Finite Element Methods

for Structural Engineers

Contents

Articles

References

Article Licenses

Finite element method in structural mechanics

The **Finite element method** (FEM) is a powerful technique originally developed for numerical solution of complex problems in structural mechanics, and it remains the method of choice for complex systems. In the FEM, the structural system is modeled by a set of appropriate **finite elements** interconnected at points called nodes. Elements may have physical properties such as thickness, coefficient of thermal expansion, density, Young's modulus, shear modulus and Poisson's ratio.

History

The origin of finite method can be traced to the matrix analysis of structures[1] where the concept of displacement or stiffness matrix approach was introduced. Finite element concepts were developed based on engineering methods in 50s. The original works such as those by Argyris [2] and Clough [3] became foundation for today's finite element structural analysis methods. Earlier books such as by Zienkiewicz [4] and more recent books such as by Yang [5] give comprehensive summary of developments in finite element structural analysis.

Element properties

- Straight or curved one-dimensional elements with physical properties such as axial, bending, and torsional stiffnesses. This type of elements is suitable for modeling cables, braces, trusses, beams, stiffeners, grids and frames. Straight elements usually have two nodes, one at each end, while curved elements will need at least three nodes including the end-nodes. The elements are positioned at the centroidal axis of the actual members.
- Two-dimensional elements for membrane action (plane stress, plane strain) and/or bending action (plates and shells). They may have a variety of shapes such as flat or curved triangles and quadrilaterals. Nodes are usually placed at the element corners and, if needed for higher accuracy, additional nodes can be placed along the element edges or even inside the element. The elements are positioned at the mid-surface of the actual layer thickness.
- Torus-shaped elements for axisymmetric problems such as thin, thick plates, shells, and solids. The cross-section of the elements are similar to the previously described types: one-dimensional for thin plates and shells, and two-dimensional for solids, and thick plates and shells.
- Three-dimensional elements for modeling 3-D solids such as machine components, dams, embankments or soil masses. Common element shapes include tetrahedrals and hexahedrals. Nodes are placed at the vertexes and possibly in the element faces or within the element.

Element interconnection and displacement

The elements are interconnected only at the exterior nodes, and altogether they should cover the entire domain as accurately as possible. Nodes will have nodal (vector) displacements or degrees of freedom which may include translations, rotations, and for special applications, higher order derivatives of displacements. When the nodes displace, they will *drag* the elements along in a certain manner dictated by the element formulation. In other words, displacements of any points in the element will be interpolated from the nodal displacements, and this is the main reason for the approximate nature of the solution.

Practical considerations

From the application point of view, it is important to model the system such that:

- Symmetry or anti-symmetry conditions are exploited in order to reduce the size of the domain.
- Displacement compatibility, including any required discontinuity, is ensured at the nodes, and preferably, along the element edges as well, particularly when adjacent elements are of different types, material or thickness. Compatibility of displacements of many nodes can usually be imposed via constraint relations—When such a feature is not available in the software package, a physical model that imposes the constraints may be used instead.
- Elements' behaviours capture the dominant actions of the actual system, both locally and globally.
- The element mesh is sufficiently fine in order to have acceptable accuracy. To assess accuracy, the mesh is refined until the important results shows little change. For higher accuracy, the aspect ratio of the elements should be as close to unity as possible, and smaller elements are used over the parts of higher stress gradient.
- Proper support constraints are imposed with special attention paid to nodes on symmetry axes.

Large scale commercial software packages often provide facilities for generating the mesh, graphical display of input and output, which greatly facilitate the verification of both input data and interpretation of the results.

Theoretical overview of FEM-Displacement Formulation: From elements to system to solution

While the theory of FEM can be presented in different perspectives or emphases, its development for structural analysis follows the more traditional approach via the virtual work principle or the minimum total potential energy principle. The virtual work principle approach is more general as it is applicable to both linear and non-linear material behaviours.

The principle of virtual displacements for the structural system expresses the mathematical identity of external and internal virtual work:

$$\text{External virtual work} = \int_V \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma}\, dV \qquad (1)$$

The virtual internal work in the right-hand-side of the above equation may be found by summing the virtual work in the individual elements—This is the crucial step where we will need displacement functions written only for the small domain rather than over the entire system. As shown in the subsequent sections, Eq.(1) leads to the following governing equilibrium equation for the system:

$$\mathbf{R} = \mathbf{K}\mathbf{r} + \mathbf{R}^o \qquad (2)$$

where

$\mathbf{R}$ = vector of nodal forces, representing external forces applied to the system's nodes.

$\mathbf{r}$ = vector of system's nodal displacements, which will, by interpolation, yield displacements at any point of the finite element mesh.

$\mathbf{R}^o$ = vector of equivalent nodal forces, representing all external effects other than the nodal forces which are already included in the preceding nodal force vector **R**. These external effects may include distributed or concentrated surface forces, body forces, thermal effects, initial stresses and strains.

$\mathbf{K}$ = system stiffness matrix, which will be established by *assembling* the *elements' stiffness matrices* : $\mathbf{k}^e$.

Once the supports' constraints are accounted for, the nodal displacements are found by solving the system of linear equations (2), symbolically:

$$\mathbf{r} = \mathbf{K}^{-1}(\mathbf{R} - \mathbf{R}^o) \qquad (3)$$

Subsequently, the strains and stresses in individual elements may be found as follows:

$$\epsilon = \mathbf{Bq} \qquad (4)$$

$$\sigma = \mathbf{E}(\epsilon - \epsilon^o) + \sigma^o = \mathbf{E}(\mathbf{Bq} - \epsilon^o) + \sigma^o \qquad (5)$$

where

$\mathbf{q}$ = vector of element's nodal displacements--a subset of the system displacement vector **r** that pertains to the element under consideration.

$\mathbf{B}$ = strain-displacement matrix that transforms nodal displacements **q** to strains at any point in the element.

$\mathbf{E}$ = elasticity matrix that transforms effective strains to stresses at any point in the element.

ϵ^o = vector of initial strains in the element.

σ^o = vector of initial stresses in the element.

By applying the virtual work equation (1) to the system, we can establish the element matrices $\mathbf{B}$, $\mathbf{k}^e$ as well as the technique of assembling the system matrices $\mathbf{R}^o$ and $\mathbf{K}$. Other matrices such as ϵ^o, σ^o, $\mathbf{R}$ and $\mathbf{E}$ can be directly set up from data input.

Interpolation or shape functions

Let $\mathbf{q}$ be the vector of nodal displacements of a typical element. The displacements at any point of the element may be found by interpolation functions as, symbolically:

$$\mathbf{u} = \mathbf{Nq} \qquad (6)$$

where

$\mathbf{u}$ = vector of displacements at any point {x,y,z} of the element.

$\mathbf{N}$ = matrix of *shape functions* serving as interpolation functions.

Equation (6) gives rise to other quantities of great interest:

- Virtual displacements consistent with virtual nodal displacements: $\delta\mathbf{u} = \mathbf{N}\delta\mathbf{q}$ (6b)
- Strains in the elements: $\epsilon = \mathbf{Du} = \mathbf{DNq}$ (7)

  where $\mathbf{D}$ = matrix of differential operators that convert displacements to strains using linear elasticity theory. Eq.(7) shows that matrix **B** in (4) is

  $$\mathbf{B} = \mathbf{DN} \qquad (8)$$
- Virtual strains consistent with element's virtual nodal displacements: $\delta\boldsymbol{\epsilon} = \mathbf{B}\delta\mathbf{q}$ (9)

Internal virtual work in a typical element

For a typical element of volume V^e, the internal virtual work due to virtual displacements is obtained by substitution of (5) and (9) into (1):

$$\text{Internal virtual work} = \int_{V^e} \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma}\, dV^e = \delta\, \mathbf{q}^T \int_{V^e} \mathbf{B}^T \left\{\mathbf{E}(\mathbf{Bq} - \epsilon^o) + \sigma^o\right\} dV^e \qquad (10)$$

Element matrices

Primarily for the convenience of reference, the following matrices pertaining to a typical elements may now be defined:

Element stiffness matrix $\mathbf{k}^e = \int_{V^e} \mathbf{B}^T\mathbf{EB}\, dV^e$ (11)

Equivalent element load vector $\mathbf{Q}^{oe} = \int_{V^e} -\mathbf{B}^T\left(\mathbf{E}\epsilon^o - \sigma^o\right) dV^e$ (12)

These matrices are usually evaluated numerically using Gaussian quadrature for numerical integration. Their use simplifies (10) to the following:

$$\text{Internal virtual work} = \delta\ \mathbf{q}^T\left(\mathbf{k}^e\mathbf{q} + \mathbf{Q}^{oe}\right) \qquad (13)$$

Element virtual work in terms of system nodal displacements

Since the nodal displacement vector **q** is a subset of the system nodal displacements **r** (for compatibility with adjacent elements), we can replace **q** with **r** by expanding the size of the element matrices with new columns and rows of zeros:

$$\text{Internal virtual work} = \delta\ \mathbf{r}^T\left(\mathbf{k}^e\mathbf{r} + \mathbf{Q}^{oe}\right) \qquad (14)$$

where, for simplicity, we use the same symbols for the element matrices, which now have expanded size as well as suitably rearranged rows and columns.

System virtual work

Summing the internal virtual work (14) for all elements gives the right-hand-side of (1):

$$\text{System internal virtual work} = \sum_e \delta\ \mathbf{r}^T\left(\mathbf{k}^e\mathbf{r}+\mathbf{Q}^{oe}\right) = \delta\ \mathbf{r}^T\Big(\sum_e \mathbf{k}^e\Big)\mathbf{r}+\delta\ \mathbf{r}^T\sum_e \mathbf{Q}^{oe} \qquad (15)$$

Considering now the left-hand-side of (1), the system external virtual work consists of:

- The work done by the nodal forces **R**: $\delta\ \mathbf{r}^T\mathbf{R} \qquad (16)$
- The work done by external forces $\mathbf{T}^e$ on the part $\mathbf{S}^e$ of the elements' edges or surfaces, and by the body forces $\mathbf{f}^e$

  $$\sum_e \int_{S^e} \delta\ \mathbf{u}^T\mathbf{T}^e\, dS^e + \sum_e \int_{V^e} \delta\ \mathbf{u}^T\mathbf{f}^e\, dV^e$$

  Substitution of (6b) gives:

  $$\delta\ \mathbf{q}^T\sum_e \int_{S^e} \mathbf{N}^T\mathbf{T}^e\, dS^e + \delta\ \mathbf{q}^T\sum_e \int_{V^e} \mathbf{N}^T\mathbf{f}^e\, dV^e$$

  or $-\delta\ \mathbf{q}^T\sum_e(\mathbf{Q}^{te} + \mathbf{Q}^{fe}) \qquad (17a)$

  where we have introduced additional element's matrices defined below:

  $$\mathbf{Q}^{te} = -\int_{S^e} \mathbf{N}^T\mathbf{T}^e\, dS^e \qquad (18a)$$

  $$\mathbf{Q}^{fe} = -\int_{V^e} \mathbf{N}^T\mathbf{f}^e\, dV^e \qquad (18b)$$

  Again, numerical integration is convenient for their evaluation. A similar replacement of **q** in (17a) with **r** gives, after rearranging and expanding the vectors $\mathbf{Q}^{te}, \mathbf{Q}^{fe}$:

  $$-\delta\ \mathbf{r}^T\sum_e(\mathbf{Q}^{te} + \mathbf{Q}^{fe}) \qquad (17b)$$

Assembly of system matrices

Adding (16), (17b) and equating the sum to (15) gives:

$$\delta\ \mathbf{r}^T\mathbf{R} - \delta\ \mathbf{r}^T\sum_e(\mathbf{Q}^{te}+\mathbf{Q}^{fe}) = \delta\ \mathbf{r}^T\Big(\sum_e \mathbf{k}^e\Big)\mathbf{r} + \delta\ \mathbf{r}^T\sum_e \mathbf{Q}^{oe}$$

Since the virtual displacements $\delta\ \mathbf{r}$ are arbitrary, the preceding equality reduces to:

$$\mathbf{R} = \Big(\sum_e \mathbf{k}^e\Big)\mathbf{r} + \sum_e \left(\mathbf{Q}^{oe}+\mathbf{Q}^{te}+\mathbf{Q}^{fe}\right)$$

Comparison with (2) shows that:

- The system stiffness matrix is obtained by summing the elements' stiffness matrices:

$$\mathbf{K} = \sum_e \mathbf{k}^e$$

- The vector of equivalent nodal forces is obtained by summing the elements' load vectors:

$$\mathbf{R}^o = \sum_e \left(\mathbf{Q}^{oe}+\mathbf{Q}^{te}+\mathbf{Q}^{fe}\right)$$

In practice, the element matrices are neither expanded nor rearranged. Instead, the system stiffness matrix $\mathbf{K}$ is assembled by adding individual coefficients k^e_{ij} to K_{kl} where the subscripts ij, kl mean that the element's nodal displacements q^e_i, q^e_j match respectively with the system's nodal displacements r_k, r_l. Similarly, $\mathbf{R}^o$ is assembled by adding individual coefficients Q^e_i to R^o_k where q^e_i matches r_k. This direct addition of k^e_{ij} into K_{kl} gives the procedure the name *Direct Stiffness Method*.

References

[1] Matrix Analysis Of Framed Structures, 3rd Edition by Jr. William Weaver, James M. Gere, Springer-Verlag New York, LLC, ISBN 9780412078613, 1966

[2] Argyris, J.H and Kelsey, S. Energy theorems and Structural Analysis Butterworth Scientific publications, London

[3] Clough, R.W, "The Finite Element in Plane Stress Analysis." Proceedings, 2nd ASCE Conference on Electronic Computations, Pittsburgh, Sep 1960

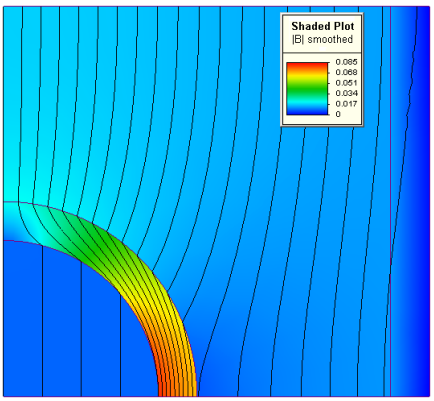
[4] The Finite Element Method for Solid and Structural Mechanics, Zienkiewicz O. C and Taylor R L ISBN 978-0-7506-6321-2 1967, McGraw Hill, New York

[5] Finite Element Structural Analysis , T.Y Yang, Prentice-Hall, Inc, Englewood, NJ, 1986

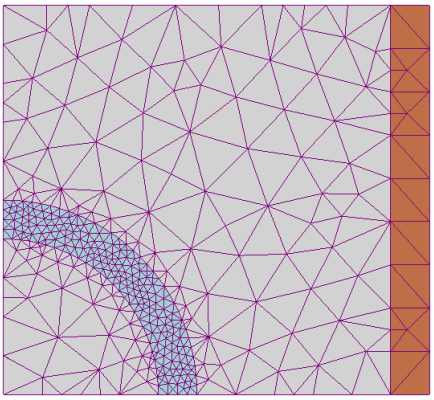
Finite element method

The **finite element method (FEM)** (its practical application often known as **finite element analysis** (FEA)) is a numerical technique for finding approximate solutions of partial differential equations (PDE) as well as of integral equations. The solution approach is based either on eliminating the differential equation completely (steady state problems), or rendering the PDE into an approximating system of ordinary differential equations, which are then numerically integrated using standard techniques such as Euler's method, Runge-Kutta, etc.

In solving partial differential equations, the primary challenge is to create an equation that approximates the equation to be studied, but is numerically stable, meaning that errors in the input and intermediate calculations do not accumulate and cause the resulting output to be meaningless. There are many ways of doing this, all with advantages and disadvantages. The Finite Element Method is a good choice for solving partial differential equations over complicated domains (like cars and oil pipelines), when the domain changes (as during a solid state reaction with a moving boundary), when the desired precision varies over the entire domain, or when the solution lacks smoothness. For instance, in a frontal crash simulation it is possible to increase prediction accuracy in "important" areas like the front of the car and reduce it in its rear (thus reducing cost of the simulation). Another example would be in Numerical weather prediction, where it is more important to have accurate predictions over developing highly-nonlinear phenomena (such as tropical cyclones in the atmosphere, or eddies in the ocean) rather than relatively calm areas.



2D FEM solution for a magnetostatic configuration (lines denote the direction and colour the magnitude of calculated flux density)



2D mesh for the image above (mesh is denser around the object of interest)

History

The finite element method originated from the need for solving complex elasticity and structural analysis problems in civil and aeronautical engineering. Its development can be traced back to the work by Alexander Hrennikoff (1941) and Richard Courant[1] (1942). While the approaches used by these pioneers are different, they share one essential characteristic: mesh discretization of a continuous domain into a set of discrete sub-domains, usually called elements. Starting in 1947, Olgierd Zienkiewicz from Imperial College gathered those methods together into what would be called the Finite Element Method, building the pioneering mathematical formalism of the method.[2]

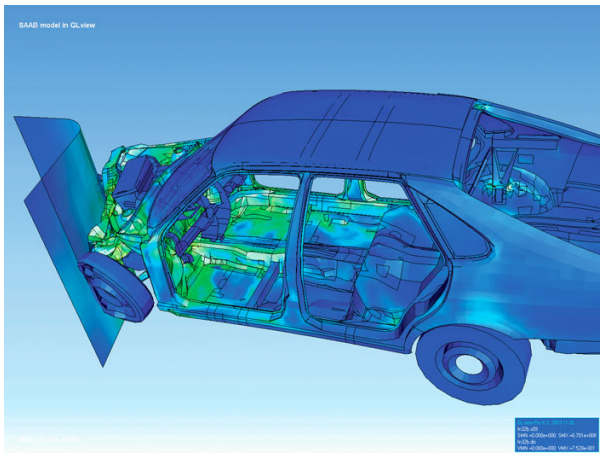
Hrennikoff's work discretizes the domain by using a lattice analogy, while Courant's approach divides the domain into finite triangular subregions to solve second order elliptic partial differential equations (PDEs) that arise from the problem of torsion of a cylinder. Courant's contribution was evolutionary, drawing on a large body of earlier results for PDEs developed by Rayleigh, Ritz, and Galerkin.

Development of the finite element method began in earnest in the middle to late 1950s for airframe and structural analysis[3] and gathered momentum at the University of Stuttgart through the work of John Argyris and at Berkeley through the work of Ray W. Clough in the 1960s for use in civil engineering. By late 1950s, the key concepts of stiffness matrix and element assembly existed essentially in the form used today. NASA issued a request for proposals for the development of the finite element software NASTRAN in 1965. The method was again provided

with a rigorous mathematical foundation in 1973 with the publication of Strang and Fix's *An Analysis of The Finite Element Method*,[4] and has since been generalized into a branch of applied mathematics for numerical modeling of physical systems in a wide variety of engineering disciplines, e.g., electromagnetism, thanks to Peter P. Silvester[5] [5] and fluid dynamics.

Application

A variety of specializations under the umbrella of the mechanical engineering discipline (such as aeronautical, biomechanical, and automotive industries) commonly use integrated FEM in design and development of their products. Several modern FEM packages include specific components such as thermal, electromagnetic, fluid, and structural working environments. In a structural simulation, FEM helps tremendously in producing stiffness and strength visualizations and also in minimizing weight, materials, and costs.



Visualization of how a car deforms in an asymmetrical crash using finite element analysis.[6]

FEM allows detailed visualization of where structures bend or twist, and indicates the distribution of stresses and displacements. FEM software provides a wide range of simulation options for controlling the complexity of both modeling and analysis of a system. Similarly, the desired level of accuracy required and associated computational time requirements can be managed simultaneously to address most engineering applications. FEM allows entire designs to be constructed, refined, and optimized before the design is manufactured.

This powerful design tool has significantly improved both the standard of engineering designs and the methodology of the design process in many industrial applications.[7] The introduction of FEM has substantially decreased the time to take products from concept to the production line.[7] It is primarily through improved initial prototype designs using FEM that testing and development have been accelerated.[8] In summary, benefits of FEM include increased accuracy, enhanced design and better insight into critical design parameters, virtual prototyping, fewer hardware prototypes, a faster and less expensive design cycle, increased productivity, and increased revenue.[7]

Technical discussion

We will illustrate the finite element method using two sample problems from which the general method can be extrapolated. It is assumed that the reader is familiar with calculus and linear algebra.

P1 is a **one-dimensional** problem

$$\text{P1} \ : \begin{cases} u''(x) = f(x) \text{ in } (0,1), \\ u(0) = u(1) = 0, \end{cases}$$

where f is given, u is an unknown function of x, and u'' is the second derivative of u with respect to x.

The **two-dimensional** sample problem is the Dirichlet problem

$$\text{P2} \ : \begin{cases} u_{xx}(x,y) + u_{yy}(x,y) = f(x,y) & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where Ω is a connected open region in the (x, y) plane whose boundary $\partial\Omega$ is "nice" (e.g., a smooth manifold or a polygon), and u_{xx} and u_{yy} denote the second derivatives with respect to x and y, respectively.

The problem P1 can be solved "directly" by computing antiderivatives. However, this method of solving the boundary value problem works only when there is only one spatial dimension and does not generalize to higher-dimensional problems or to problems like $u + u'' = f$. For this reason, we will develop the finite element method for P1 and outline its generalization to P2.

Our explanation will proceed in two steps, which mirror two essential steps one must take to solve a boundary value problem (BVP) using the FEM.

- In the first step, one rephrases the original BVP in its weak form. Little to no computation is usually required for this step. The transformation is done by hand on paper.
- The second step is the discretization, where the weak form is discretized in a finite dimensional space.

After this second step, we have concrete formulae for a large but finite dimensional linear problem whose solution will approximately solve the original BVP. This finite dimensional problem is then implemented on a computer.

Weak formulation

The first step is to convert P1 and P2 into their equivalents weak formulation. If u solves P1, then for any smooth function v that satisfies the displacement boundary conditions, i.e. $v = 0$ at $x = 0$ and $x = 1$, we have

(1) $$\int_0^1 f(x)v(x)\,dx = \int_0^1 u''(x)v(x)\,dx.$$

Conversely, if u with $u(0) = u(1) = 0$ satisfies (1) for every smooth function $v(x)$ then one may show that this u will solve P1. The proof is easier for twice continuously differentiable u (mean value theorem), but may be proved in a distributional sense as well.

By using integration by parts on the right-hand-side of (1), we obtain

(2) $$\begin{aligned}\int_0^1 f(x)v(x)\,dx &= \int_0^1 u''(x)v(x)\,dx \\ &= u'(x)v(x)|_0^1 - \int_0^1 u'(x)v'(x)\,dx \\ &= -\int_0^1 u'(x)v'(x)\,dx = -\phi(u,v).\end{aligned}$$

where we have used the assumption that $v(0) = v(1) = 0$.

A proof outline of existence and uniqueness of the solution

We can loosely think of $H_0^1(0,1)$ to be the absolutely continuous functions of $(0,1)$ that are 0 at $x = 0$ and $x = 1$ (see Sobolev spaces). Such functions are (weakly) "once differentiable" and it turns out that the symmetric bilinear map ϕ then defines an inner product which turns $H_0^1(0,1)$ into a Hilbert space (a detailed proof is nontrivial). On the other hand, the left-hand-side $\int_0^1 f(x)v(x)dx$ is also an inner product, this time on the Lp space $L^2(0,1)$. An application of the Riesz representation theorem for Hilbert spaces shows that there is a unique u solving (2) and therefore P1. This solution is a-priori only a member of $H_0^1(0,1)$, but using elliptic regularity, will be smooth if f is.

The weak form of P2

If we integrate by parts using a form of Green's identities, we see that if u solves P2, then for any v,

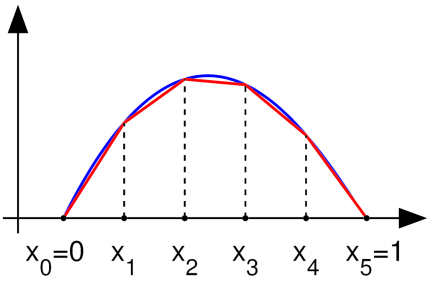
$$\int_\Omega fv\,ds = -\int_\Omega \nabla u \cdot \nabla v\,ds = -\phi(u,v),$$

where ∇ denotes the gradient and $\cdot$ denotes the dot product in the two-dimensional plane. Once more ϕ can be turned into an inner product on a suitable space $H_0^1(\Omega)$ of "once differentiable" functions of Ω that are zero on $\partial\Omega$. We have also assumed that $v \in H_0^1(\Omega)$ (see Sobolev spaces). Existence and uniqueness of the solution can also be shown.

Discretization

The basic idea is to replace the infinite dimensional linear problem:

Find $u \in H_0^1$ such that



A function in H_0^1, with zero values at the endpoints (blue), and a piecewise linear approximation (red).

$$\forall v \in H_0^1,\ -\phi(u,v) = \int fv$$

with a finite dimensional version:

(3) Find $u \in V$ such that

$$\forall v \in V,\ -\phi(u,v) = \int fv$$

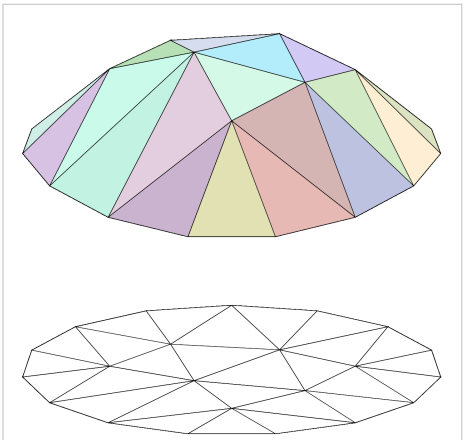
where V is a finite dimensional subspace of H_0^1. There are many possible choices for V (one possibility leads to the spectral method). However, for the finite element method we take V to be a space of piecewise polynomial functions.

For problem P1, we take the interval $(0,1)$, choose n values of x with $0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = 1$ and we define V by

$$V = \{v : [0,1] \to \mathbb{R} \ : v \text{ is continuous, } v|_{[x_k, x_{k+1}]} \text{ is linear for } k = 0, \ldots, n, \text{ and } v(0) = v(1) = 0\}$$

where we define $x_0 = 0$ and $x_{n+1} = 1$. Observe that functions in V are not differentiable according to the elementary definition of calculus. Indeed, if $v \in V$ then the derivative is typically not defined at any $x = x_k$, $k = 1, \ldots, n$. However, the derivative exists at every other value of x and one can use this derivative for the purpose of integration by parts.

For problem P2, we need V to be a set of functions of Ω. In the figure on the right, we have illustrated a triangulation of a 15 sided polygonal region Ω in the plane (below), and a piecewise linear function (above, in color) of this polygon which is linear on each triangle of the triangulation; the space V would consist of functions that are linear on each triangle of the chosen triangulation.

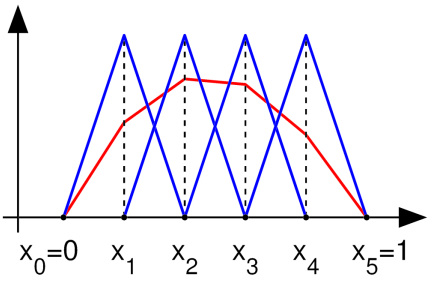


A piecewise linear function in two dimensions.

One often reads V_h instead of V in the literature. The reason is that one hopes that as the underlying triangular grid becomes finer and finer, the solution of the discrete problem (3) will in some sense converge to the solution of the original boundary value problem P2. The triangulation is then indexed by a real valued parameter $h > 0$ which one takes to be very small. This parameter will be related to the size of the largest or average triangle in the triangulation. As we refine the triangulation, the space of piecewise linear functions V must also change with h, hence the notation V_h. Since we do not perform such an analysis, we will not use this notation.

Choosing a basis

To complete the discretization, we must select a basis of V. In the one-dimensional case, for each control point x_k we will choose the piecewise linear function v_k in V whose value is 1 at x_k and zero at every x_j, $j \neq k$, i.e.,



Basis functions v_k (blue) and a linear combination of them, which is piecewise linear (red).

$$v_k(x) = \begin{cases} \frac{x - x_{k-1}}{x_k - x_{k-1}} & \text{if } x \in [x_{k-1}, x_k], \\ \frac{x_{k+1} - x}{x_{k+1} - x_k} & \text{if } x \in [x_k, x_{k+1}], \\ 0 & \text{otherwise,} \end{cases}$$

for $k = 1, \ldots, n$; this basis is a shifted and scaled tent function. For the two-dimensional case, we choose again one basis function v_k per vertex x_k of the triangulation of the planar region Ω. The function v_k is the unique function of V whose value is 1 at x_k and zero at every x_j, $j \neq k$.

Depending on the author, the word "element" in "finite element method" refers either to the triangles in the domain, the piecewise linear basis function, or both. So for instance, an author interested in curved domains might replace the triangles with curved primitives, and so might describe the elements as being curvilinear. On the other hand, some authors replace "piecewise linear" by "piecewise quadratic" or even "piecewise polynomial". The author might then say "higher order element" instead of "higher degree polynomial". Finite element method is not restricted to triangles (or tetrahedra in 3-d, or higher order simplexes in multidimensional spaces), but can be defined on quadrilateral subdomains (hexahedra, prisms, or pyramids in 3-d, and so on). Higher order shapes (curvilinear elements) can be defined with polynomial and even non-polynomial shapes (e.g. ellipse or circle).

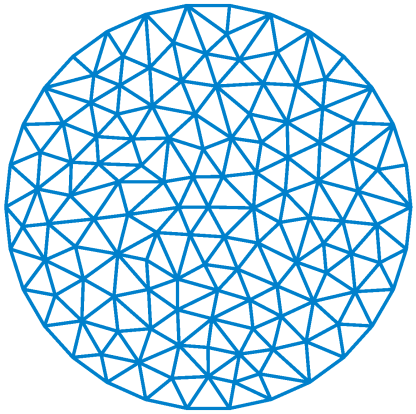
Examples of methods that use higher degree piecewise polynomial basis functions are the hp-FEM and spectral FEM.

More advanced implementations (adaptive finite element methods) utilize a method to assess the quality of the results (based on error estimation theory) and modify the mesh during the solution aiming to achieve approximate solution within some bounds from the 'exact' solution of the continuum problem. Mesh adaptivity may utilize various techniques, the most popular are:

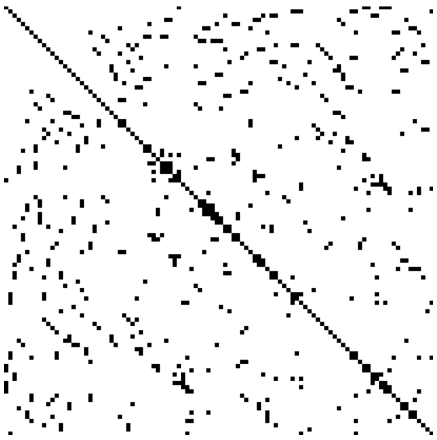
- moving nodes (r-adaptivity)
- refining (and unrefining) elements (h-adaptivity)
- changing order of base functions (p-adaptivity)
- combinations of the above (hp-adaptivity)

Small support of the basis

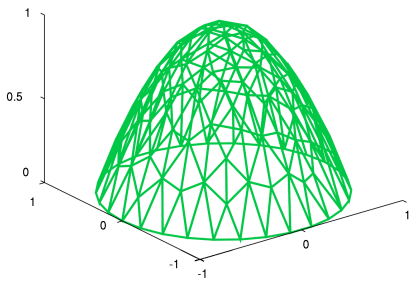
The primary advantage of this choice of basis is that the inner products



Solving the two-dimensional problem $u_{xx} + u_{yy} = -4$ in the disk centered at the origin and radius 1, with zero boundary conditions.
(a) The triangulation.



(b) The sparse matrix *L* of the discretized linear system.



(c) The computed solution, $u(x, y) = 1 - x^2 - y^2$.

$$\langle v_j, v_k \rangle = \int_0^1 v_j v_k \, dx$$

and

$$\phi(v_j, v_k) = \int_0^1 v_j' v_k' \, dx$$

will be zero for almost all j, k. (The matrix containing $\langle v_j, v_k \rangle$ in the (j, k) location is known as the Gramian matrix.) In the one dimensional case, the support of v_k is the interval $[x_{k-1}, x_{k+1}]$. Hence, the integrands of $\langle v_j, v_k \rangle$ and $\phi(v_j, v_k)$ are identically zero whenever $|j - k| > 1$.
Similarly, in the planar case, if x_j and x_k do not share an edge of the triangulation, then the integrals

$$\int_\Omega v_j v_k \, ds$$

and

$$\int_\Omega \nabla v_j \cdot \nabla v_k \, ds$$

are both zero.

Matrix form of the problem

If we write $u(x) = \sum_{k=1}^{n} u_k v_k(x)$ and $f(x) = \sum_{k=1}^{n} f_k v_k(x)$ then problem (3), taking $v(x) = v_j(x)$ for $j = 1, ..., n$, becomes

$$-\sum_{k=1}^{n} u_k \phi(v_k, v_j) = \sum_{k=1}^{n} f_k \int v_k v_j dx \text{ for } j = 1, ..., n \text{ .} \quad (4)$$

If we denote by $\mathbf{u}$ and $\mathbf{f}$ the column vectors $(u_1, ..., u_n)^t$ and $(f_1, ..., f_n)^t$, and if we let

$$L = (L_{ij})$$

and

$$M = (M_{ij})$$

be matrices whose entries are

$$L_{ij} = \phi(v_i, v_j)$$

and

$$M_{ij} = \int v_i v_j dx$$

then we may rephrase (4) as

$$-L\mathbf{u} = M\mathbf{f}. \quad (5)$$

It is not, in fact, necessary to assume $f(x) = \sum_{k=1}^{n} f_k v_k(x)$. For a general function $f(x)$, problem (3) with $v(x) = v_j(x)$ for $j = 1, ..., n$ becomes actually simpler, since no matrix M is used,

$$-L\mathbf{u} = \mathbf{b}, \quad (6)$$

where $\mathbf{b} = (b_1, ..., b_n)^t$ and $b_j = \int f v_j dx$ for $j = 1, ..., n$.

As we have discussed before, most of the entries of L and M are zero because the basis functions v_k have small support. So we now have to solve a linear system in the unknown $\mathbf{u}$ where most of the entries of the matrix L, which we need to invert, are zero.

Such matrices are known as sparse matrices, and there are efficient solvers for such problems (much more efficient than actually inverting the matrix.) In addition, L is symmetric and positive definite, so a technique such as the conjugate gradient method is favored. For problems that are not too large, sparse LU decompositions and Cholesky decompositions still work well. For instance, Matlab's backslash operator (which uses sparse LU, sparse Cholesky, and other factorization methods) can be sufficient for meshes with a hundred thousand vertices.

The matrix L is usually referred to as the stiffness matrix, while the matrix M is dubbed the mass matrix.

General form of the finite element method

In general, the finite element method is characterized by the following process.

- One chooses a grid for Ω. In the preceding treatment, the grid consisted of triangles, but one can also use squares or curvilinear polygons.
- Then, one chooses basis functions. In our discussion, we used piecewise linear basis functions, but it is also common to use piecewise polynomial basis functions.

A separate consideration is the smoothness of the basis functions. For second order elliptic boundary value problems, piecewise polynomial basis function that are merely continuous suffice (i.e., the derivatives are discontinuous.) For higher order partial differential equations, one must use smoother basis functions. For instance, for a fourth order problem such as $u_{xxxx} + u_{yyyy} = f$, one may use piecewise quadratic basis functions that are C^1.

Another consideration is the relation of the finite dimensional space V to its infinite dimensional counterpart, in the examples above H_0^1. A conforming element method is one in which the space V is a subspace of the element space for the continuous problem. The example above is such a method. If this condition is not satisfied, we obtain a nonconforming element method, an example of which is the space of piecewise linear functions over the mesh which are continuous at each edge midpoint. Since these functions are in general discontinuous along the edges, this finite dimensional space is not a subspace of the original H_0^1.

Typically, one has an algorithm for taking a given mesh and subdividing it. If the main method for increasing precision is to subdivide the mesh, one has an *h*-method (*h* is customarily the diameter of the largest element in the mesh.) In this manner, if one shows that the error with a grid h is bounded above by Ch^p, for some $C < \infty$ and $p > 0$, then one has an order *p* method. Under certain hypotheses (for instance, if the domain is convex), a piecewise polynomial of order d method will have an error of order $p = d + 1$.

If instead of making *h* smaller, one increases the degree of the polynomials used in the basis function, one has a *p*-method. If one combines these two refinement types, one obtains an *hp*-method (hp-FEM). In the hp-FEM, the polynomial degrees can vary from element to element. High order methods with large uniform *p* are called spectral finite element methods (SFEM). These are not to be confused with spectral methods.

For vector partial differential equations, the basis functions may take values in $\mathbb{R}^n$.

Comparison to the finite difference method

The finite difference method (FDM) is an alternative way of approximating solutions of PDEs. The differences between FEM and FDM are:

- The most attractive feature of the FEM is its ability to handle complicated geometries (and boundaries) with relative ease. While FDM in its basic form is restricted to handle rectangular shapes and simple alterations thereof, the handling of geometries in FEM is theoretically straightforward.
- The most attractive feature of finite differences is that it can be very easy to implement.
- There are several ways one could consider the FDM a special case of the FEM approach. E.g., first order FEM is identical to FDM for Poisson's equation, if the problem is discretized by a regular rectangular mesh with each rectangle divided into two triangles.
- There are reasons to consider the mathematical foundation of the finite element approximation more sound, for instance, because the quality of the approximation between grid points is poor in FDM.
- The quality of a FEM approximation is often higher than in the corresponding FDM approach, but this is extremely problem dependent and several examples to the contrary can be provided.

Generally, FEM is the method of choice in all types of analysis in structural mechanics (i.e. solving for deformation and stresses in solid bodies or dynamics of structures) while computational fluid dynamics (CFD) tends to use FDM or other methods like finite volume method (FVM). CFD problems usually require discretization of the problem into a large number of cells/gridpoints (millions and more), therefore cost of the solution favors simpler, lower order approximation within each cell. This is especially true for 'external flow' problems, like air flow around the car or airplane, or weather simulation in a large area.

Various types of finite element methods

Generalized finite element method

The Generalized Finite Element Method (GFEM) uses local spaces consisting of functions, not necessarily polynomials, that reflect the available information on the unknown solution and thus ensure good local approximation. Then a partition of unity is used to “bond” these spaces together to form the approximating subspace. The effectiveness of GFEM has been shown when applied to problems with domains having complicated boundaries, problems with micro-scales, and problems with boundary layers.[9]

hp-FEM

The hp-FEM combines adaptively elements with variable size h and polynomial degree p in order to achieve exceptionally fast, exponential convergence rates.[10]

hpk-FEM

The hpk-FEM combines adaptively elements with variable size h, polynomial degree of the local approximations p and global differentiability of the local approximations $(k-1)$ in order to achieve best convergence rates.

Other applications of finite elements analysis

FEA has also been proposed to use in stochastic modelling, for numerically solving probability models. See the references list[11] .[12]

References

[1] Giuseppe Pelosi (2007). "The finite-element method, Part I: R. L. Courant: Historical Corner". doi:10.1109/MAP.2007.376627.

[2] E. Stein (2009), Olgierd C. Zienkiewicz, a pioneer in the development of the finite element method in engineering science. Steel Construction, 2 (4), 264-272.

[3] Matrix Analysis Of Framed Structures, 3rd Edition by Jr. William Weaver, James M. Gere, 3rd Edition, Springer-Verlag New York, LLC, ISBN 978-0-412-07861-3, First edition 1966

[4] Strang, Gilbert; Fix, George (1973). *An Analysis of The Finite Element Method*. Prentice Hall. ISBN 0130329460.

[5] Roberto Coccioli, Tatsuo Itoh, Giuseppe Pelosi, Peter P. Silvester (1996). "Finite-element methods in microwaves: a selected bibliography". doi:10.1109/74.556518.

[6] http://impact.sourceforge.net

[7] Hastings, J. K., Juds, M. A., Brauer, J. R., *Accuracy and Economy of Finite Element Magnetic Analysis*, 33rd Annual National Relay Conference, April 1985.

[8] McLaren-Mercedes (2006). "Vodafone McLaren-Mercedes: Feature - Stress to impress" (http://web.archive.org/web/20061030200423/http://www.mclaren.com/features/technical/stress_to_impress.php). Archived from the original (http://www.mclaren.com/features/technical/stress_to_impress.php) on 2006-10-30. . Retrieved 2006-10-03.

[9] Babuška, Ivo; Banerjee, Uday; Osborn, John E. (June 2004). "Generalized Finite Element Methods: Main Ideas, Results, and Perspective". *International Journal of Computational Methods* **1** (1): 67–103. doi:10.1142/S0219876204000083.

[10] P. Solin, K. Segeth, I. Dolezel: Higher-Order Finite Element Methods, Chapman & Hall/CRC Press, 2003

[11] "Methods with high accuracy for finite element probability computing" by Peng Long, Wang Jinliang and Zhu Qiding, in Journal of Computational and Applied Mathematics 59 (1995) 181-189

[12] Achintya Haldar and Sankaran mahadan: "Reliability Assessment Using Stochastic Finite Element Analysis", John Wiley & sons.

External links

web.comlab.ox.ac.uk/people/endre.suli/fem.pdf

- NAFEMS (http://www.nafems.org) -- The International Association for the Engineering Analysis Community
- Finite Element Analysis Resources (http://www.feadomain.com)- Finite Element news, articles and tips
- Finite-element Methods for Electromagnetics (http://www.fieldp.com/femethods.html) - free 320-page text
- Finite Element Books (http://www.solid.ikp.liu.se/fe/index.html)- books bibliography
- Mathematics of the Finite Element Method (http://math.nist.gov/mcsd/savg/tutorial/ansys/FEM/)
- Finite Element Methods for Partial Differential Equations (http://web.comlab.ox.ac.uk/people/endre.suli/fem.pdf) - Lecture notes by Endre Süli
- FEA Described - what is it, what is it for. (http://knol.google.com/k/fea)

List of finite element software packages

This is a list of software packages that implement the finite element method for solving partial differential equations or aid in the pre- and post-processing of finite element models.

Free/Open source

- CalculiX is an Open Source FEA project. The solver uses a partially compatible ABAQUS file format. The pre/post-processor generates input data for many FEA and CFD applications.
- Code Aster: French software written in Python and Fortran, GPL license.
- DUNE, Distributed and Unified Numerics Environment GPL Version 2 with Run-Time Exception, written in C++
- Elmer FEM solver: Open source multiphysical simulation software developed by Finnish Ministry of Education's CSC, written primarily in Fortran
- FEBio, Finite Elements for Biomechanics
- FEniCS Project: a LGPL-licensed software package developed by American and European researchers
- FreeFem++: GPL software
- Hermes Project: Modular C/C++ library for rapid development of space- and space-time adaptive hp-FEM solvers.
- Impact: Dynamic Finite Element Program Suite, for dynamic events like crashes, written in Java, GNU license
- OOFEM: Object Oriented Finite EleMent solver, written in C++, GPL v2 license
- OpenFOAM (Field Operation And Manipulation). Originally for CFD only, but now includes finite element analysis through tetrahedral decomposition of arbitrary grids.
- OpenSees is an Open System for Earthquake Engineering Simulation
- Z88: FEM-software available for Windows and Linux/UNIX, written in C, GPL license

Proprietary/Commercial

- Abaqus: Franco-American software from SIMULIA, owned by Dassault Systemes
- ADINA
- Advance Design BIM software for FEM structural analysis, including international design eurocodes, a solution developed by GRAITEC
- ANSA: An advanced CAE pre-processing software for complete model build up.
- ANSYS: American software
- AutoForm: Swiss origin German software for Sheet metal forming process chain
- COMSOL Multiphysics COMSOL Multiphysics Finite Element Analysis Software formerly Femlab
- Creo Elements / Pro Mechanica: A p-version finite element program that is embedded in the MCAD application Creo Elements Pro, from PTC (Parametric Technology Corporation)
- Diffpack Software for finite element analysis and partial differential equations
- Falcon2.0 : Lightweight FEM POST Processor and Viewer for 3D UNV and NASTRAN files
- FEFLOW: simulates groundwater flow, mass transfer and heat transfer in porous media
- Femap, Siemens PLM Software: A pre and post processor for Windows
- FEMtools, Dynamic Design Solutions: A toolbox for static and dynamic simulation, verification, validation and updating of finite element models. Includes also modules for structural optimization and for obtaining experimental reference data.
- FlexPDE
- Flux : American electromagnetic and thermal FEA
- Genie: DNV (Det Norske Veritas) Software

- HydroGeoSphere: A 3D control-volume finite element hydrologic model, simulating surface and subsurface water flow and solute and thermal energy transport
- JMAG: Japanese software
- LS-DYNA, LSTC - Livermore Software Technology Corporation
- LUSAS: UK Software
- MADYMO: TASS - TNO Automotive Safety Solutions
- Nastran: American software, from MSC Software
- nastran/EM: Nastran Suit for highly advanced Durability & NVH Analyses of Engines; born from the AK32 Benchmark of Audi, BMW, Daimler, Porsche & VW; Source Code available
- NEi Fusion, NEi Software: 3D CAD modeler + Nastran FEA
- NEi Nastran, NEi Software: General purpose Finite Element Analysis
- NEi Works, NEi Software: Embedded Nastran for SolidWorks users
- NISA: Indian software
- PAK: Serbian software for linear and nonlinear structural analysis, heat conduction, fluid mechanics with heat transfer, coupled problems, biomechanics, fracture mechanics and fatigue.
- PZFlex: American software for wave propagation and piezoelectric devices
- Quickfield : Physics simulating software
- Radioss: A linear and nonlinear solver owned by Altair Engineering
- Range Software: Multiphysics simulation software
- SAMCEF: CAE package developed by the Belgian company
- SAP2000: American software
- STRAND7: Developed in Sydney Australia by Strand7 Pty. Ltd. Marketed as Straus7 in Europe.
- StressCheck developed by ESRD, Inc (USA) emphasizing solution accuracy by utilizing high order elements
- V*flo*: Physics-based distributed hydrologic modeling software, developed by Vieux & Associates, Inc.
- Visualfea: User friendly finite element analysis program developed by Intuition Software,Inc.
- Zébulon: French software

References

External links

- Public Domain FE Programs (http://homepage.usask.ca/~ijm451/finite/fe_resources/node139.html) listed by Ian MacPhedran (http://homepage.usask.ca/~ijm451/)
- What is the status of open source finite element codes? (http://imechanica.org/node/470) - a discussion thread at the imechanica.org forums

Article Sources and Contributors

Finite element method in structural mechanics *Source*: http://en.wikipedia.org/w/index.php?oldid=437763034 *Contributors*: Art187, Arthena, Auntof6, Australian Lawn Bowler, Basar, Bearcat, Betacommand, Colonies Chris, ELApro, EagleFan, Grafen, Hu12, Inwind, Jitse Niesen, Kandasa, Kgorman-ucb, Luna Santin, Michael Hardy, Pownuk, Prashanthns, Skier Dude, Slffea, Smalljim, Snoktruix, Sureshan, TVBZ28, TYelliot, 20 anonymous edits

Finite element method *Source*: http://en.wikipedia.org/w/index.php?oldid=438426702 *Contributors*: 15.253, 198.144.199.xxx, 6.27, Ae-a, Alansohn, Alec1887, Andre Engels, Andres Agudelo, Annom, Apapte, Arnero, ArnoldReinhold, Art187, Avb, Behshour, Ben pcc, BenFrantzDale, Bfg, Billlion, Boemmels, C quest000, C.Fred, CLW, Caco de vidro, Can't sleep, clown will eat me, Ccchambers, Charles Matthews, CharlesC, Charvest, Ciphers, Cj67, Ckatz, Coco, Cometstyles, Conversion script, Corpx, Czenek, Daa89563, Darko Faba, Diegotorquemada, Drbreznjev, Drdavidhill, Drorata, ED-tech, EconoPhysicist, Eijkhout, EndingPop, Epbr123, Fama Clamosa, Favonian, Francis Ocoma, Fredrik, GTBacchus, Giftlite, Hellisess, Hongooi, Hu12, Igny, Inwind, Ipapasa, IvanLanin, J-Wiki, Jackfork, Jackzhp, Jdolbow, Jitse Niesen, Jmath666, Jvierine, Kallog, Kandasa, Kenyob, Kevinmon, Kjetil1001, Kkmurray, Klochkov.ivan, Knepley, Knockrc, Kri, Kusma, Laesod, Langmore, Ldemasi, Liberatus, Liugr, Loisel, LokiClock, M karzarj, MaNeMeBasat, Mackant1, Marc omorain, MarcosAlvarez, Martinthoegersen, Martynas Patasius, Mat cross, Mattblack82, Matusz, Michael Hardy, Momo san, MrOllie, Naddy, Nickj, Olaftn, Oleg Alexandrov, Olegalexandrov, PEHowland, Paisa, PascalDeVincenzo, Pauli133, PavelSolin, Pentti71, Perth3, Pfortuny, Pjrm, Pjvbra, Pownuk, Publichealthguru, RPHv, Retaggio, Rjwilmsi, Rklawton, Salgueiro, Salih, Sdrakos, Sigmund, Silly rabbit, Singleheart, Skunkboy74, Snegtrail, Soni007, Soundray, Spinningspark, Stephenkirkup, StevePny, SunCreator, Sveinns, TVBZ28, The Anome, Thubing, Tobias Hoevekamp, Tomeasy, UpstateNYer, User A1, Vadimbern, Waldir, Wdl1961, Wegesrand, Wholmestu, Wikipelli, WojciechSwiderski, Woodstone, WorlwideLeader, XJamRastafire, YuriyMikhaylovskiy, Zureks, Zzuuzz, 虞海, 310 anonymous edits

List of finite element software packages *Source*: http://en.wikipedia.org/w/index.php?oldid=444316555 *Contributors*: Altair-Mark, Amyderouvray, Ar2285, Arjaan, Art187, Aushulz, Belgiansteve, Chowbok, CptPaul, Cuylaerts, DlubMacH, Doedie, Dsrawlins, Elabbasi, Emijrp, EndingPop, Erich Payer, Espenbe, Ewyart, Fcuneo1234, Finite-element-ologist, Forgotten brainchild, Fred SC, GeeKaa, Gknor, HarveyGer, IgnitionMarketing, Impetusafea, Ipapasa, Ivan isabella, JackSchmidt, Jbaylor, Josejmcriollo, Jpereira.ca, Jylee12345, Kate P Stuart, Kenyob, Kotashibata, Lenient7, LilHelpa, Lindemann42, Marupio, Mecanismo, Michael Hardy, MichaelLNorth, Misuca, MrOllie, Murphy667, Mwtoews, Ndrivako, PascalDeVincenzo, Paulovieira, PavelSolin, Peedeebee, ProfessorXY, Radumash, Rwwww, Salih, Smoorefu, Snegtrail, Sskipsims, Terrykubat, Tickingclocks, Tomas.soltys, TrueGrid, Truthunmasked, Ub aab4, Vtaber02, Wikiandpet, 132 anonymous edits

Image Sources, Licenses and Contributors

Image:FEM example of 2D solution.png *Source*: http://en.wikipedia.org/w/index.php?title=File:FEM_example_of_2D_solution.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Zureks

Image:Example of 2D mesh.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Example_of_2D_mesh.png *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Zureks

Image:FAE visualization.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:FAE_visualization.jpg *License*: Public Domain *Contributors*: Assassingr, Inductiveload, MB-one, Maksim, 1 anonymous edits

Image:Finite element method 1D illustration1.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Finite_element_method_1D_illustration1.png *License*: Public Domain *Contributors*: Joelholdsworth, Krishnavedala, Maksim, WikipediaMaster

Image:Piecewise linear function2D.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Piecewise_linear_function2D.svg *License*: Public Domain *Contributors*: Oleg Alexandrov

Image:Finite element method 1D illustration2.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Finite_element_method_1D_illustration2.png *License*: Public Domain *Contributors*: Joelholdsworth, Maksim, Perhelion, WikipediaMaster

Image:Finite element triangulation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Finite_element_triangulation.svg *License*: Public Domain *Contributors*: Oleg Alexandrov

Image:Finite element sparse matrix.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Finite_element_sparse_matrix.png *License*: Public Domain *Contributors*: Oleg Alexandrov

Image:Finite element solution.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Finite_element_solution.svg *License*: Public Domain *Contributors*: Oleg Alexandrov

License