From Functional Data to Smooth Functions

Jim Ramsay

- We need a flexible method for constructing functions from noisy discrete data.
- The method should be able to reproduce any feature that interests us in a function, no matter how complicated.
- The computation should be reasonably fast, even when tens or hundreds of thousands of discrete values are available.
- In this talk, we consider the most popular technique, *basis function expansions*.

- We describe two basis function systems in detail:
  - Fourier bases
  - B-spline bases
- as well as some other important systems.
- We also ask about how to estimate derivatives,
- including a bad idea.

Outline

- A basis function system is a set of K known functions $\phi_k(t)$ that are:
  - linearly independent of each other
  - can be extended to include any number K in the system
- A function $x(t)$ is constructed as a linear combination of these basis functions:

$$x(t) = \sum_{k=1}^{K} c_k \phi_k(t)$$

- If vector $\mathbf{c}$ contains the coefficients, and the vector ϕ contains the basis functions, then

$$x(t) = \mathbf{c}'\phi(t).$$

Basis function systems and derivatives

- In principle, computing derivatives is easy:

$$D^m x(t) = \sum_{k=1}^{K} c_k D^m \phi_k(t)$$

- but not all basis functions have derivatives that behave reasonably, or can even be calculated.

The monomial basis

- Polynomials are perhaps the oldest and best known basis function expansion.
- A polynomial is the form

$$x(t) = \sum_{k=1}^{K} c_k t^{k-1}.$$

- The basis functions are the *monomials*: $1, t, t^2, t^3, \ldots$
- Polynomials can work fine for simple problems only requiring $K = 5$ or so, but have severe problems tracking sharp localized features, and can run into computational problems for unequally spaced data.

Polynomials and derivatives

- Derivative estimation is a big problem for polynomials because their derivatives become less and less complex, the higher the order of derivative.
- For a polynomial of degree m, the derivative of order $m + 1$ is zero.
- But in most real–world systems, derivatives become more complex as the order of the derivative increases.

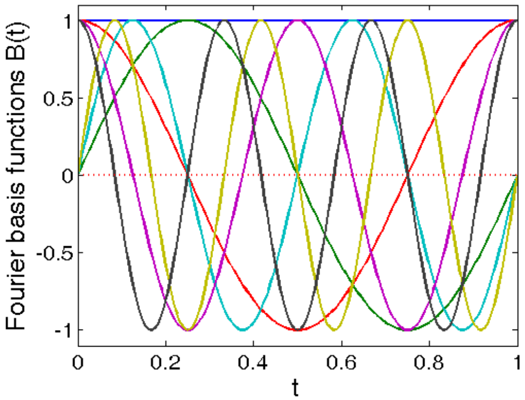
Outline

- The basis functions are sine and cosine functions of increasing frequency:

$$1, \sin(\omega t), \cos(\omega t), \sin(2\omega t), \cos(2\omega(t), \ldots$$

$$\sin(m\omega t), \cos(m\omega(t), \ldots$$

- The constant ω defines the period of oscillation of the first sine/cosine pair. This is $\omega = 2\pi/P$ where P is the period.
- $K = 2M + 1$ where M is the largest number of oscillations in period P that are required.


Fourier basis functions B(t)
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1
t

Advantages of Fourier basis functions

- Fourier bases were the only alternative to monomial bases until the middle of the 20th century.
- They have excellent computational properties, especially if the times of observation are equally spaced.
- They are natural for describing data which are periodic, such as the annual weather data, gait cycle data and so on.
- Their periodicity is a problem, however, for nonperiodic data, such as the growth curves.
- But the Fourier basis is still the first choice in many fields, such as signal analysis, even when the data are not periodic.

Fourier bases and derivatives

- Computing derivatives is easy since

$$D\sin(\omega t) = \omega\cos(\omega t)$$

$$D\cos(\omega t) = -\omega\sin(\omega t)$$

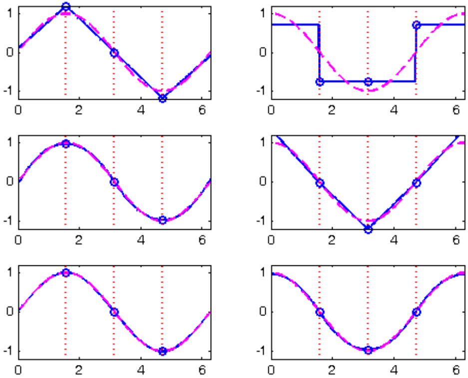
- We say that this system is closed under differentiation; the derivative of a Fourier series expansion is also a Fourier series expansion.
- The Fourier series is infinitely differentiable.

Outline

- Splines are polynomial segments joined end-to-end.
- The segments are constrained to be smooth at the join.
- The values of t at which adjacent segments are joined are called *knots*.
- The *order* m (order = degree + 1) of the polynomial segments and
- the location of the knots define the spline basis system.

An example of spline functions

- The following figure shows splines of three orders, each with three knot values.
- The splines are defined so as to offer the best fit to a sine function, shown in the left panels.
- How well the derivatives of these splines fit the derivative of the sine, the cosine, is shown in the right panels.


1
0
-1
0
2
4
6

Derivatives and splines

- Because splines are constructed from polynomials, computing their derivative at any point between two knots is simple. There, the highest nontrivial order of derivative is $m - 1$ for order m splines.
- At a knot, it is usual to require that the derivatives up to order $m - 2$ also join. That is, the derivative of order $m - 2$ of a spline function is usually *continuous*.
- The most popular choice of order is 4, implying continuous second derivatives. The second derivatives have straight line segments.

Spline functions and degrees of freedom

- How can we quantify the flexibility of a spline function of order m?
- In the usual case, there are $m - 1$ constraints on the adjacent polynomials, corresponding to the requirement that $m - 2$ derivatives plus the function values are required to match at the knot.
- Given the first segment, with m degrees of freedom, this means that we gain one degree of freedom with each knot to the right of the first segment.
- The total number of degrees of freedom is

$$\textit{order } m + \textit{number of interior knots}$$

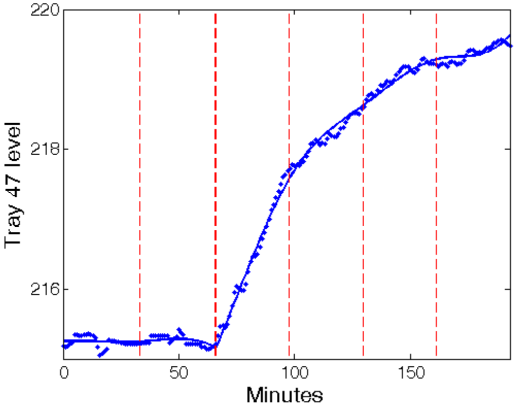
How are knots chosen?

- Knots are often spaced equally.
- But two important rules should be observed in placing knots:
  - Place more knots where you know there is strong curvature, and fewer where the function changes slowly.
  - But be sure that there is at least one data point in any interval.
- Later, we will consider placing a knot at each point of observation.

Spline functions and coincident knots

- Sometimes we need less smoothness at a specific point.
- For example, we will see problems where a function needs to be continuous at a point, but its derivative is discontinuous.
- When multiple knots are placed at the same point, the convention is that a spline loses one derivative for each additional knot.
- An order 4 spline with 3 coincident knots is continuous at that point, but does not have a first derivative.
- An order 4 spline with 4 coincident knots is discontinuous at that point.

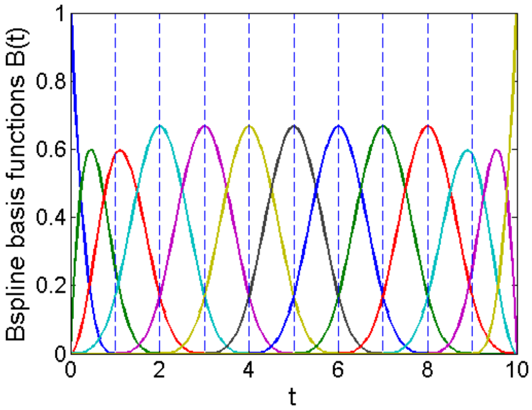
There are three coincident knots at the second location for the refinery data to permit a discontinuous first derivative.



The B-spline basis system

- Any spline function with K degrees of freedom can be expressed as a linear combination of K basis spline functions .
- Among many possibilities, the B-spline system, developed in the 1940's, is the most popular.
- B-spline basis functions are themselves spline functions.
- Any B-spline basis function is positive over at most m adjacent intervals.
- This ensures that computation is fast for even tens of thousands of basis functions.

13 order 4 B-spline basis functions



Outline

- Basis systems can be constructed in many other ways:

  Power Basis: $t^{\lambda_1}, t^{\lambda_2}, t^{\lambda_3}, \ldots$ where the powers are distinct but not necessarily integers or even positive.

  Exponential Basis: $e^{\lambda_1 t}, e^{\lambda_2 t}, e^{\lambda_3 t}, \ldots$ where the λ's are distinct.

Wavelet bases

- A recent development, wavelet bases combine some of the advantages of both Fourier and B-spline bases.
- They are especially good at tracking sharp highly localized features,
- and separating a signal into components which reflect both specific frequencies and specific locations on the *t*-axis.
- Because of their computational efficiency, they are often used for image compression.
- For example, the FBI uses wavelets to store fingerprint information.

The constant basis

Let's not neglect the simplest basis system of all: consisting of a single basis function $\phi_1(t) = 1$. We often need to fit a constant to data.

Empirical basis functions

- We will look at functional *principal components analysis* later.
- This is essentially a method for estimating orthogonal basis functions from functional data that capture as much of the variation as possible given a fixed number of basis functions K.

Designer or customized basis functions

- Later, when we come to differential equation models, we will see how to tailor a basis system to the known characteristics of a set of data.
- Designer bases like these can be much more efficient at representing functional variation.

Outline

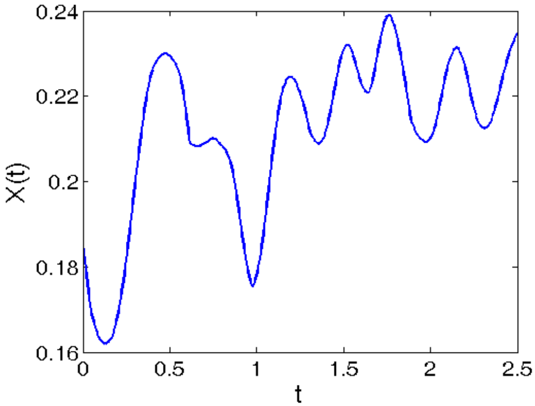
- It is common practice to estimate a derivative by taking the difference between adjacent function values divided by the difference between adjacent time values:

$$\Delta x(t_i) = \frac{x(t_{i+1}) - x(t_i)}{t_{i+1} - t_i}$$

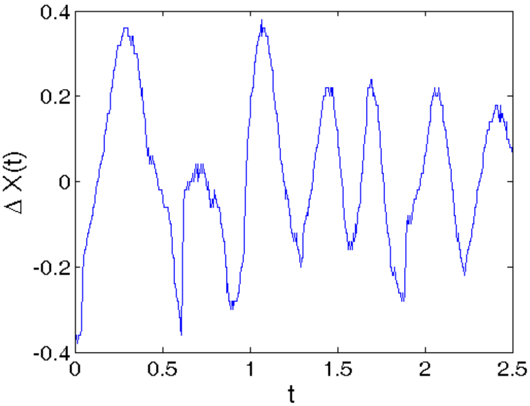
- The second derivative can be estimated by applying this differencing process to the first difference ratios.
- This only works for very smooth functions observed without appreciable error.
- Even the smallest amount of error is greatly magnified by differencing.

- The following displays show what happens when we difference a record of pen position that has a signal-to-noise ratio of 500-to-1, and which is sampled 200 times per second.
- The third order difference ratios are virtually worthless as an estimates of the values of the third derivative, which we will need in later analyses.

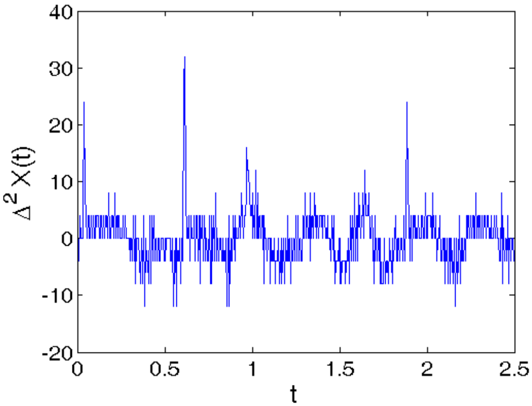
The pen position function



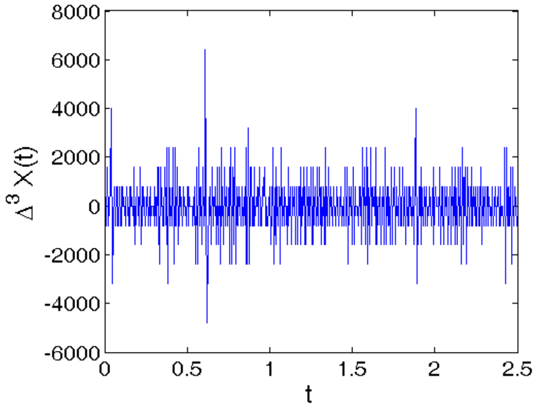
The first difference ratios



The second difference ratios



The third difference ratios



Where we go from here

- Now we need to see how to fit a basis function expansion to noisy data.
- The simplest process is through least squares approximation.
- This is essentially the use of multiple regression analysis, where the covariates are the basis function values corresponding to time sampling points.
- This works reasonably well, but we will see how to do even better later.

Outline

- Controlling smoothness by limiting the number of basis functions is discontinuous; roughness penalties allow continuous control over smoothness.
- We want to be able to define “smooth” in ways that are appropriate to our problems.
  - We may want a smooth derivative rather than just a smooth function.
  - What is smooth in one situation is not smooth in another. Smoothness has to be defined differently for periodic functions, for example.
- We find that roughness penalty smoothing gives better results.
- Roughness penalties are connected to fitting data by a differential equation; they are models for process dynamics.

Outline

We have two competing objectives:

1. Fit the data well; keep bias low.
2. Keep the fit smooth so as to
   - filter out noise
   - get better estimates of derivatives

$$\textit{Mean squared error} = \textit{Bias}^2 + \textit{Sampling Variance}$$

We can often greatly reduce MSE by trading a little bias off against a lot of sampling variance.

Quantifying roughness

- **The classic: curvature in the function**

$$\mathrm{PEN}_2(x) = \int [D^2 x(s)]^2 \, ds \, .$$

$[D^2 x(s)]^2$ measures the *squared curvature* in x at s. This penalty measures *total squared curvature*.

- **Curvature in acceleration:**

$$\mathrm{PEN}_4(x) = \int [D^4 x(s)]^2 \, ds$$

- These two penalties also define what we mean by “smooth”; any function that has zero penalty is “hyper–smooth.” A straight line for the classic, a cubic polynomial for the acceleration penalty.

Harmonic acceleration

- If the process is periodic, it is natural to think of a *constant* $+$ *sinusoid* as “hyper–smooth”.
- This suggests that we use

$$\mathrm{PEN}_H(x) = \int [D^3 x(s) + \omega^2 Dx(s)]^2 \, ds$$

  where $2\pi/\omega$ is the period.
- The functions 1, *sin*(ωt), and cos(ωt) all have zero penalties, as does any linear combination of them.

Some questions to think about

- Writing $Lx(s) = D^3x(s) + \omega^2 Dx(s)$, we have

$$\mathrm{PEN}_H(x) = \int [Lx(s)]^2 \, ds$$

- Can we think of other *differential operators* L that might be useful?
- If we have a small number of "hyper–smooth" functions in mind, can we find a differential operator L that will assign zero penalty to them?
- Can use the data themselves to tell us something about the right differential operator L?

Outline

- Notation:
  - **y** is the n-vector of data y_j to be smoothed.
  - **t** is the n-vector of values of t_j.
  - **W** is a symmetric positive definite weight matrix.
  - $x(\mathbf{t})$ is the n-vector of fitted values, and $x(t)$ has the basis function expansion

$$x(t) = \sum_{k}^{K} c_k \phi_k(t) = \mathbf{c}'\phi(t)$$

- The penalized least squares criterion is

$$\mathrm{PENSSE}_\lambda(x|\mathbf{y}) = [\mathbf{y} - x(\mathbf{t})]'\mathbf{W}[\mathbf{y} - x(\mathbf{t})] + \lambda\,\mathrm{PEN}(x)\ ,$$

How the smoothing parameter works

Smoothing parameter λ controls the amount of roughness.

- As $\lambda \to 0$, roughness matters less and less, and $x(t)$ fits the data better and better.
- As $\lambda \to \infty$, roughness matters more and more, and $x(t)$ becomes more and more "hyper–smooth."
- Our job is to find the right value where we trade enough bias off against sampling variance to minimize mean squared error.

The roughness penalty matrix

- For the classic penalty,

$$
\begin{aligned}
\mathrm{PEN}_2(x) &= \int [D^2 \mathbf{c}' \phi(t)]^2 \, dt \\
&= \mathbf{c}' \int [D^2 \phi(t)][D^2 \phi'(t)] \, dt \, \mathbf{c} \\
&= \mathbf{c}' \mathbf{R} \mathbf{c} \qquad (1)
\end{aligned}
$$

- The order K *roughness penalty matrix* $\mathbf{R}$ is

$$
\mathbf{R} = \int [D^2 \phi(t)][D^2 \phi'(t)] \, dt = \int (D^2 \phi)(D^2 \phi')
$$

- substitute L for D^2 for more general roughness penalties.

The roughness penalized estimates for c and y

- $\mathbf{\Phi}$ is the n by K matrix of basis function values $\phi_k(t_j)$.
- The penalized least squares criterion becomes

$$\texttt{PENSSE}(y|c) = (\mathbf{y} - \mathbf{\Phi c})'\mathbf{W}(\mathbf{y} - \mathbf{\Phi c}) + \lambda \mathbf{c}'\mathbf{Rc} \ .$$

- This is quadratic in **c**, and is minimized by

$$\hat{\mathbf{x}} = (\mathbf{\Phi}'\mathbf{W\Phi} + \lambda \mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{Wy} \ .$$

The smoothing matrix $\mathbf{S}_{\phi,\lambda}$

- The data-fitting vector $\hat{\mathbf{y}} = x(\mathbf{t})$ is

$$\hat{\mathbf{y}} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{W}\mathbf{y} \ ,$$

- Smoothing matrix

$$\mathbf{S}_{\phi,\lambda} = \mathbf{\Phi}(\mathbf{\Phi}'\mathbf{W}\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}'\mathbf{W}$$

maps the data into the fit, and has many useful applications.

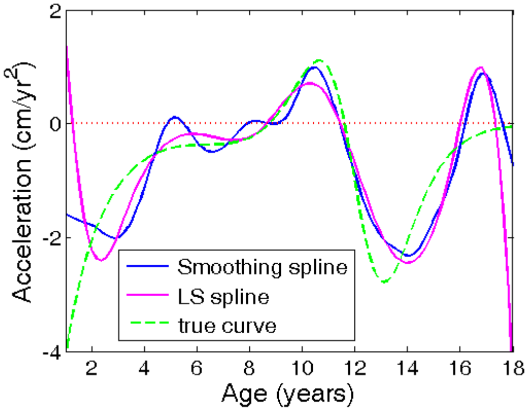
Equivalent degrees of freedom $df(\lambda)$

- It is useful to compare a fit using a roughness penalty to one using a fixed number of basis functions.
- A measure of the "degrees of freedom" in a roughness penalized fit is
$$df(\lambda) = \text{trace}\, \mathbf{S}_{\phi,\lambda}$$
- This corresponds to the number of basis functions K in an un–penalized fit.

Outline

- The term “smoothing spline” has come to mean the following procedure:
  - Use natural or B-spline basis functions.
  - Place a knot at each data point t_j.
  - Use a penalty on D^2x.
- However, we find that
  - We can often achieve the same results by just using a number K of basis functions that is “large” relative to the resolution of the data.
  - We certainly want to be able to play with alternative roughness penalties.
  - Other basis functions systems are also desirable.

Two estimates of an acceleration curve.



Outline

Cross–validation for choosing the smoothing parameter λ

- In cross–validation,we
  - set aside a subset of data, the *validation sample*
  - call the balance of the data the *training sample*
  - fit the model to the training sample
  - assess fit to the validation sample
  - choose the λ value that gives the best fit

- We can also, for a sequence of values of λ,
  - set aside each observation (t_j, y_j) in turn
  - fit the data with the rest of the sample,
  - sum fits to the left out values to get a *cross–validated error sum of squares* $\mathtt{CV}(\lambda)$.
  - select the λ value that minimizes $\mathtt{CV}(\lambda)$.

Generalized cross–validation for choosing the smoothing parameter λ

- Cross-validation is time-consuming, and tends too often to under–smooth the data.
- The generalized cross-validation criterion is

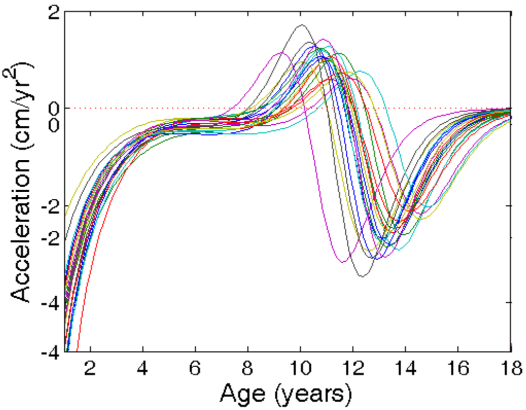
$$GCV(\lambda) = \left(\frac{n}{n - df(\lambda)}\right)\left(\frac{\mathrm{SSE}}{n - df(\lambda)}\right)$$

  where *df* is the equivalent degrees of freedom of the smoothing operator.
- The right factor is just the unbiassed estimate s_e^2 of residual variance familiar in regression analysis.
- The left factor further “discounts” this measure further to allow for the influence of optimizing with respect to λ.

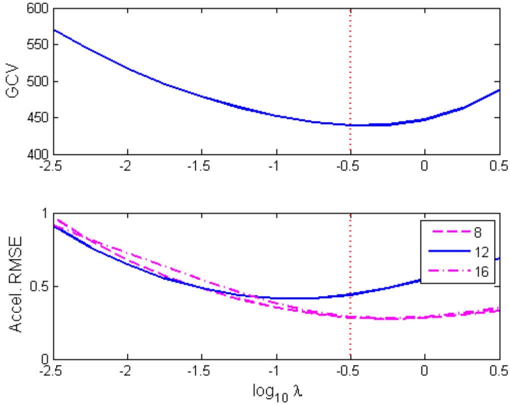
Outline

- How does GCV work in a simulated data example?
- A parametric growth model by Pierre Jolicoeur at the Université de Montréal offers a nice test problem.
- We simulate 1000 samples, each observation being a random sample from realistic Jolicoeur models plus realistic error.
- We smooth using a range of values of λ, and note the value giving the best value of GCV.
- How well do we estimate the Jolicoeur acceleration curves?

20 Jolicoeur acceleration curves



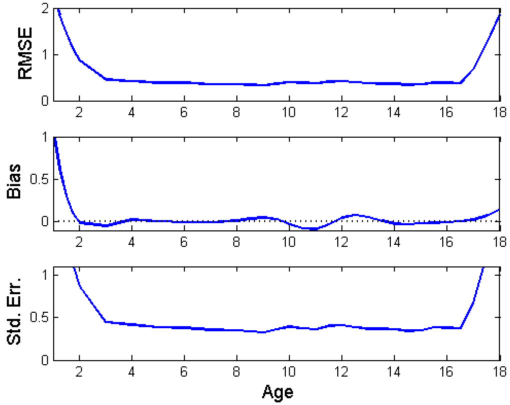
GCV and Root-Mean-Squared-Error



What we see

- In the top panel, GCV favors $\lambda = 0.1$.
- This is about right for optimal MSE for ages 8 and 16, but less smoothing would be better for age 12, in the middle of the pubertal growth spurt.
- One smoothing parameter value does not work best for all ages, but
- The value chosen by GCV certainly does a fine job.

RMSE, Bias, and Standard Error



What we see

- The performance of the spline smoothing estimate deteriorates badly at the extremes.
- The sharp curvature at the pubertal growth spurt also causes some problems.
- Except at the extremes and PGS, the bias is negligible.
- The standard error is about the same as RMSE.
- Would we do better at the extremes if the smooth respected monotonicity?

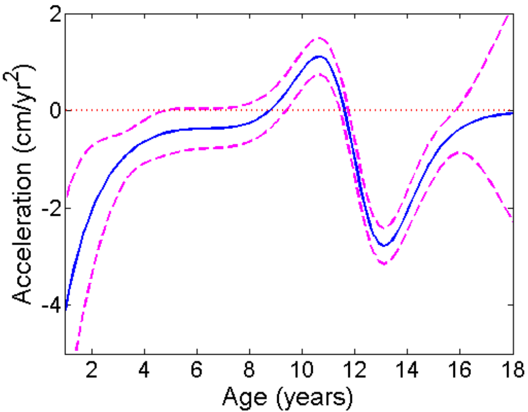
Outline

- Because the mapping from data **y** to the coefficient vector **c** is linear, it is a simple matter to work out the standard error of any linear functional of a curve defined by **c**.
- The variance of a quantity $\rho(x)$ associated with linear mapping **M** from $\hat{\mathbf{c}}$ to $\hat{\rho}(x)$ is

$$\texttt{Var}[\hat{\rho}(x)] = \mathbf{M}\mathbf{S}_{\phi,\lambda}\mathbf{\Sigma}_e\mathbf{S}_{\phi,\lambda}\mathbf{M}'$$

- Simple, that is, if we can get a good estimate of the variance-covariance matrix $\mathbf{\Sigma}_e$ of the residual vector.

95% point–wise confidence limits for growth acceleration



Summary

- Roughness penalization, also called *regularization*, is a flexible and effective way to ensure that an estimated function is “smooth.”
- We can tailor the definition of “smooth” to our needs.
- The roughness penalty idea extends to any type of *functional parameter* that we want to estimate from the data.
- Roughness penalties are one of the main ways in which we exploit the smoothness that we assume in the process generating the data.

Roughness and energy

- "Roughness" is like *energy* in physics
- Roughness requires energy to produce, and smoothness implies limited energy.
- Where we imagine that the amount of energy behind the data is limited, it is natural to assume smoothness.