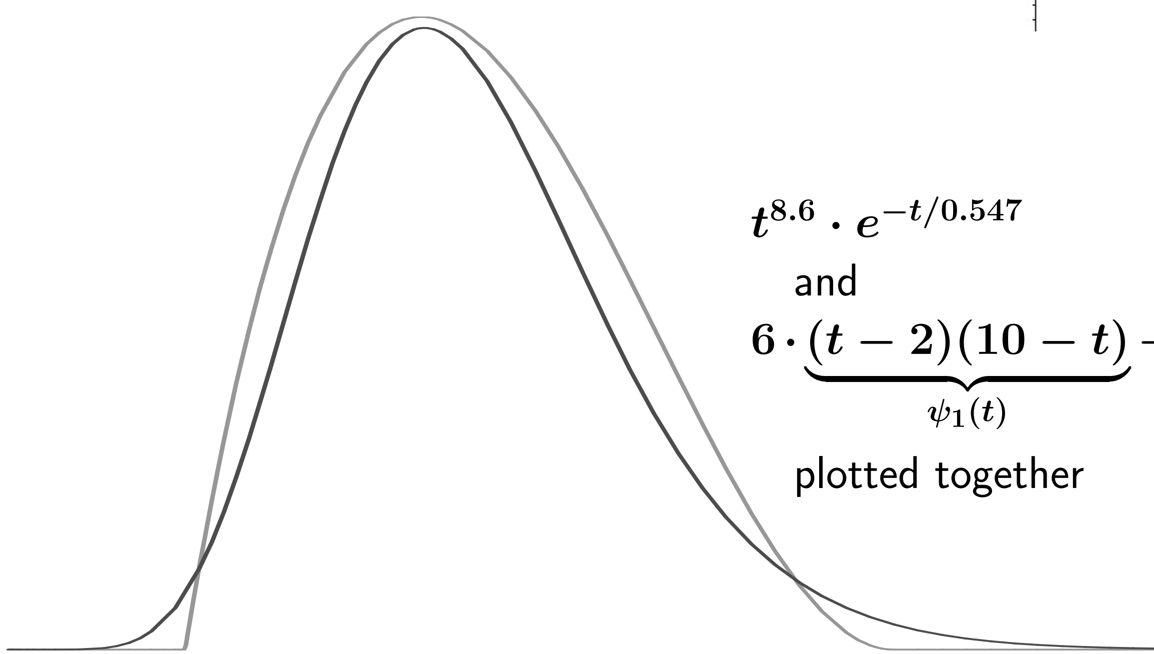


◇ Polynomials:

$$\psi_1(t) = (t-2)(10-t)$$
$$\psi_2(t) = (t-2)(10-t)(6-t)$$
$$\psi_3(t) = (t-2)(10-t)(t-4)(8-t)$$

for $2 < t < 10$





$t^{8.6} \cdot e^{-t/0.547}$

and

$$6 \cdot \underbrace{(t-2)(10-t)}_{\psi_1(t)} + 1.5 \cdot \underbrace{(t-2)(10-t)(6-t)}_{\psi_2(t)}$$

plotted together

◇ Other possibilities include trigonometric and sinc functions

↪ Both are common bases for function approximation in numerical analysis

◇ Want to keep number of basis functions N_a small, so that model for signal doesn't have too many parameters

- Generate a simulated dataset with the `ideal.1D` time series in every voxel (with varying amplitude), a baseline, and some noise (with varying standard deviation):
  - ◇ Make a 1 slice dataset from something handy, just to use as a template:
    `3dZcutup -prefix zcut -keep 10 10 epi07+orig`
  - ◇ Calculate a dataset with this geometry (64×64×1) with the `ideal.1D` function intensity and noise variance varying in 8×8 blocks across the image:

```
3dcalc -a 'zcut+orig[0]' \
       -b ideal.1D       \
       -datum float      \
       -prefix sim:time  \
       -expr '100+(int(i/8)+1)*0.25*b+int(j/8)*0.25*gran(0,1)'
```

  - ↪ `3dcalc` is the voxel-by-voxel dataset calculator program
  - ↪ `-a 'zcut+orig[0]'` means to read in the #0 sub-brick of this dataset and call its voxel values by the symbol `a`
  - ↪ `-b ideal.1D` means to read in this time series file and call its values by the symbol `b` (since this has no spatial dimension, each spatial voxel from `b` at a given time index will have the same value, unlike from `a`)
    - ▷ Since the `-a` dataset doesn't have a time axis, the `-b` time series will be used to provide a time axis in the output of 106 points (same length as `ideal.1D`)

- ↪ `-datum float` means to write the output dataset in floating point format (don't need to deal with scaling issues then)
- ↪ `-prefix sim:time` is the prefix part of the output dataset filename
- ↪ `-expr '100+(int(i/8)+1)*0.25*b+int(j/8)*0.25*gran(0,1)'` is the mathematical expression that determines the output at each voxel
  - ▷ Symbols `b`, `i`, `j` are used
  - ▷ Dataset `a` was only read in to act as a template for the output dataset (i.e., to set up its spatial axes and dimensions)
  - ▷ Since there was no `-i` or `-j` option, 3dcalc assigns the x-axis voxel index to the symbol `i` (range=0..63) and the y-axis voxel index to the symbol `j` (range=0..63)
  - ▷ `100` is the baseline
  - ▷ `(int(i/8)+1)` ranges 1..8 across image horizontally, in blocks 8 voxels wide
    - `(int(i/8)+1)*0.25*b` gives signal from 0.25 to 2 times `b`
  - ▷ `int(j/8)` ranges 0..7 across image vertically, in blocks 8 voxels wide
    - `int(j/8)*0.25*gran(0,1)` gives noise increasing down image, in blocks
    - `gran(0,1)`=Gaussian deviate with mean=0, standard deviation=1

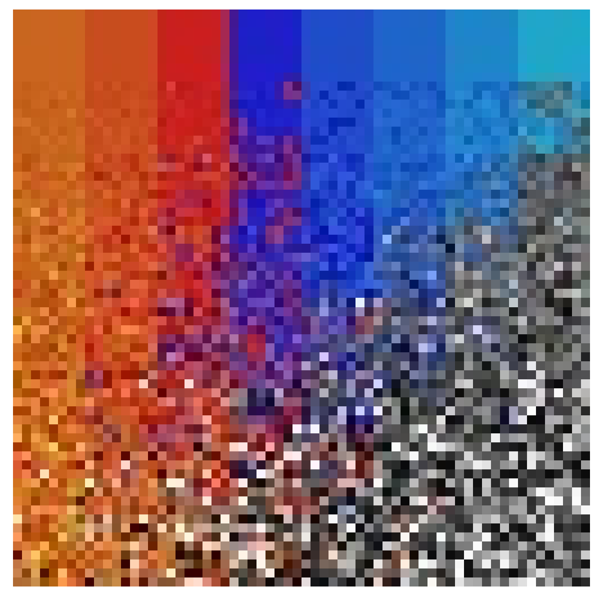
◇ Result is spatially square 3D+time dataset with varying amounts of signal and noise that we can use for testing

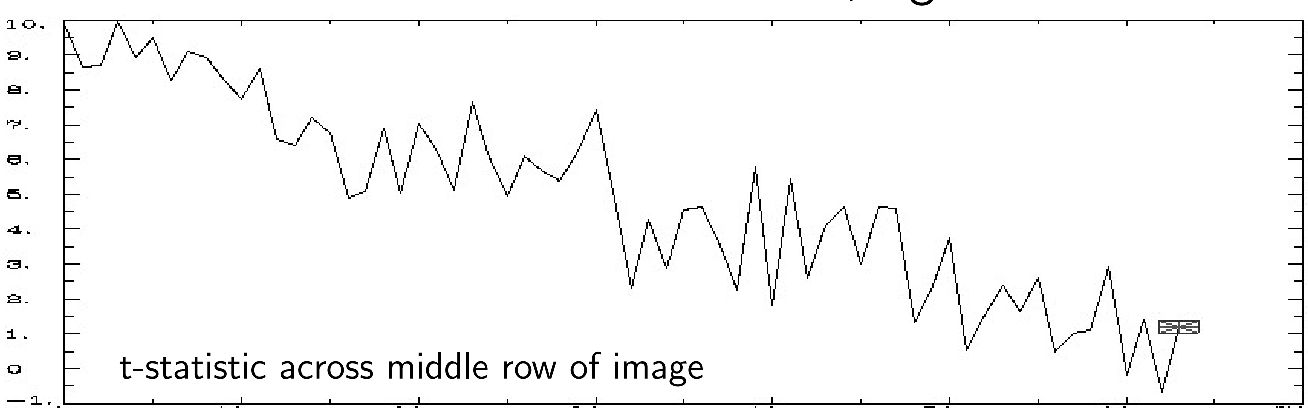
◇ Simple regression analysis with 3dDeconvolve:

```
3dDeconvolve -input sim:time+orig                  \
             -num_stimts 1                         \
             -stim_file  1 ideal.1D                \
             -stim_label 1 Response                \
             -stim_minlag 1 0  -stim_maxlag 1 0    \
             -fitts sim_regress_fitts              \
             -tout -bucket sim_regress_stats
```

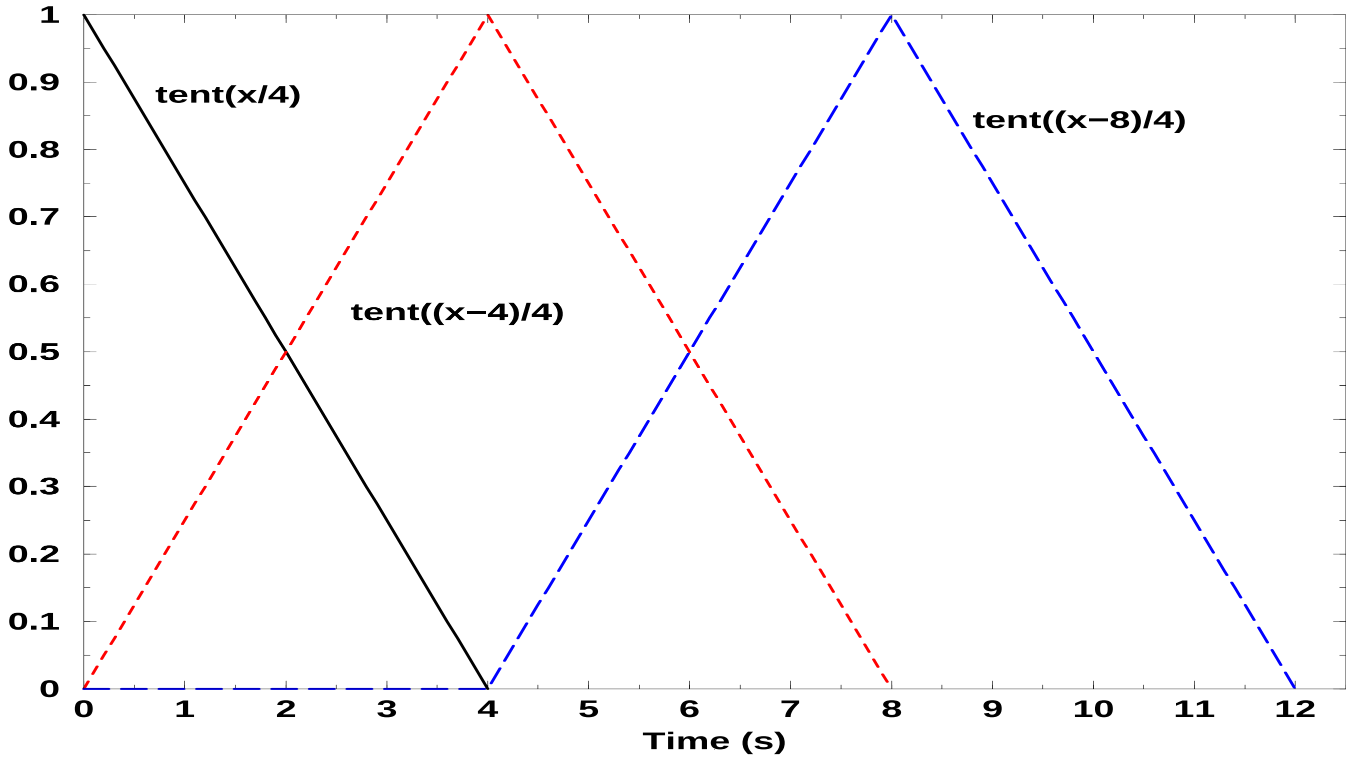
↪ In AFNI:

- Set Function→Func to sub-brick #4 Response[0] Coef
- Set Function→Thr to sub-brick #5 Response[0] t-tstat
- Set Function→** to 1; slider to p ≈1.0-4 (t≈4)
- Set Function→Pos ON; pbar # to 11
- Set Datamode→Misc→Voxel Coords ON
- Turn on See Function, open Axial Image
- Noise increases downwards; signal leftwards





- However, our goal is deconvolution analysis, not simple regression
  - ◇ Assume hemodynamic response to individual stimulus lasts no more than 12 s (depends on type of stimulus)
  - ↪ Actual `waver -GAM` function is significant only from 2..10 s post-stimulus
  - ◇ Model it as the sum of 3 "tent" functions:

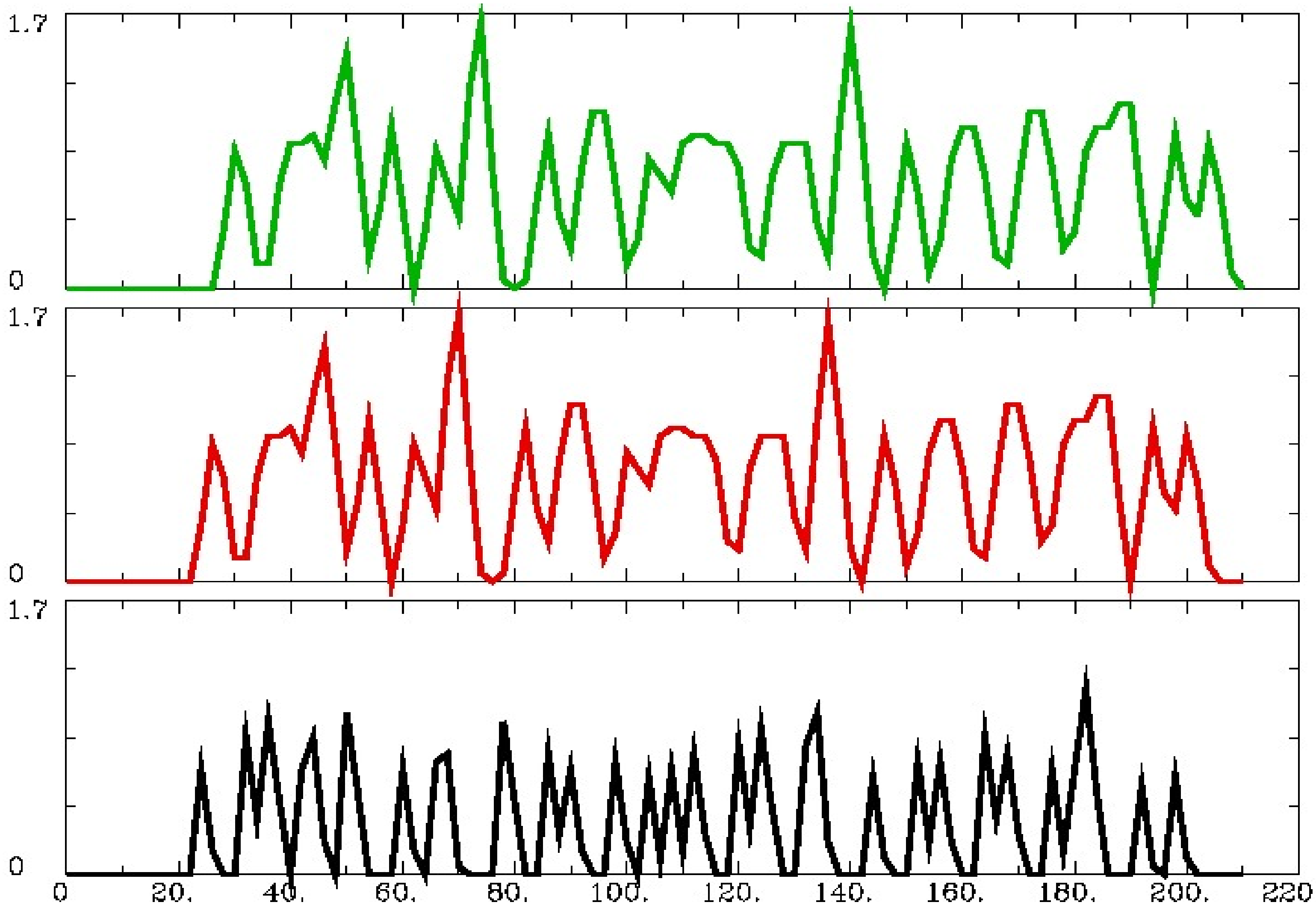


↪ $\mathtt{tent(x)} = \begin{cases} 1-|\mathtt{x}| & \text{if } -1 < \mathtt{x} < 1 \\ 0 & \text{if } |\mathtt{x}| \geq 1 \end{cases}$

↪ Built in function in 3dcalc, waver, etc.

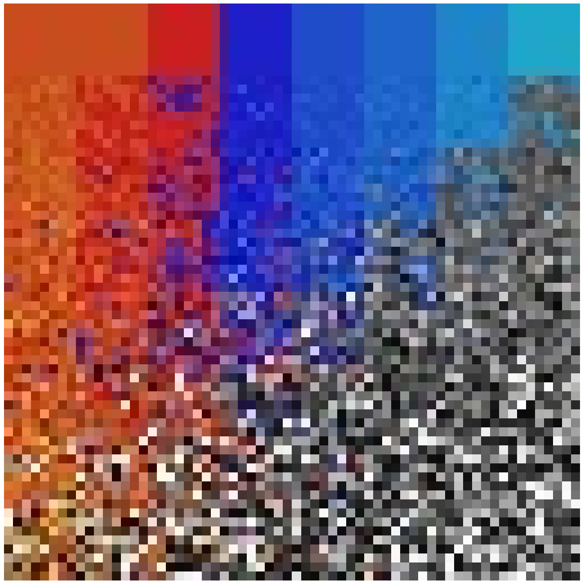
◇ We generate response time series for each of the 3 basis functions `tent(t/4)`, `tent((t-4)/4)`, and `tent((t-8)/8)`:

```
waver -dt 2.0 -EXPR ’tent(t/4)’ -peak 1 -tstim ‘cat tstim.1D‘ > tent0.1D
waver -dt 2.0 -EXPR ’tent((t-4)/4)’ -peak 1 -tstim ‘cat tstim.1D‘ > tent4.1D
waver -dt 2.0 -EXPR ’tent((t-8)/4)’ -peak 1 -tstim ‘cat tstim.1D‘ > tent8.1D
```



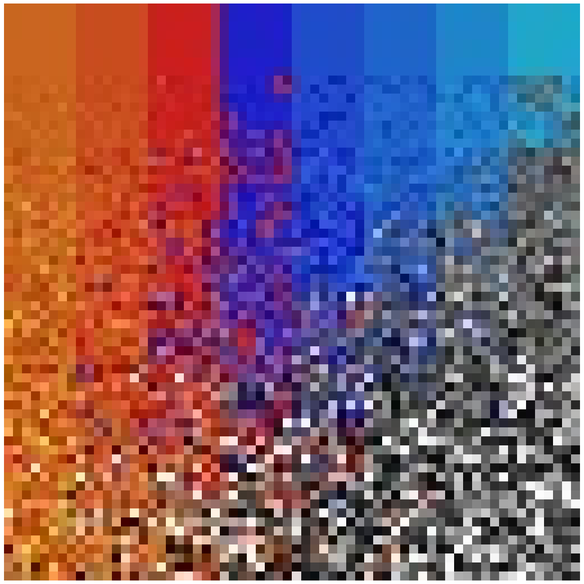
◇ Then we run 3dDeconvolve with these 3 regressors:

```
3dDeconvolve -input sim:time+orig.HEAD          \
             -num_stimts 3                      \
             -stim_file 1 tent0.1D              \
             -stim_file 2 tent4.1D              \
             -stim_file 3 tent8.1D              \
             -stim_label 1 tent0                \
             -stim_label 2 tent4                \
             -stim_label 3 tent8                \
             -stim_minlag 1 0  -stim_maxlag 1 0 \
             -stim_minlag 2 0  -stim_maxlag 2 0 \
             -stim_minlag 3 0  -stim_maxlag 3 0 \
             -fitts sim_decon_fitts             \
             -fout -bucket sim_decon_stats
```



↑Deconvolution output↑

- Function→Func = #4 tent4[0] Coef
- Function→Thr = #8 Full F-stat
- Function→** = 1; slider p ≈1.0-4 (F≈7.8)
- Function→Pos ON; # = 11



↑Regression output (from before)↑

◇ To calculate hemodynamic response function in each voxel, need to combine basis functions with amplitudes from output dataset sim_decon_stats+orig:

```
3dcalc -a 'sim_decon_stats+orig[2]'  \
       -b 'sim_decon_stats+orig[4]'  \
       -c 'sim_decon_stats+orig[6]'  \
       -dt 0.1                       \
       -taxis 121                    \
       -datum float                  \
       -prefix sim:hrf_fit           \
       -expr 'a*tent(t/4)+b*tent((t-4)/4)+c*tent((t-8)/4)'
```

↪ Symbols a, b, c are for 3D (no time) volumes

↪ -dt 0.1 sets TR for manufactured 3D+time dataset to 0.1 s

↪ -taxis 121 sets the number of time points to 121

  ▷ Need this since all input datasets are 3D only — no time axes

↪ Symbol t is time in -expr [default symbol value]

◇ Also make a 1D file with the true gamma deviate hemodynamic response function, for comparison: `waver -GAM -dt 0.1 > gamma.1D`

◇ In AFNI:

↪ Switch Anatomy to sim:hrf_fit; open Axial Graph

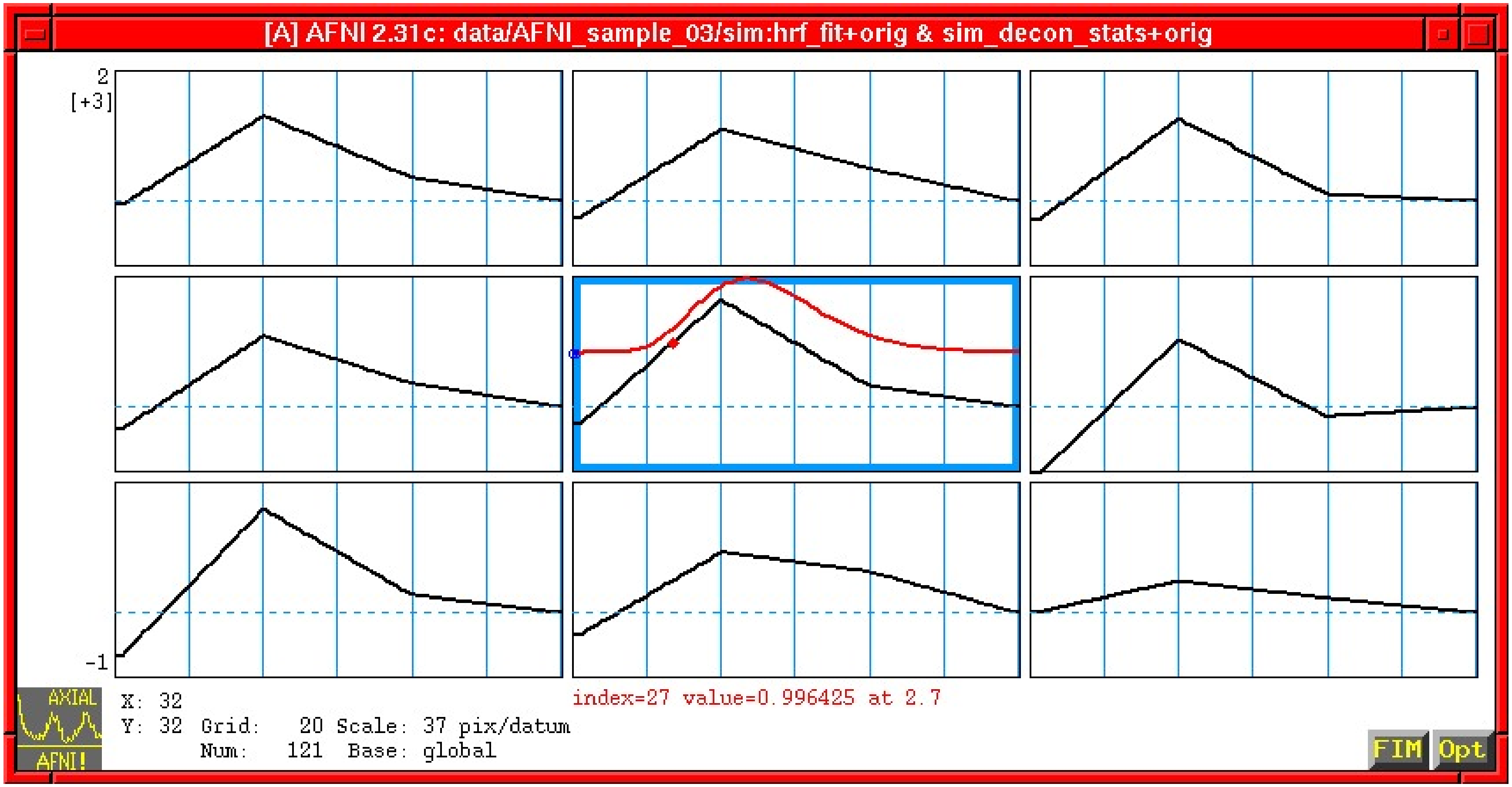
↪ FIM⟶Pick Ideal and choose gamma.1D timeseries

↪ Opt⟶Colors, Etc.⟶Ideal: Use Thick Lines ON

↪ Opt⟶Colors, Etc.⟶Grid color to lt-blue2 (say)

↪ Opt⟶Baseline⟶Global ON; Opt⟶Baseline⟶Set Global to −1

↪ Press h key to get horizontal line at y=0; press a to autoscale



Other 3dDeconvolve Tricks

- Program 3dTshift lets you time shift (interpolate) the slices in a 3D+time dataset to the same time origin
  - ◇ Not necessarily a good idea
- Program 3dvolreg is used to do image registration or realignment (subject of another presentation)
  - ◇ Time shifting can also be done in 3dvolreg
  - ◇ You can also use estimated movement parameters from 3dvolreg as "regressors of no interest" (RONI) in 3dDeconvolve
  - ◇ The idea is to get rid of any residual effects correlated to the subject's movement
  - ◇ However, you don't want to have lags for these RONI, since physical effects of movement on image are immediate
  - ◇ Also, you don't want to include the RONI in the F statistic map, which means you need to use the -stim_base option for these regressors
- Option -nodata can be used to evaluate the regression design to determine if it is well-determined
- Option -input1D can be used to do the regression on a single time series file (instead of on an entire dataset of time series)

- Option -mask restricts the analysis to just a specified set of voxels (for speed)
  - ◇ Program 3dAutomask can be used to automatically make a mask dataset from a 3D+time input dataset: `3dAutomask -prefix Fred_mask Fred_time+orig`
- Program 3dTcat can be used to catenate multiple 3D+time datasets into one big file for input into 3dDeconvolve
  - ◇ Option -concat is then needed to tell 3dDeconvolve where each individual imaging runs starts
  - ↪ So response from stimulus at end of run #1 doesn't "bleed" into run #2
  - ↪ So each run can have its own baseline parameters
- Option -polort lets you specify a higher order (than linear) polynomial to use as the baseline model in the regression
- Option -progress 1000 will print (to screen) intermediate results every 1000 voxels
- Program 3dDeconvolve_f does all calculations in single precision
  ⇒ is about 40% faster than 3dDeconvolve
- If you have a multi-CPU machine with shared memory (SMP), -jobs N option lets you spread computations over N processes
  ⇒ Significant speedup results on dual-CPU Linux machines
- You need to read the 3dDeconvolve manual!

3dDeconvolve Script

- This is an experimental script to help you run 3dDeconvolve with irregular stimulus timing
- It asks you a sequence of questions: for names of regressor files, etc.
- It runs waver to create regressors from stimulus timing
- It allows the use of basis functions to model $h(t)$
- It graphs the regressors from waver
- It runs 3dDeconvolve to fit the model
- It runs 3dcalc to combine basis functions to give $h(t)$ in each voxel
- Currently written in Matlab; plans are to rewrite in a free language (Python?)
- Available on request

Nonlinear Regression to Model a Time Series

- For some applications, a nonlinear model may make most sense
  - ◇ Single event FMRI: where you have only one stimulus in an imaging run, and are not sure of the form the response will take
  - ◇ Pharmaceutical injections; Changing subject's affect by video presentation
- Program 3dNLFIM and plugin NLfit let you model time series in each voxel in an arbitrary way
  - ◇ You provide a "model function" in C that returns the model time series, given a set of parameters
  - ◇ Program/plugin drive the model to find the set of parameters that best fits each data time series
  - ◇ 3DNLfim program produces F-statistic for goodness-of-fit test
  - ◇ NLfit plugin produces fitted time series for graphical (Double Plot) exploration
- Nonlinear constraints let you put arbitrary boundaries on what the fitting model will look for; for example:
  - ◇ Require positive responses
  - ◇ Don't allow shape parameters b and c in $At^b e^{-t/c}$ to be "unreasonable"