# Note on Finite Element Methods

Makoto Nakajima, UIUC

January 2006

## 1 Introduction

We study how to approximate a function with one variable, using the so-called finite element method.

## 2 The Problem

**Problem 1**
*We want to approximate a continuous function $f(x)$ which is defined over $[\underline{x}, \overline{x}]$.*

A couple of remarks:

1. It's important that $f(x)$ is a continuous function. All the approximation methods using finite element method or weighted residual method work only for continuous function.
2. It's important that the domain of $f(x)$ is compact. We could extend the method to extrapolate the function, but there are some issues there.

## 3 The Basics

Finite element method means dividing the domain of the function into finite number of sub-intervals and approximating the original function in each of the intervals using some simple functions, like polynomials. The points on the domain which separate the intervals are called knots (or grid points). Usually we use the value (and the derivative) of the function at each knot, to approximate the original function. If we only have the value at each knot, the data is called the Lagrange data. If, in addition, we have derivatives at each knot, the data is called the Hermite data.

There are different kinds of finite element methods depending on what type of function is used to approximate the original function $f(x)$ in each interval.

Of course, the precision becomes higher if we increase the number of knots (equivalently, reducing the size of each interval on average). We can also improve precision without increasing the number of knots by wisely choosing how to locate the knots.

## 4 Piecewise-Linear Interpolation

The most natural choice is to use linear functions and connect the knots. Below is the algorithm:

**Algorithm 1 (Piecewise-Linear Interpolation)**

1. *Pick the number of knots, $n_x$.*

2. *Set the knots* $\{x_i\}_{i=1}^{n_x}$ *in the following way:*

   $$\underline{x} = x_1 < x_2 < ... < x_{n_x-1} < x_{n_x} = \overline{x}$$

3. *Evaluate the function* $f(x)$ *at* $\{x_i\}_{i=1}^{n_x}$*. Call them* $\{y_i\}_{i=1}^{n_x}$*.*

4. *This is all we need to approximate the original function. Suppose a point* $x \in [\underline{x}, \overline{x}]$ *is given. The first step is to find* $j$ *such that:*

   $$x_j \leq x \leq x_{j+1}$$

5. *Then the piecewise linear function evaluated at* $x$ $(y)$ *can be computed using the following formula:*

   $$y = y_j + \frac{(y_{j+1} - y_j)}{(x_{j+1} - x_j)}(x - x_j)$$

A couple of remarks below:

1. Another way to understand the piecewise-linear interpolation is that we give two conditions (value of the function at the two knots) for each interval. Since a linear function has two coefficients, these two conditions are sufficient to pin down two coefficients of the linear function in each interval. This is the basic idea used for all the other finite element methods.

2. The pros of the piecewise-linear approximation is that it's a robust approximation method, in the sense that it preserves monotonicity and concavity of the original function. It doesn't do a lot, but it doesn't do unwanted things.

3. The other benefit is that it's simple and easy to implement.

4. We can improve the accuracy of approximation without increasing the number of knots by putting more knots where the curvature in absolute level is large, and putting less knots where the curvature is low. But this requires some prior knowledge about the function that is approximated.

5. The cons of the method is that the approximated function is not differentiable at knots. In other words, the derivative is not continuous; there are jumps at knots.

6. Related property is that the second derivative is zero except for knots where there is no second derivative.

## 5 Cubic Spline Interpolation

Cubic spline interpolation is using third order polynomials to approximate the function in each interval. First of all, notice that, with $n_x$ knots, we have $n_x - 1$ intervals. For an interval $i$, a third order polynomial takes the following form:

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

We have 4 coefficients for each of the intervals $i = 1, ..., n_x - 1$. Therefore, we have $4(n_x - 1)$ coefficients to be pinned down. If we have the same number of conditions, we can pin down all of the coefficients exactly.

1. If we have the value of the original function $f(x)$ at $n_x$ knots, we already have $n_x$ conditions.
2. Because of the continuity requirement, we have $n_x - 2$ conditions like the following:

$$f_i(x_{i+1}) = f_{i+1}(x_{i+1})$$

3. If we impose differentiability, we can obtain another $n_x - 2$ conditions like the following:

$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$$

4. If we impose twice differentiability, we can obtain another $n_x - 2$ conditions like the following:

$$f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})$$

Summing up, we obtain $4n_x - 6$ conditions. How many more do we need? We need 2 more. The remaining tow can be really arbitrary. Below are the two popular choices:

1. **Natural cubic spline:** Assume that the third derivatives at two end points $x_1$ and $x_{n_x}$ are zero.
2. **"Not-a-knot" cubic spline:** Assume that the function is three times differentiable at $x_2$ and $x_{n_x-1}$. Why the name? Because this condition effectively implies that the spline functions at interval 1 and 2, and $n_x - 2$ and $n_x - 1$ are the same cubic spline functions.

With one of these additional conditions above, we get $4(n_x - 1)$ conditions, exactly enough to pin down the same number of coefficients.

A bit formally, the algorithm looks like the following:

**Algorithm 2 (Cubic Spline Interpolation)**

1. *Pick the number of knots, $n_x$.*
2. *Set the knots $\{x_i\}_{i=1}^{n_x}$ in the following way:*

$$\underline{x} = x_1 < x_2 < ... < x_{n_x-1} < x_{n_x} = \overline{x}$$

3. *Evaluate the function $f(x)$ at $\{x_i\}_{i=1}^{n_x}$. Call them $\{y_i\}_{i=1}^{n_x}$.*
4. *Solve system of $4(n_x - 1)$ equations to obtain $4(n_x - 1)$ coefficients $\{(a_i, b_i, c_i, d_i)\}_{i=1}^{n_x-1}$ which characterize the $(n_x - 1)$ third order polynomials that are used to approximate the original function $f(x)$.*
5. *Suppose we need to evaluate the approximated function at a point $x$. The first step is to find $\bar{i}$ such that*

$$x_{\bar{i}} \leq x \leq x_{\bar{i}+1}$$

6. *Then we can evaluate the cubic spline function at $x$ ($y$) using the following formula:*

$$y = a_{\bar{i}}x^3 + b_{\bar{i}}x^2 + c_{\bar{i}}x + d_{\bar{i}}$$

Below are some remarks:

1. Since solving for the coefficients is a costly (time-consuming) process (we are solving a system of equations), we should solve for the coefficients once and store the coefficients so that we can use the coefficients to evaluate the interpolated function without solving for the coefficients again.
2. The good property is that the function is twice differentiable.
3. A potential problem is that the approximation procedure is does not always preserve monotonicity nor concavity. This might screw up the algorithm, if the algorithm assumes a monotonic concave function.

## 6 Shape-Preserving Spline Interpolation

Shape-preserving spline gives a nice combination of differentiability and shape-preserving (monotonicity and concavity) property. Let's see how to construct shape-preserving spline interpolation function.

First of all, in order to preserve the shape of the original function, we need to know the shape of the original function. In this method, the information on the shape of the original function is obtained from using not only the values but also the derivatives at the knots. In other words, standard implementation of this shape-preserving interpolation uses Hermite data. After we learn how to construct interpolating function using Hermite data, we are going to extend our algorithm to the case where the derivatives are not available (Lagrange data).

Again, let's suppose we have $n_x$ knots, and therefore, $n_x - 1$ intervals. Suppose also that we have:

$$y_i = f(x_i) \qquad \forall i = 1, ..., n_x$$

$$d_i = f'(x_i) \qquad \forall i = 1, ..., n_x$$

Take one representative interval. Because we have both values and derivatives at both end points of the interval, we already have 4 conditions. If we want to preserve concavity, using quadratic function is a good way to go, but the coefficients of a quadratic function are over-identified; we already have 4 conditions, whereas a quadratic function has 3 coefficients. So what can we do?

The way we take is to put a *sub-knot* in each of the intervals. Let's denote them as $\{z_i\}_{i=1}^{n_x-1}$. Obviously, we have:

$$x_i \leq z_i \leq x_{i+1} \qquad \forall i = 1, ..., n_x - 1$$

Let's call the interval $[x_i, z_i]$ as an sub-interval 1 for interval $i$, and the interval $[z_i, x_{i+1}]$ as an sub-interval 2 for interval $i$. We are going to use one quadratic function for each sub-interval. Since a quadratic function is characterized by 3 parameters, we have 6 parameters to be pinned down for each interval (3 for each sub-interval). On the other hand, we have so far 4 conditions for each interval. We need to add 2 more conditions. It's relatively straightforward to come up with the 2 additional conditions, because we want to impose continuity and differentiability at the sub-knots. Now we have 6 coefficients and 6 conditions for each interval.

Denote the quadratic function used in the sub-interval $[x_i, z_i]$ as $f_{i,1}(x)$ and the quadratic function used for the sub-interval $[z_i, x_{i+1}]$ as $f_{i,2}(x)$. Then, we know the followings:

$$f_{i,1}(x_i) = y_i$$

$$f_{i,2}(x_{i+1}) = y_{i+1}$$

$$f'_{i,1}(x_i) = d_i$$

$$f'_{i,2}(x_{i+1}) = d_{i+1}$$

$$f_{i,1}(z_i) = f_{i,2}(z_i)$$

$$f'_{i,1}(z_i) = f'_{i,2}(z_i)$$

Using the first four conditions above, we can construct the two quadratic functions using the following forms:

$$f_{i,1}(x) = a_{i,1}(x - x_i)^2 + d_i(x - x_i) + y_i$$

$$f_{i,2}(x) = a_{i,2}(x - x_{i+1})^2 + d_{i+1}(x - x_{i+1}) + y_{i+1}$$

Using the functional forms above, the remaining two conditions imply:

$$a_{i,1}(z_i - x_i)^2 + d_i(z_i - x_i) + y_i = a_{i,2}(z_i - x_{i+1})^2 + d_{i+1}(z_i - x_{i+1}) + y_{i+1}$$

$$2a_{i,1}(z_i - x_i) + d_i = 2a_{i,2}(z_i - x_{i+1}) + d_{i+1}$$

The two equations above contain three unknowns ($a_{i,1}$, $a_{i,2}$, and $z_i$). By combining them, we can eliminate one of the three (we eliminate $a_{i,1}$). After a bit messy algebra, we can obtain:

$$a_{i,2} = \frac{d_{i+1}z_i + d_{i+1}x_i + d_i x_i - d_i z_i - 2d_{i+1}x_{i+1} + 2(y_{i+1} - y_i)}{2(z_i - x_{i+1})(x_{i+1} - x_i)}$$

We can show that for any choice of $z_i$, we can solve for $a_{1,i}$ and $a_{i,2}$ and thus we can construct two quadratic functions which satisfy the 6 conditions that we listed.

So the problem is how choose $z_i$. In choosing the position of $z_i$, we use the derivatives, and the slope of the linear function connecting the two end points $\overline{d}_i$, which is defined as:

$$\overline{d}_i \equiv \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

What we are going to do is to look at the derivatives at knots, and also $\overline{d}_i$ to determine the shape of the function over the sub-interval, and choose the derivative at $z_i$ (which we denote as $\delta_i$) such that the shape is preserved by our interpolation function. Once $\delta_i$ is chosen, there is a one-to-one relationship between $\delta_i$ and $z_i$. As a preparation, let's obtain the formula for $\delta_i$:

$$\delta_i = 2\overline{d}_i - \frac{d_i(z_i - x_i) + d_{i+1}(x_{i+1} - z_i)}{x_{i+1} - x_i}$$

Now, let's consider different cases one by one:

1. $\overline{d}_i < d_i$ and $\overline{d}_i < d_{i+1}$, or, equivalently, $(\overline{d}_i - d_i)(\overline{d}_i - d_{i+1}) > 0$, then there is an inflection point in the interval. In other words, the function is not concave nor convex in the interval. In this case, we are not restricted in choosing the location of $z_i$. In this case, a suggestion is just to choose the midpoint:

$$z_i = \frac{x_i + x_{i+1}}{2}$$

2. $\overline{d}_i > d_i$ and $\overline{d}_i > d_{i+1}$. This case yields the same condition as above and is treated in the same way as above.

3. $d_i \le \overline{d}_i \le d_{i+1}$. This means that the function is convex in the interval. Notice that no matter if the function is increasing or decreasing, the property is preserved in the interpolating function, because of the convexity of the interpolating function. If the function is not monotone (the sign of $d_i$ is different from the sign of $d_{i+1}$), a local optimum is created during the interpolating algorithm, again because of the convexity.

   In this case, what kind of condition is imposed to the choice of $z_i$? We need to choose $z_i$ such that:

$$d_i \le \delta_i \le d_{i+1}$$

   holds. Then, we need to solve the inequalities for $z_i$. First of all, $d_i \le \delta_i$ plus the formula for $\delta_i$ that we obtained yield:

$$z_i \ge x_{i+1} - \frac{2(x_{i+1} - x_i)(\overline{d}_i - d_i)}{d_{i+1} - d_i}$$

   Next, $\delta_i \le d_{i+1}$ plus the formula for $\delta_i$ that we obtained yield:

$$z_i \le x_i + \frac{2(x_{i+1} - x_i)(d_{i+1} - \overline{d}_i)}{d_{i+1} - d_i}$$

   We need to choose $z_i$ such that the two inequalities are satisfied, in order to preserve the concavity of the original function $f(x)$ over $[x_i, x_{i+1}]$.

   Notice that the first inequality is always satisfied if:

$$\frac{\overline{d}_i - d_i}{d_{i+1} - d_i} \ge \frac{1}{2}$$

   or

$$2(\overline{d}_i - d_i) \ge (d_{i+1} - d_i)$$

   or

$$\overline{d}_i \ge \frac{d_{i+1} + d_i}{2}$$

For the second inequality, it is always satisfied if:

$$\frac{d_{i+1} - \overline{d}_i}{d_{i+1} - d_i} \geq \frac{1}{2}$$

or

$$2(d_{i+1} - \overline{d}_i) \geq (d_{i+1} - d_i)$$

or

$$\overline{d}_i \leq \frac{d_{i+1} + d_i}{2}$$

Comparing the two inequalities that we derived from the original two inequalities, it is easy to see that there are actually only two cases, which are:

(a) If $\overline{d}_i \leq \frac{d_{i+1}+d_i}{2}$, $z_i$ has to be chosen such that:

$$x_{i+1} - \frac{2(x_{i+1} - x_i)(\overline{d}_i - d_i)}{d_{i+1} - d_i} \leq z_i \leq x_{i+1}$$

A suggested choice is to take the midpoint between the two, which is:

$$z_i = x_{i+1} - \frac{(x_{i+1} - x_i)(\overline{d}_i - d_i)}{d_{i+1} - d_i}$$

(b) If $\overline{d}_i > \frac{d_{i+1}+d_i}{2}$, $z_i$ has to be chosen such that:

$$x_i \leq z_i \leq x_i + \frac{2(x_{i+1} - x_i)(d_{i+1} - \overline{d}_i)}{d_{i+1} - d_i}$$

A suggested choice is to take the midpoint between the two, which is:

$$z_i = x_i + \frac{(x_{i+1} - x_i)(d_{i+1} - \overline{d}_i)}{d_{i+1} - d_i}$$

4. $d_i \geq \overline{d}_i \geq d_{i+1}$. This means that the function is concave in the interval. Notice that no matter if the function is increasing or decreasing, the property is preserved in the interpolating function, because of the convexity of the interpolating function. If the function is not monotone (the sign of $d_i$ is different from the sign of $d_{i+1}$), a local optimum is created during the interpolating algorithm, again because of the concavity.

   It's easy to see that all the inequalities are just flipped from the previous case. Therefore, we need to locate $z_i$ following the criteria below:

   (a) If $\overline{d}_i > \frac{d_{i+1}+d_i}{2}$, $z_i$ has to be chosen such that:

$$x_{i+1} - \frac{2(x_{i+1} - x_i)(\overline{d}_i - d_i)}{d_{i+1} - d_i} \leq z_i \leq x_{i+1}$$

   A suggested choice is to take the midpoint between the two, which is:

$$z_i = x_{i+1} - \frac{(x_{i+1} - x_i)(\overline{d}_i - d_i)}{d_{i+1} - d_i}$$

(b) If $\overline{d}_i \leq \frac{d_{i+1}+d_i}{2}$, $z_i$ has to be chosen such that:

$$x_i \leq z_i \leq x_i + \frac{2(x_{i+1}-x_i)(d_{i+1}-\overline{d}_i)}{d_{i+1}-d_i}$$

A suggested choice is to take the midpoint between the two, which is:

$$z_i = x_i + \frac{(x_{i+1}-x_i)(d_{i+1}-\overline{d}_i)}{d_{i+1}-d_i}$$

Now we are ready to summarize the algorithm for shape-preserving quadratic spline interpolation, with Hermite data:

**Algorithm 3 (Shape-Preserving Quadratic Spline Interpolation with Hermite Data)**

1. *Pick the number of knots, $n_x$.*

2. *Set the knots $\{x_i\}_{i=1}^{n_x}$ in the following way:*

$$\underline{x} = x_1 < x_2 < ... < x_{n_x-1} < x_{n_x} = \overline{x}$$

3. *Evaluate the function $f(x)$ at $\{x_i\}_{i=1}^{n_x}$. Call them $\{y_i\}_{i=1}^{n_x}$. We also have derivatives at the knots. Denote them as $\{d_i\}_{i=1}^{n_x}$.*

4. *Compute the slope of the line connecting two adjacent knots and call them $\{\overline{d}_i\}_{i=1}^{n_x-1}$.*

5. *Determine the location of sub-knots for each interval $\{z_i\}_{i=1}^{n_x-1}$ using the following method:*

   *(a) In case $(d_i-\overline{d}_i)(d_{i+1}-\overline{d}_i) \geq 0$:*

$$z_i = \frac{x_i + x_{i+1}}{2}$$

   *(b) In case $|d_i-\overline{d}_i| > |d_{i+1}-\overline{d}_i|$:*

$$z_i = x_i + \frac{(x_{i+1}-x_i)(d_{i+1}-\overline{d}_i)}{d_{i+1}-d_i}$$

   *(c) In case $|d_i-\overline{d}_i| \leq |d_{i+1}-\overline{d}_i|$:*

$$z_i = x_{i+1} + \frac{(x_{i+1}-x_i)(d_i-\overline{d}_i)}{d_{i+1}-d_i}$$

6. *For each interval $i$, once the location of the sub-knot $z_i$ is determined, we only need to use the continuity condition and the differentiability condition at $z_i$ to solve for $a_{i,1}$ and $a_{i,2}$. Once we get these parameters, basically we have found the quadratic functions for the two sub-intervals.*

7. *Suppose we need to evaluate the approximated function at a point $x$. The first step is to find $\bar{i}$ such that*

$$x_{\bar{i}} \leq x \leq x_{\bar{i}+1}$$

8. *The next step is to detect which sub-interval (1 or 2) does $x$ fall into. Specifically, set $\overline{j} = 1$ if $x \leq z_i$ and set $\overline{j} = 2$ otherwise.*

9. *Then we can evaluate the cubic spline function at $x$ ($y$) using one of the following interpolating function (according to $\overline{j}$):*

$$f_{i,1}(x) = a_{i,1}(x - x_i)^2 + d_i(x - x_i) + y_i$$

$$f_{i,2}(x) = a_{i,2}(x - x_{i+1})^2 + d_{i+1}(x - x_{i+1}) + y_{i+1}$$

Finally, how we can construct the derivatives for each knot, if we only have the value of the function at each knot? In other words, how we can modify the algorithm to make it work also for the Lagrange data? One suggestion is to construct the derivatives using the following method:

1. For each $i = 1, ..., n_x - 1$, compute:

$$L_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

$$\overline{d}_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

2. For each $i = 2, ..., n_x - 1$, assign $d_i = 0$ if $\overline{d}_{i-1}\overline{d}_i \leq 0$. In this case, there is a local optimum between $x_{i-1}$ and $x_{i+1}$. So we assume that $x_i$ is the local optimum.

3. Otherwise, the function is monotone between $x_{i-1}$ and $x_{i+1}$. We then basically construct $d_i$ using interpolation. Specifically, use the following formula to compute $d_i$:

$$d_i = \frac{L_{i-1}\overline{d}_{i-1} + L_i\overline{d}_i}{L_{i-1} + L_i}$$

4. For $i = 1$:

$$d_1 = \frac{3\overline{d}_1 - d_2}{2}$$

5. For $i = n_x$

$$d_{n_x} = \frac{3\overline{d}_{n_x-1} - d_{n_x-1}}{2}$$

Below is the summary:

**Algorithm 4 (Shape-Preserving Quadratic Spline Interpolation with Lagrange Data)**

1. *Pick the number of knots, $n_x$.*

2. *Set the knots $\{x_i\}_{i=1}^{n_x}$ in the following way:*

$$\underline{x} = x_1 < x_2 < ... < x_{n_x-1} < x_{n_x} = \overline{x}$$

3. *Evaluate the function $f(x)$ at $\{x_i\}_{i=1}^{n_x}$. Denote them $\{y_i\}_{i=1}^{n_x}$. Compute derivatives at the knots ($\{d_i\}_{i=1}^{n_x}$.) using the method explained above.*

*4. From here, follow step 4 on in Algorithm 3.*

A couple of remarks:

1. It's differentiable and preserves monotonicity and concavity of the original function.
2. Since we use polynomials for approximation, it's easy to obtain analytical derivatives at any point on $[\underline{x}, \overline{x}]$.
3. Be careful with the end conditions when Lagrange data is used (derivatives are constructed, not given), because we are using a sort of extrapolation in obtaining the derivatives at the end knots. For some cases, the derivatives at the end knots do not capture the property of the curvature of the original function and thus might screw up the computation (if the derivative does not increase (or decrease) enough at the end of the state space, we might have difficulty in obtaining internal solutions).