03 - Elementos finitos Lagrangianos para tensión/compresión axial

Diego Andrés Alvarez Marín
Profesor Asociado
Universidad Nacional de Colombia
Sede Manizales

https://github.com/diegoandresalvarez/elementosfinitos

Contenido

- Algoritmos de interpolación
- Polinomios de Lagrange
- Funciones de forma unidimensionales
- Interpolación isoparamétrica
- Integración con cuadraturas de Gauss-Legendre
- Puntos de esfuerzo de Barlow
- Requisitos necesarios y deseables de un EF para la convergencia de la solución.

Algoritmos de interpolación en MATLAB

`vq = interp1(x,v,xq,method,extrapolation)`

Tomado de: https://www.mathworks.com/help/matlab/ref/interp1.html

Method	Description	Continuity	Comments
`'linear'`	Linear interpolation. The interpolated value at a query point is based on linear interpolation of the values at neighboring grid points in each respective dimension. This is the default interpolation method.	C^0	• Requires at least 2 points • Requires more memory and computation time than nearest neighbor
`'nearest'`	Nearest neighbor interpolation. The interpolated value at a query point is the value at the nearest sample grid point.	Discontinuous	• Requires at least 2 points • Modest memory requirements • Fastest computation time
`'next'`	Next neighbor interpolation. The interpolated value at a query point is the value at the next sample grid point.	Discontinuous	• Requires at least 2 points • Same memory requirements and computation time as `'nearest'`
`'previous'`	Previous neighbor interpolation. The interpolated value at a query point is the value at the previous sample grid point.	Discontinuous	• Requires at least 2 points • Same memory requirements and computation time as `'nearest'`
`'pchip'`	Shape-preserving piecewise cubic interpolation. The interpolated value at a query point is based on a shape-preserving piecewise cubic interpolation of the values at neighboring grid points.	C^1	• Requires at least 4 points • Requires more memory and computation time than `'linear'`
`'cubic'`	Cubic convolution used in MATLAB® 5.	C^1	• Requires at least 3 points • Points must be uniformly spaced • This method falls back to `'spline'` interpolation for irregularly-spaced data • Similar memory requirements and computation time as `'pchip'`
`'v5cubic'`	Same as `'cubic'`.	C^1	
`'makima'`	Modified Akima cubic Hermite interpolation. The interpolated value at a query point is based on a piecewise function of polynomials with degree at most three. The Akima formula is modified to avoid overshoots.	C^1	• Requires at least 2 points • Produces fewer undulations than `'spline'`, but does not flatten as aggressively as `'pchip'` • Computation is more expensive than `'pchip'`, but typically less than `'spline'` • Memory requirements are similar to those of `'spline'`
`'spline'`	Spline interpolation using not-a-knot end conditions. The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension.	C^2	• Requires at least 4 points • Requires more memory and computation time than `'pchip'`

En: https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/1D/EF_barra_n_nodos/

Ver: `comparing_interpolation_algorithms.{m, py}`

Polinomios de Lagrange

Ver: http://www.matematicasvisuales.com/html/analisis/interpolacion/lagrange.html

El **polinomio interpolador de Lagrange** se define como el polinomio de grado $n-1$ que pasa por los n puntos $\{(x_i, y_i) : i = 1,\ 2,\ \ldots n\}$ donde todos los x_i se asumen distintos. Dicho polinomio se define como la combinación lineal:

$$L(x) = \sum_{i=1}^{n} N_i(x) y_i$$

de *bases polinómicas de Lagrange*:

$$N_i(x) = \prod_{j=1,\, i\neq j}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_1}{x_i - x_1} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}.$$

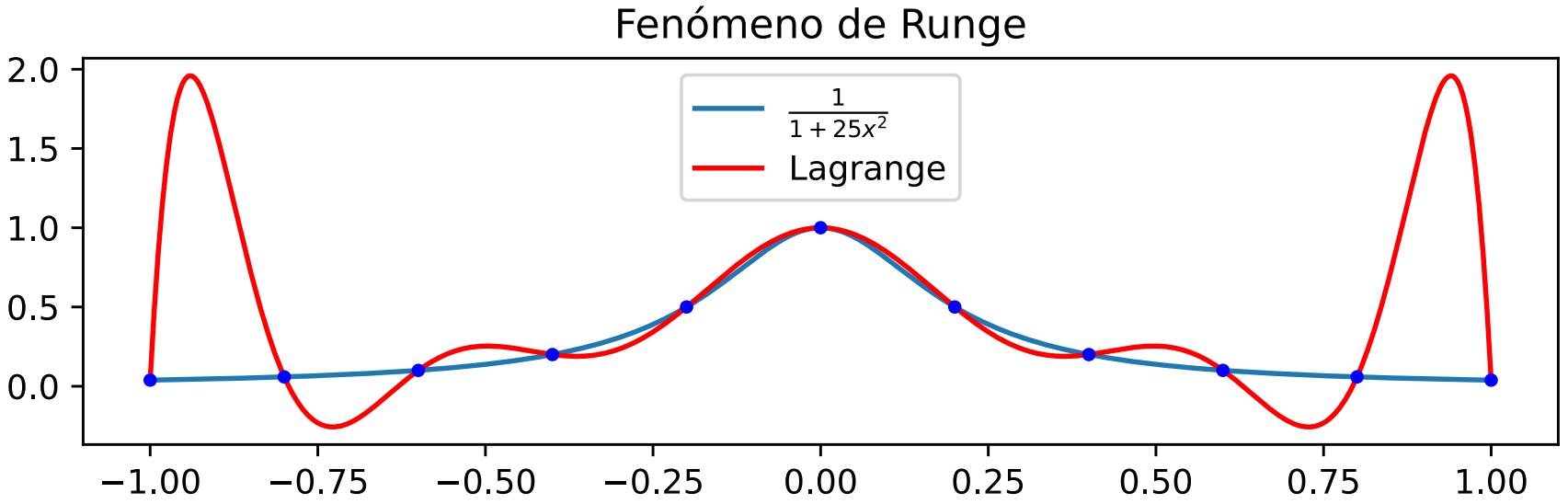
Observe que el numerador de N_i es un polinomio $p_i(x) = (x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)$, el cual tiene la peculiaridad de que $p_i(x_j) = 0$ si $i \neq j$. El denominador de N_i es $p_i(x_i)$ y su mision es normalizar p_i para que $N_i(x_j) = \delta_{ij}$, donde δ_{ij} representa un delta Kronecker.

Joseph-Louis Lagrange (1736 - 1813)

Edward Waring (1736 - 1798)

El matemático italiano-francés Joseph-Louis Lagrange, publicó el método de los polinomios de Lagrange en 1795; sin embargo, el método fue descubierto por primera vez en 1779 por el matemático inglés Edward Waring.

Fenómeno de Runge



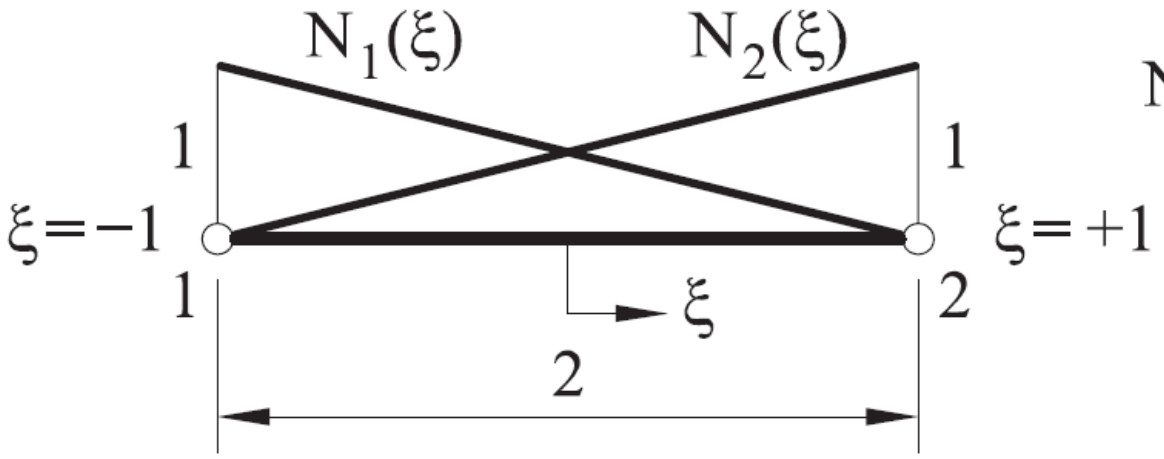
Los polinomios interpoladores de Lagrange sufren del llamado fenómeno de Runge, el cual se manifiesta como "**oscilaciones salvajes**" de dicho polinomio interpolador. Este fenómeno sucede cuando se usa interpolación polinómica con polinomios de alto grado utilizando nodos aproximadamente equidistantes. Esto demuestra que el uso de polinomios de alto grado no necesariamente mejora la precisión.

Funciones de forma (2 nodos)

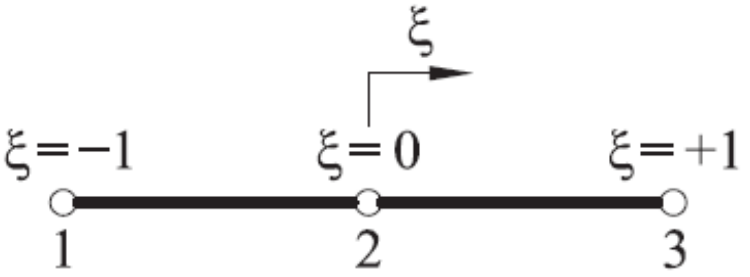
Para un elemento lagrangiano de dos nodos con $\xi_1 = -1$ y $\xi_2 = 1$:

$$N_1 = \frac{\xi - \xi_2}{\xi_1 - \xi_2} = \frac{1}{2}(1 - \xi)$$

$$N_2 = \frac{\xi - \xi_1}{\xi_2 - \xi_1} = \frac{1}{2}(1 + \xi)$$



Funciones de forma (3 nodos)

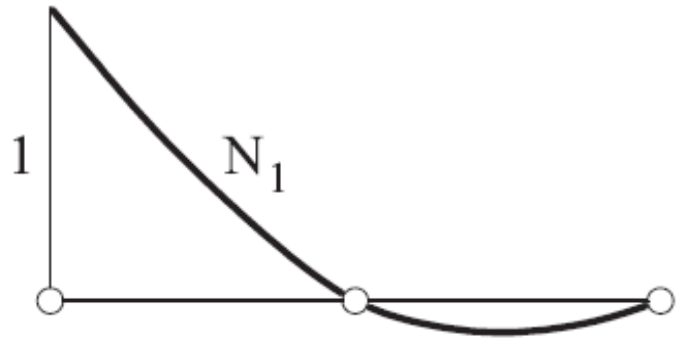


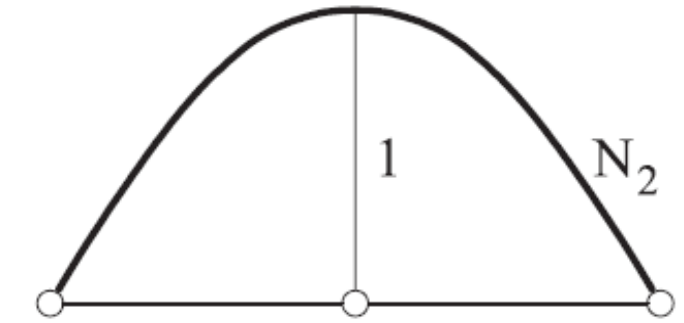
Para un elemento lagrangiano de tres nodos en $\xi_1 = -1$, $\xi_2 = 0$ y $\xi_3 = +1$ se deduce:

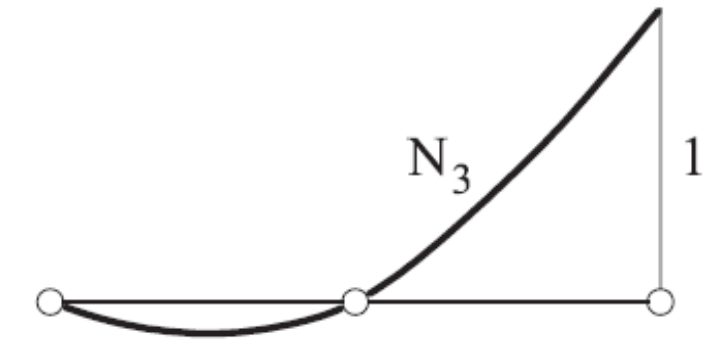
$$N_1(\xi) = \frac{(\xi - \xi_2)(\xi - \xi_3)}{(\xi_1 - \xi_2)(\xi_1 - \xi_3)} = \frac{1}{2}\xi(\xi - 1)$$

$$N_2(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_3)}{(\xi_2 - \xi_1)(\xi_2 - \xi_3)} = (1 + \xi)(1 - \xi)$$

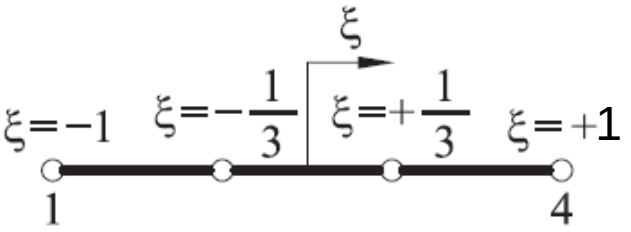
$$N_3(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_2)}{(\xi_3 - \xi_1)(\xi_3 - \xi_2)} = \frac{1}{2}\xi(\xi + 1)$$

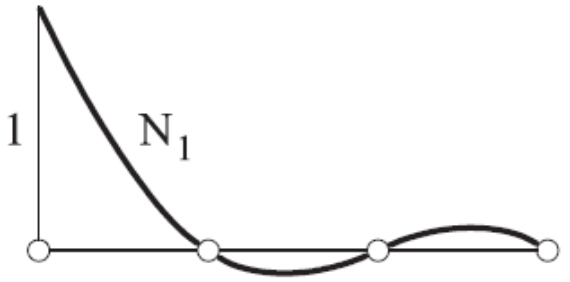






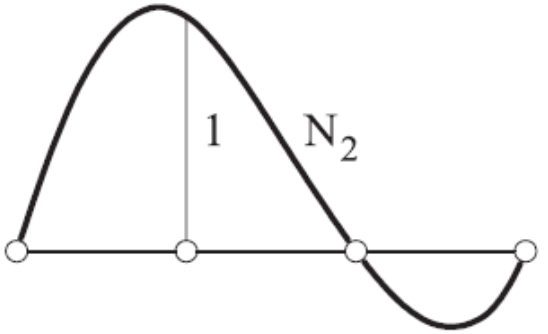
Funciones de forma (4 nodos)





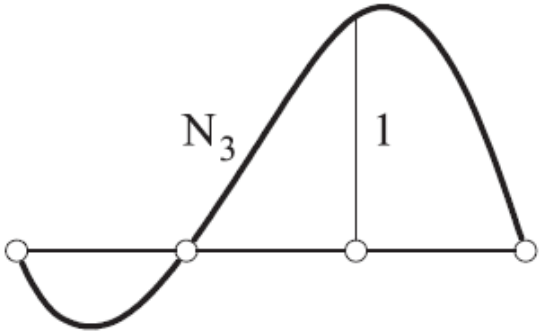
Para un elemento lagrangiano de cuatro nodos en $\xi_1 = -1$, $\xi_2 = -1/3$, $\xi_3 = +1/3$ y $\xi_4 = +1$ se deduce:

$$N_1(\xi) = \frac{(\xi-\xi_2)(\xi-\xi_3)(\xi-\xi_4)}{(\xi_1-\xi_2)(\xi_1-\xi_3)(\xi_1-\xi_4)} = -\frac{9}{16}(\xi+1/3)(\xi-1/3)(\xi-1)$$

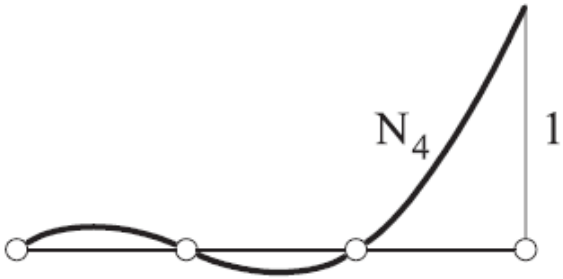


$$N_2(\xi) = \frac{(\xi-\xi_1)(\xi-\xi_3)(\xi-\xi_4)}{(\xi_2-\xi_1)(\xi_2-\xi_3)(\xi_2-\xi_4)} = +\frac{27}{16}(\xi+1)(\xi-1/3)(\xi-1)$$

$$N_3(\xi) = \frac{(\xi-\xi_1)(\xi-\xi_2)(\xi-\xi_4)}{(\xi_3-\xi_1)(\xi_3-\xi_2)(\xi_3-\xi_4)} = -\frac{27}{16}(\xi+1)(\xi+1/3)(\xi-1)$$



$$N_4(\xi) = \frac{(\xi-\xi_1)(\xi-\xi_2)(\xi-\xi_3)}{(\xi_4-\xi_1)(\xi_4-\xi_2)(\xi_4-\xi_3)} = +\frac{9}{16}(\xi+1)(\xi+1/3)(\xi-1/3)$$



```
%% Funciones de Forma Lagrangianas de cuatro nodos

% Calculo las funciones de forma
syms xi
N1 = factor(poly2sym(polyfit([-1 -1/3 1/3 1],[1 0 0 0],3),xi));
N2 = factor(poly2sym(polyfit([-1 -1/3 1/3 1],[0 1 0 0],3),xi));
N3 = factor(poly2sym(polyfit([-1 -1/3 1/3 1],[0 0 1 0],3),xi));
N4 = factor(poly2sym(polyfit([-1 -1/3 1/3 1],[0 0 0 1],3),xi));

% Imprimo las funciones de forma
fprintf('\n\nFunciones de Forma Lagrangianas de CUATRO nodos:\n')
fprintf('\n\nN1 = '), pretty(N1);  fprintf('\n\nN2 = '), pretty(N2)
fprintf('\n\nN3 = '), pretty(N3);  fprintf('\n\nN4 = '), pretty(N4)

% Grafico las funciones de forma
figure                  % Creo un lienzo
grid on                 % creo la rejilla
hold on;                % Para que no se sobreescriban los graficos
h1 = ezplot(N1, [-1 1]); set(h1, 'Color', 'r', 'LineWidth', 2);
h2 = ezplot(N2, [-1 1]); set(h2, 'Color', 'b', 'LineWidth', 2);
h3 = ezplot(N3, [-1 1]); set(h3, 'Color', 'c', 'LineWidth', 2);
h4 = ezplot(N4, [-1 1]); set(h4, 'Color', 'm', 'LineWidth', 2);
legend('N1(\xi)','N2(\xi)','N3(\xi)','N4(\xi)','Location','Best');
title('Funciones de Forma Lagrangianas de CUATRO nodos')
xlabel('\xi');
ylabel('N_i(\xi)');
plot([-1 -1/3 1/3 1],[0 0 0 0], 'ko', [-1 -1/3 1/3 1],[1 1 1 1], 'ko')
axis([-1 1 -0.4 1.2])
```

Ver programas de MATLAB/PYTHON:
funciones_forma_lagrangianos1D.{m, py}
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/1D/EF_barra_n_nodos

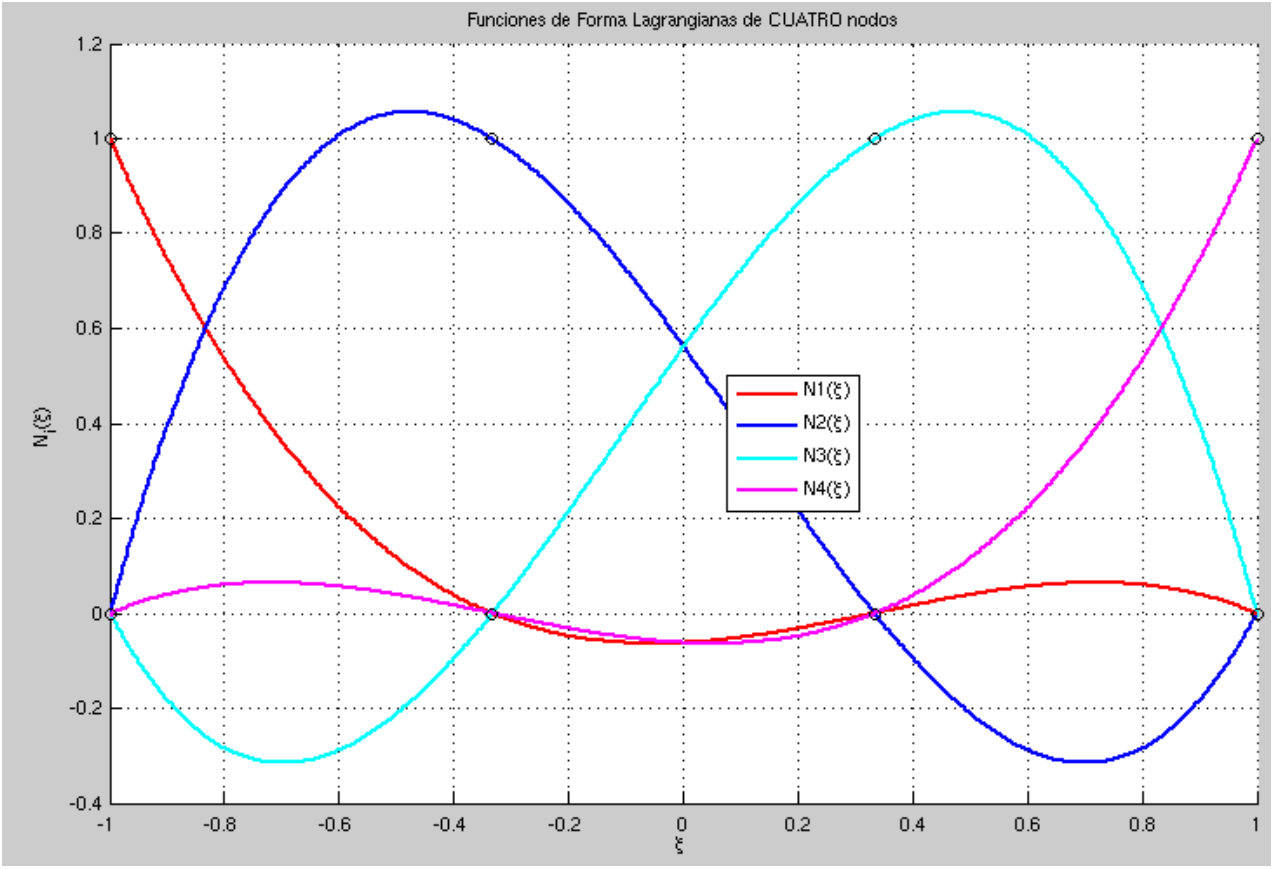
Funciones de Forma Lagrangianas de CUATRO nodos:

$$N1 = -\frac{(xi - 1)\,(3\,xi - 1)\,(3\,xi + 1)}{16}$$

$$N2 = \frac{9\,(xi - 1)\,(3\,xi - 1)\,(xi + 1)}{16}$$

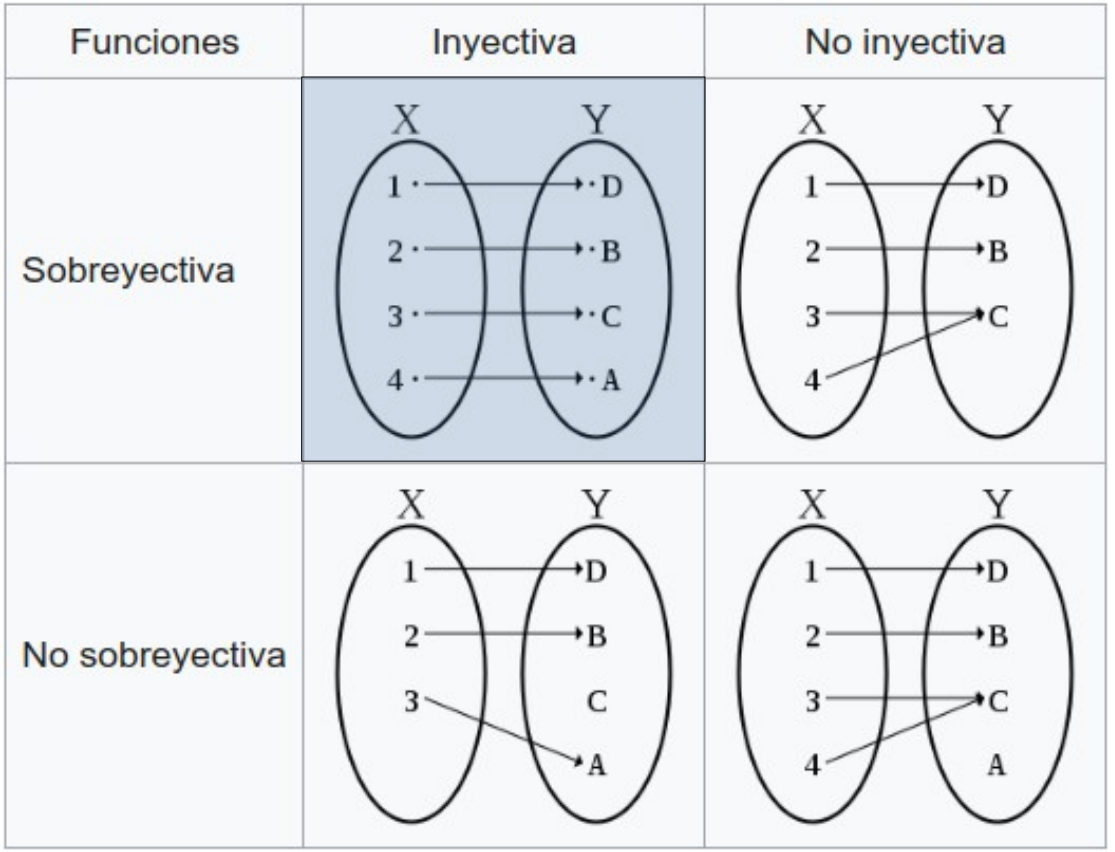
$$N3 = -\frac{9\,(xi - 1)\,(3\,xi + 1)\,(xi + 1)}{16}$$

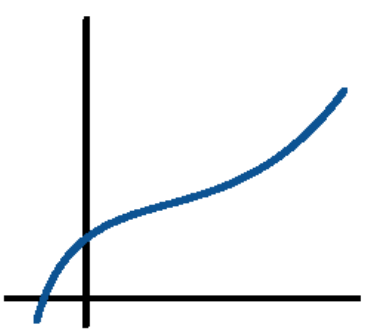
$$N4 = \frac{(3\,xi - 1)\,(3\,xi + 1)\,(xi + 1)}{16}$$

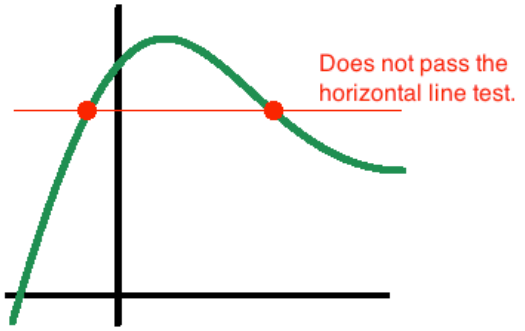


Función biyectiva

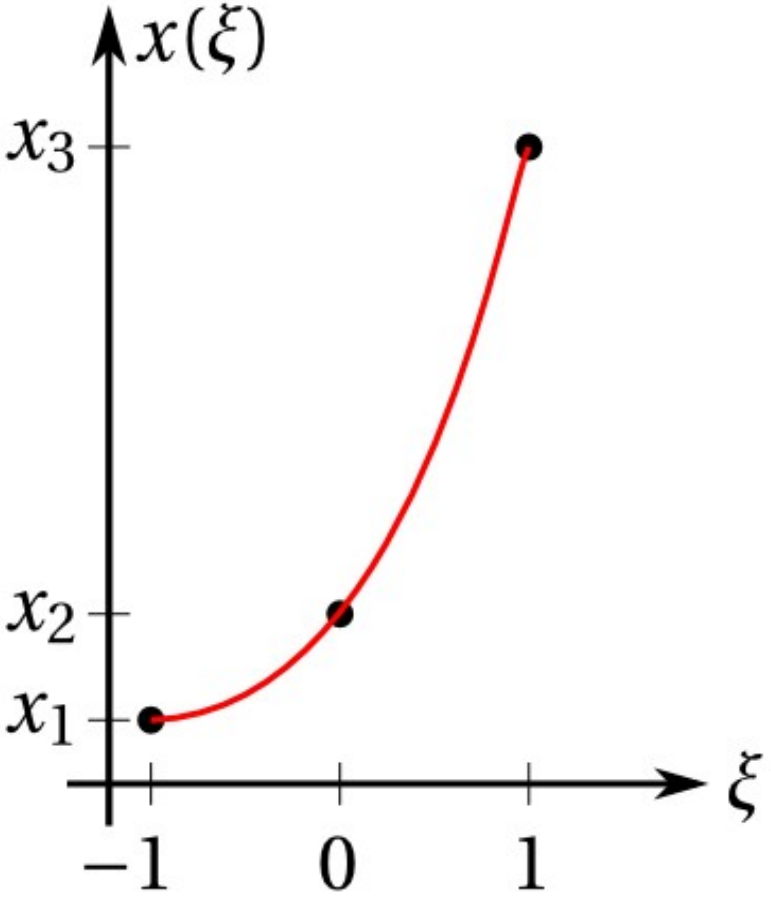
Una función es **biyectiva** si es al mismo tiempo inyectiva y sobreyectiva; es decir, si todos los elementos del conjunto de salida tienen una imagen distinta en el conjunto de llegada (**inyectiva, one-to-one**), y a cada elemento del conjunto de llegada le corresponde un elemento del conjunto de salida (**sobreyectiva, onto**).







Interpolación isoparamétrica



La interpolación isoparamétrica utiliza las mismas funciones de forma del elemento para interpolar la geometría del mismo mediante la ecuación:

$$x(\xi) = \sum_{i=1}^{n} N_i(\xi) x_i$$

Esta transformación debe ser biyectiva, para lo cual se impone que $\frac{\mathrm{d}x}{\mathrm{d}\xi}$ debe ser de signo constante en todo el elemento, generalmente imponiéndose que $\frac{\mathrm{d}x(\xi)}{\mathrm{d}\xi} > 0$ para todo $\xi \in [-1, 1]$.

Tipos de interpolación

$$x(\xi) = \sum_{i=1}^{m} N_i'(\xi) x_i \qquad u(\xi) = \sum_{i=1}^{n} N_i''(\xi) u_i$$

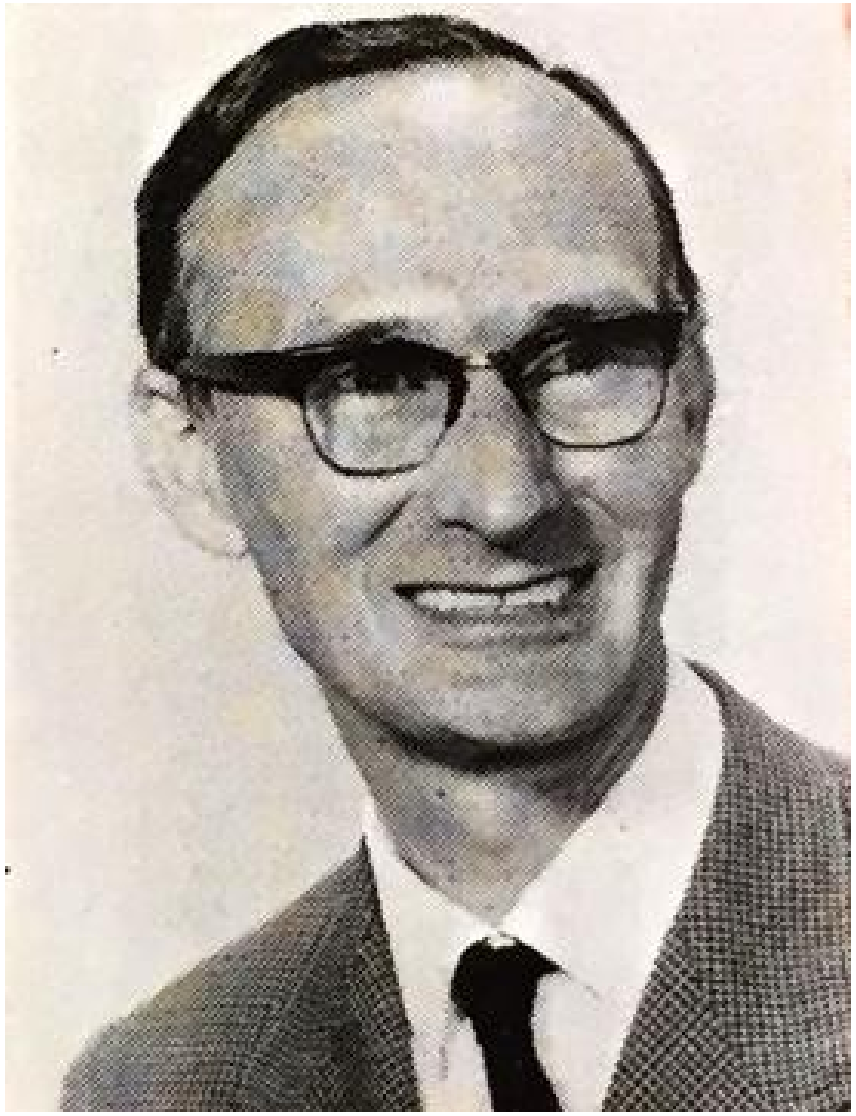
- Isoparamétrica (*m*=*n*)
- Superparamétrica (*m*>*n*)
- Subparamétrica (*m*<*n*)

Historia

Los elementos isoparamétricos se originan del trabajo de ***Ian C. Taig***, quien derivó el primer cuadrilátero isoparamétrico de 4 nodos. ***Bruce M. Irons*** extendió estas ideas para formular elementos isoparamétricos de alto orden.

- Taig, I.C. and Kerr, R.I., *Some problems in the discrete element representation of aircraft structures*. In Matrix Methods of Structural Analysis, B.M. Fraeijs de Veubeke (ed.), Pergamon Press, 1964.
- Taig, I.C., *Structural analysis by the matrix displacement method*. Electric Aviation Report, No. 5017, 1961.
- Irons, B.M. and Ahmad, S., *Techniques of finite elements*. Ellis Harwood, Chichester, 1980.
- Irons, B.M., *Numerical integration applied to finite element method*. Conf. Use of Digital Computers in Struct. Eng., Univ. Newcastle, 1966.

Bruce Moncur Irons (1924 - 1983)



- Físico y matemático inglés.
- Ingeniero de la Rolls-Royce (allí aprendió y desarrolló el MEF ~ 1959-1965).
- Ganó en 1974 el premio von Karman.
- Profesor de University of Wales in Swansea (1966) y University of Calgary (1974), a pesar de que nunca hizo un doctorado.
- Hizo importantes aportes al MEF: desarrolló los elementos isoparamétricos (junto con Ian C. Taig), los elementos serendípitos, el patch test y el frontal solver.
- Sufrió de esclerosis múltiple.

https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620220102
https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620200615

Integración numérica

Las integrales definidas pueden resolverse utilizando mediante métodos numéricos; estos métodos son aproximados, sin embargo el error de la aproximación que dependerá del método que se use y de los parámetros utilizados; este error puede disminuirse incrementando la carga computacional.

- Sumas de Riemann
- Método de Newton-Cotes (trapecio, Simpson, etc)
- Método de Romberg (interpolaciones de Richardson)
- Cuadraturas de Gauss (ej: Gauss-Legendre, Gauss-Hermite, etc)

Cuadratura de Gauss-Legendre

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

La cuadratura de Gauss-Legendre es un método de integración de funciones en el intervalo $[-1, 1]$. La popularidad de la cuadratura de Gauss Legendre se debe a que utiliza un número mínimo de puntos de integración para conseguir un error determinado en el cálculo de la integral, por consiguiente minimiza el número de veces que hay que evaluar el integrando. Adicionalmente, una cuadratura de grado n integra exactamente un polinomio de orden $2n-1$ o menor. El error de integración del método esta dado por:

$$E = \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi) \quad \text{donde } \xi \in (-1, 1).$$

Cuadraturas de Gauss Legendre

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

N=1	x_i	w_i
1	0	2
N=2	x_i	w_i
1	$-\sqrt{\frac{1}{3}} \approx$ -0.57735026919	1
2	$\sqrt{\frac{1}{3}} \approx$ 0.57735026919	1
N=3	x_i	w_i
1	$-\sqrt{\frac{3}{5}} \approx$ -0.774596669241	$\frac{5}{9} \approx$ 0.555555555556
2	0	$\frac{8}{9} \approx$ 0.888888888889
3	$\sqrt{\frac{3}{5}} \approx$ 0.774596669241	$\frac{5}{9} \approx$ 0.555555555556
N=4	x_i	w_i
1	$-\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}} \approx$ -0.861136311594053	$\frac{18-\sqrt{30}}{36} \approx$ 0.347854845137454
2	$-\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}} \approx$ -0.339981043584856	$\frac{18+\sqrt{30}}{36} \approx$ 0.652145154862546
3	$\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}} \approx$ 0.339981043584856	$\frac{18+\sqrt{30}}{36} \approx$ 0.652145154862546
4	$\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}} \approx$ 0.861136311594053	$\frac{18-\sqrt{30}}{36} \approx$ 0.347854845137454

Cuadraturas de Gauss Legendre

N=5	x_i	w_i
1	$-\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}} \approx$ -0.906179845938664	$\frac{322-13\sqrt{70}}{900} \approx$ 0.236926885056189
2	$-\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}} \approx$ -0.538469310105683	$\frac{322+13\sqrt{70}}{900} \approx$ 0.478628670499366
3	0	$\frac{128}{225} \approx$ 0.568888888888889
4	$\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}} \approx$ 0.538469310105683	$\frac{322+13\sqrt{70}}{900} \approx$ 0.478628670499366
5	$\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}} \approx$ 0.906179845938664	$\frac{322-13\sqrt{70}}{900} \approx$ 0.236926885056189

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

Cuando se requiere calcular una integral en el intervalo $[a, b]$, se debe convertir esta al intervalo $[-1, 1]$, con el objeto de poder aplicar la cuadratura de Gauss-Legendre.

Este cambio de variables es:

$$x(\xi) = \frac{b-a}{2}\xi + \frac{a+b}{2}$$

lo cual permite replantear la integral como integral como:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}\xi + \frac{a+b}{2}\right)\mathrm{d}\xi;$$

de este modo, al aplicar la cuadratura de Gauss-Legendre resulta:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}\xi_i + \frac{a+b}{2}\right).$$

Carl Friedrich Gauss
(1777 - 1855)

Carl Gustav Jacob Jacobi
(1804 - 1851)

Carl Friedrich Gauss fue el primero en derivar la regla de cuadratura de Gauss-Legendre, y lo hizo mediante un cálculo con fracciones continuas en 1814. Gauss calculó los nodos y pesos con 16 dígitos hasta el orden $n = 7$ a mano. Carl Gustav Jacob Jacobi descubrió la conexión entre la regla de cuadratura y la familia ortogonal de polinomios de Legendre. Como no existe una fórmula de forma cerrada para los pesos y nodos de cuadratura, durante muchas décadas las personas solo pudieron usarlos a mano para n pequeños, y el cálculo de la cuadratura tuvo que hacerse haciendo referencia a tablas que contenían el peso y los valores de los nodos. En 1942, estos valores solo se conocían hasta $n = 16$.

Cuadraturas de Gauss-Legendre

Los polinomios de Legendre se pueden definir utilizando la llamada *recursión de Bonnet*:

$$P_0(\xi) = 1$$
$$P_1(\xi) = \xi$$
$$P_n(\xi) = \frac{1}{n}\Big[(2n-1)\xi P_{n-1}(\xi) - (n-1)P_{n-2}(\xi)\Big]$$

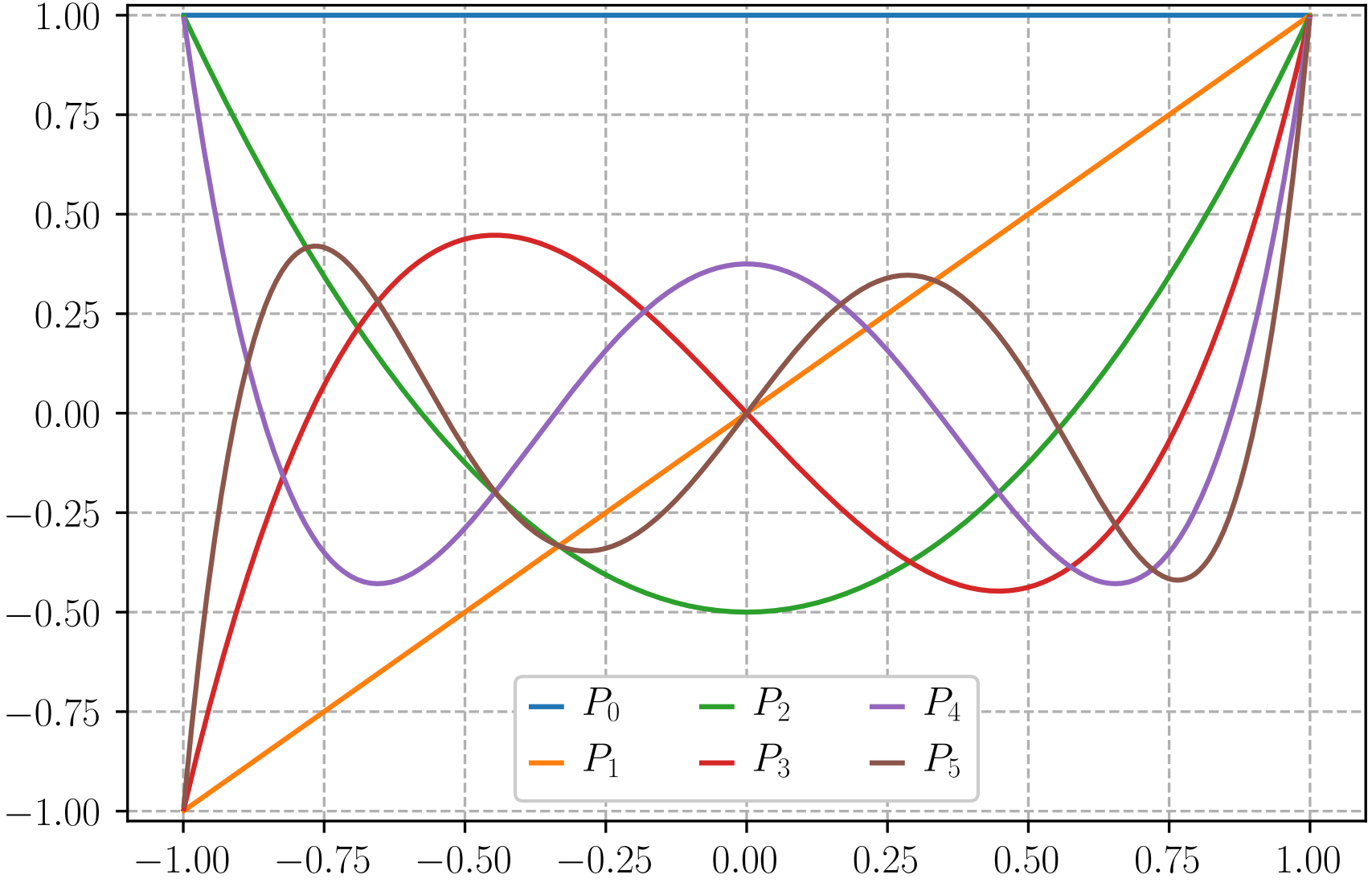
Tenga en cuenta que los valores ξ_i son las raíces del polinomio P_n y los pesos están dados por:

$$w_i = \frac{2}{(1-\xi_i^2)\,[P_n'(\xi_i)]^2} \quad \text{para } i = 1, \ldots, n$$

```
function [xi,w,P] = gausslegendre_quad(m)
% Integration using a Gauss-Legendre quadrature

%% Polynomials (Bonnet's recursion):
P = cell(m+1, 1);
P{0 + 1} = 1;
P{1 + 1} = [1 0];
for n = 2:m
   P{n + 1} = ((2*n-1)*[P{n-1 + 1} 0] - (n-1)*[0 0 P{n-2 + 1}])/n;
end

%% Roots
xi = sort(roots(P{m + 1}));

%% Weights:
s = polyder(P{m + 1});
w = 2./((1 - xi.^2).*polyval(s,xi).^2);
```

Ver programas de MATLAB/PYTHON: `gausslegendre_quad.{m, py}`
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/1D/EF_barra_n_nodos
NOTA: estudie con cuidado ambas funciones que tienen implementaciones muy diferentes

Polinomios de Legendre $P_n(x)$



- Integración de:

$$\int_0^{0.8} 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5 \, \mathrm{d}x$$

Solución exacta:

```
syms x
int(0.2 + 25*x - 200*x^2 + 675*x^3 - 900*x^4 + 400*x^5,x,0,0.8)
```

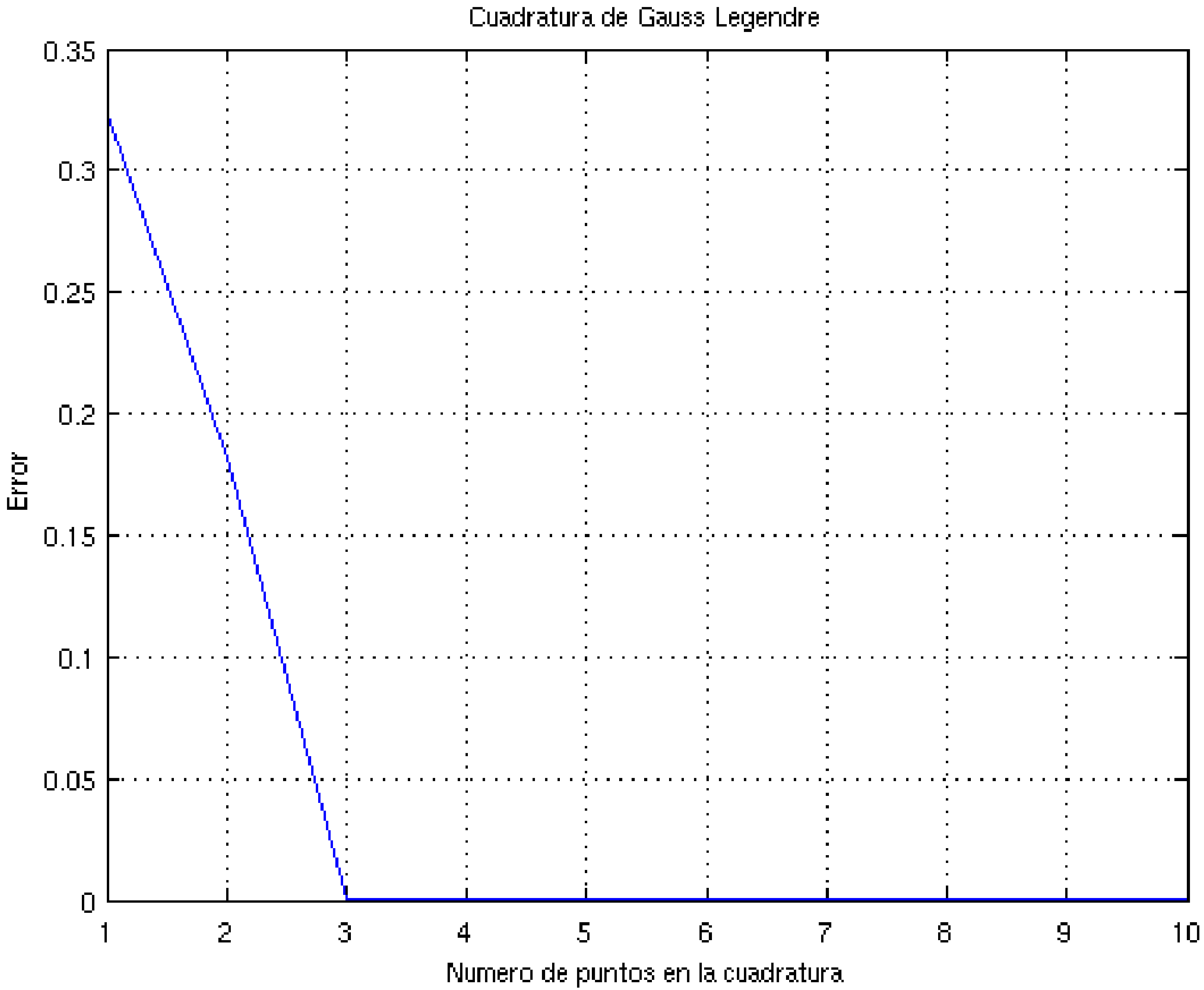
Siendo la respuesta

```
ans =
3076/1875
```

Integración con las cuadraturas de Gauss-Legendre:

```
err = zeros(10,1);          % Separo la memoria
a = 0; b = 0.8;             % Limites de integracion
f = @(x) 0.2 + 25*x - 200*x.^2 +675*x.^3 - 900*x.^4 + 400*x.^5; % la funcion
for m = 1:10;               % Vario el numero de puntos de la cuadratura
   [x,c] = gausslegendre_quad(m);  % calculo w y c de la cuadratura
   err(m) = abs(((b-a)/2)*sum(c.*f((b+a)/2 + (b-a)*x/2)) - 3076/1875);
end;
figure                      % creo un lienzo
plot(err)                   % grafico el error
xlabel('Numero de puntos en la cuadratura');
ylabel('Error');
title('Cuadratura de Gauss Legendre');
grid                        % pongo la rejilla
```

Ver programas de MATLAB/PYTHON: cuadratura_poly_sin_GL.{m, py}
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/1D/EF_barra_n_nodos
NOTA: estudie con cuidado ambas funciones que tienen implementaciones muy diferentes



Análisis de resultados: Recuerde que la cuadratura de Gauss-Legendre de orden n integra EXACTAMENTE un polinomio de grado 2n-1 o menor. Por lo tanto con una cuadratura de grado 3 o mayor se integra exactamente este polinomio (que es de cuarto orden)

- Integración de:

$$\int_0^{\pi/2} \sin x \, dx$$

Solución exacta:

```
syms x
int(sin(x),x,0,pi/2)
```

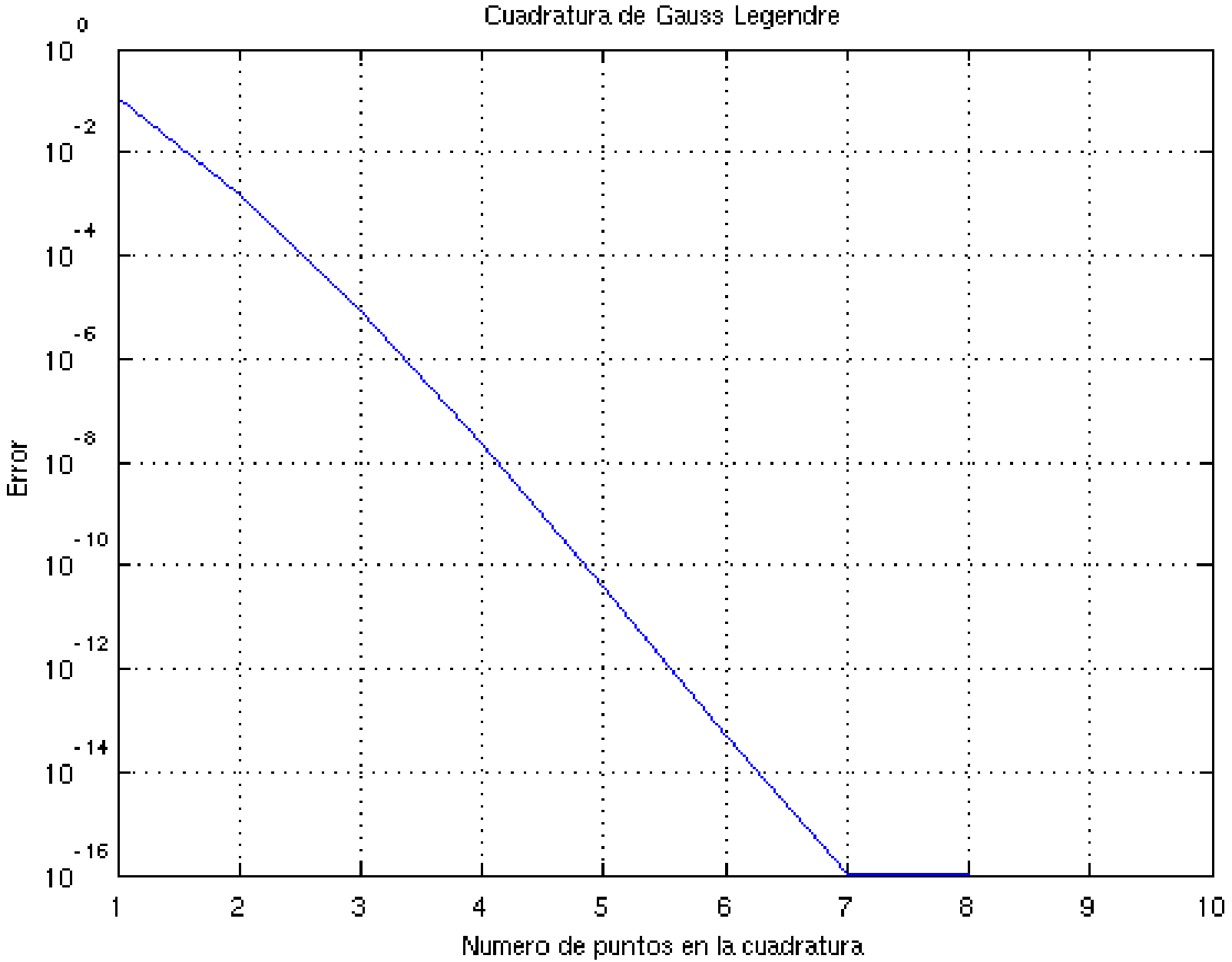
Siendo la respuesta

```
ans =
1
```

Integración con las cuadraturas de Gauss-Legendre:

```
err = zeros(10,1);
a = 0; b = pi/2;
f = @(x) sin(x);
for m = 1:10;
   [x,c] = gausslegendre_quad(m);
   err(m) = abs(((b-a)/2)*sum(c.*f((b+a)/2 + (b-a)*x/2)) - 1);
end;
figure
semilogy(err); % dibujo en escala logarítmica del eje Y (para apreciar el error)
xlabel('Numero de puntos en la cuadratura');
ylabel('Error');
title('Cuadratura de Gauss Legendre');
grid
```

Ver programas de MATLAB/PYTHON: cuadratura_poly_sin_GL.{m, py}
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/codigo/1D/EF_barra_n_nodos
NOTA: estudie con cuidado ambas funciones que tienen implementaciones muy diferentes



Análisis de resultados: Con la cuadratura de orden 8 o mayor se obtiene un error menor de 1e-18, y 1 - 1e-18 excede la precisión de representación de decimales del computador. Por lo tanto el error en la integración el computador lo aproxima a cero (**error de truncamiento**)

Solución utilizando el método de los elementos finitos

Mirar la deducción de ecuaciones:

- `03_EF_1d_n_nodos_isoparametricos.pdf`

Y su implementación en MATLAB/PYTHON:

- `calculo_K_y_f.{m, py}`
- `barra_con_carga_axial_3_nodos_K_f_dado.m`
- `barra_con_carga_axial_3_nodos_gauss_legendre.m`

Dichos archivos se encuentran en:
https://github.com/diegoandresalvarez/elementosfinitos/tree/master/diapositivas

Puntos óptimos para el cálculo del esfuerzo o puntos de Barlow

Los esfuerzos se deben calcular en los puntos de integración de Gauss-Legendre, utilizando integración reducida.

Barlow, J. (1976), *Optimal stress locations in finite element models*. International Journal for Numerical Methods in Engineering, Vol 10 (2), p. 243-251. https://doi.org/10.1002/nme.1620100202

Este criterio para el cálculo de esfuerzos es también válido en más dimensiones

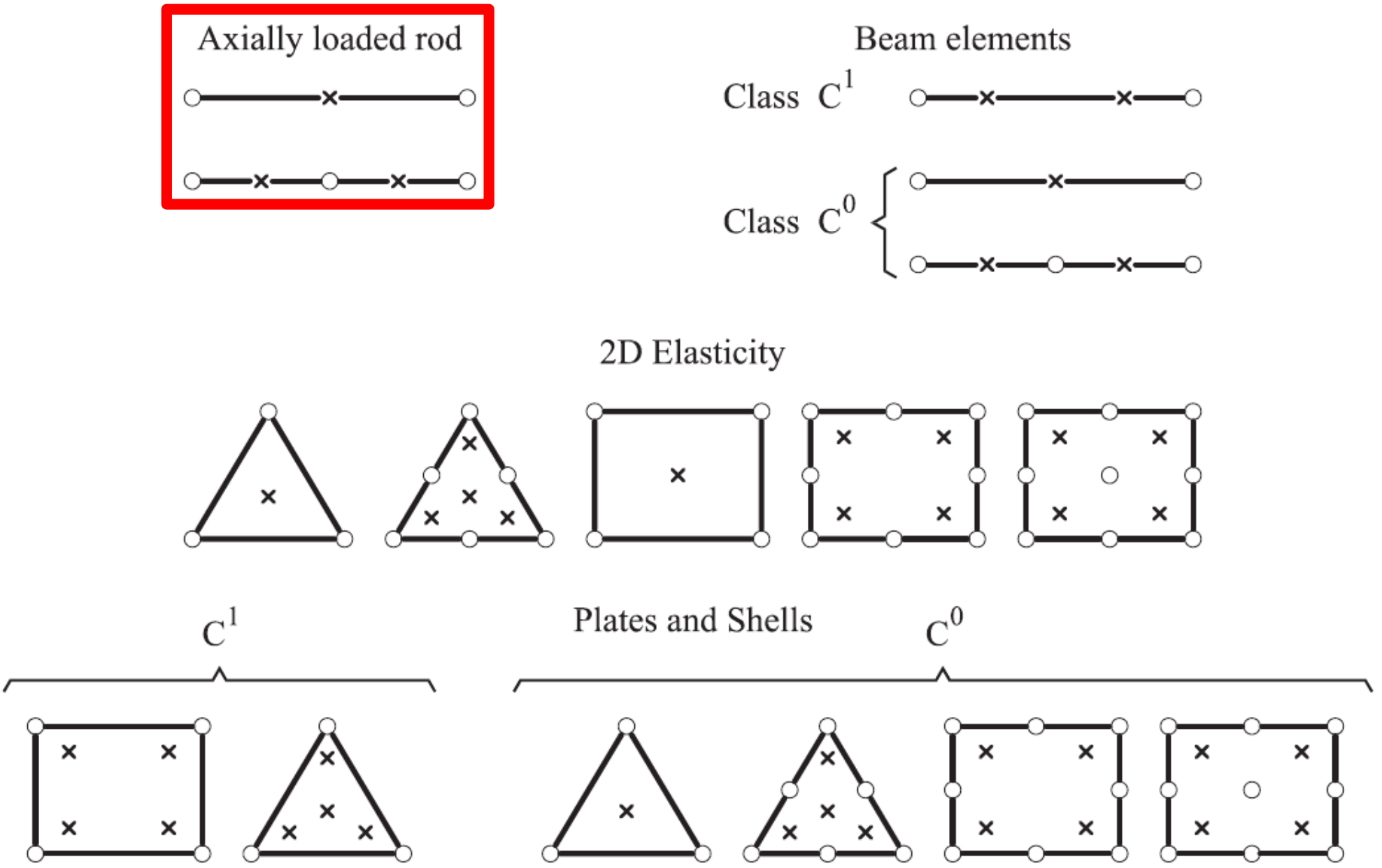
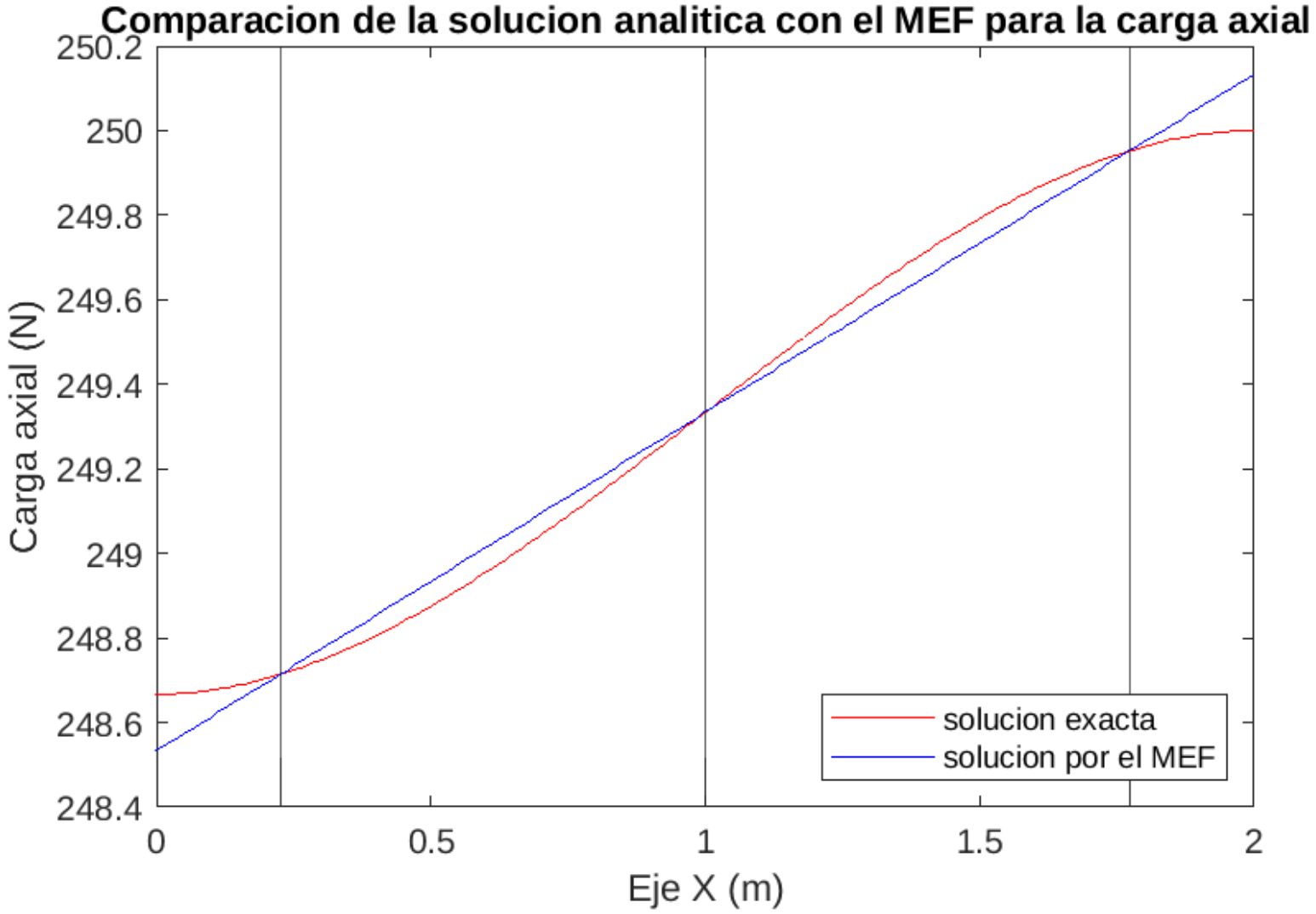


Fig. 6.12 Optimal points for computation of strains and stresses in some 1D and 2D elements

Aquí se está resolviendo el mismo problema de la barra sometida a una carga axial b, pero esta vez $b(x) = x^2 - 2x$. El problema se está resolviendo con un solo EF de 4 nodos. Observe que los mejores puntos para estimar los esfuerzos corresponden a los 3 puntos de integración de Gauss-Legendre (los puntos de Barlow).



En: https://github.com/diegoandresalvarez/elementosfinitos/blob/master/codigo/1D/
Ver: `EF_barra_n_nodos/barra_con_carga_axial_4_nodos_gauss_legendre.m`

Organización básica de un programa de EFs de barra

SUBROUTINE **INPUT**
Input data defining geometry and mechanical properties.

Input data:
- Element type
- Mesh topology and nodal coordinates
- Material properties
- Boundary conditions
- Coordinates and weights of Gauss quadrature

SUBROUTINE **STIFFNESS**
Evaluate the stiffness matrix for each element

Compute at each Gauss point r :
- Material properties (EA)
- Derivatives $\frac{\partial N_i}{\partial \xi}$
- $J^{(e)} = \sum_i \frac{\partial N_i}{\partial \xi} x_i$
- $\mathbf{B}$

Compute:
- $\mathbf{K}^{(e)} = \sum_r [J^{(e)} \mathbf{B}^T (EA) \mathbf{B}]_r W_r$

SUBROUTINE **LOAD**
Evaluate the equivalent nodal force vector for each element

Compute at each Gauss point r :
- Shape functions N_i
- $J^{(e)} = \sum_i \frac{\partial N_i}{\partial \xi} x_i$

Compute:
- $\mathbf{f}^{(e)} = \sum_r [J^{(e)} \mathbf{N}^T \mathbf{t}]_r W_r$

SUBROUTINE **SOLVE**
Assembly and solution of
$\mathbf{K}\ \mathbf{a} = \mathbf{f}$

Gaussian elimination [Ral]
Frontal method [HO2]
Profile solver [ZTZ], etc.

SUBROUTINE **STRESS**
Evaluate strains and stresses for each element.

$\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}$
$\boldsymbol{\sigma} = \mathbf{D}\ \mathbf{B}\ \mathbf{a}$

STOP

Convergencia de la solución

4.3.1.2 Convergence property of the FEM

For complex problems in general, the exact or analytical solution (if it exists) cannot be written in the form of a combination of monomials. Therefore, the FEM using polynomial shape functions will not be able to produce the exact solution for such a problem. The question now is, how can one ensure that the FEM can produce a *good* approximation of the solution for a complex problem? The insurance is given by the *convergence* property of the FEM, which states that the FEM solution will converge to the exact solution that is continuous at arbitrary accuracy when the element size becomes infinitely small, and as long as the complete linear polynomial basis is included in the basis to form the FEM shape functions. The theoretical background for this convergence feature of the FEM is due to the fact that any continuous function can always be approximated by a first order polynomial with a second order refinement error. This fact can be revealed by using the local Taylor expansion, based on which a continuous (displacement) function $u(x)$ can always be approximated using the following equation:

$$u = u_i + \left.\frac{\partial u}{\partial x}\right|_i (x - x_i) + O(h^2) \tag{4.45}$$

where h is the characteristic size that relates to $(x-x_i)$, or the size of the element.

According to Eq. (4.45), one may argue that the use of a constant can also reproduce the function u, but with an accuracy of $O(h^1)$. However, the constant displacements produced by the elements will not be continuous in between elements, unless the entire displacement field is constant (rigid movement), which is trivial. Therefore, to guarantee the convergence of a continuous solution, a complete polynomial up to at least the first order is used.

G.R. Liu and S.S. Quek (2014). The Finite Element Method: A Practical Course. 2nd edition. Butterworth-Heinemann

Convergencia de la solución

4.3.1.3 Rate of convergence of FEM results

The Taylor expansion up to the order of p can be given as

$$u = u_i + \left.\frac{\partial u}{\partial x}\right|_i (x - x_i) + \left.\frac{1}{2!}\frac{\partial^2 u}{\partial x^2}\right|_i (x - x_i)^2 + \cdots + \left.\frac{1}{p!}\frac{\partial^p u}{\partial x^p}\right|_i (x - x_i)^p + O\left(h^{p+1}\right) \tag{4.46}$$

If the complete polynomials up to the pth order are used for constructing the shape functions, the first $(p+1)$ terms in Eq. (4.46) will be reproduced by the FEM shape function. The error is of the order of $O(h^{p+1})$; the order of the rate of convergence is therefore $O(h^{p+1})$. For linear elements we have $p=1$, and the order of the rate of convergence for the displacement is therefore $O(h^2)$. This implies that if the element size is halved, the error of the results in displacement will be reduced to one quarter: known as *quadratic convergence*.

G.R. Liu and S.S. Quek (2014). The Finite Element Method: A Practical Course. 2nd edition. Butterworth-Heinemann

Requisitos **necesarios** para la convergencia de la solución (1/3)

- **Condición de continuidad:** el campo de desplazamientos debe ser continuo al interior de cada elemento
- **Condición de derivabilidad:** la aproximación polinómica escogida debe ser derivable al menos hasta el orden de las derivadas que aparecen en el PTV. Por ejemplo en:

$$\iiint_{V^{(e)}} \delta\varepsilon^{(e)}(x)\sigma^{(e)}(x)\,\mathrm{d}V = \int_{x_1^{(e)}}^{x_n^{(e)}} \delta u^{(e)}(x) b^{(e)}(x)\,\mathrm{d}x + \sum_{i=1}^{n} \delta u_i^{(e)} R_i^{(e)}$$

$$\boldsymbol{K}_{pq}^{(e)} \approx \sum_{m=1}^{M} \frac{\mathrm{d}N_p(\xi_m)}{\mathrm{d}\xi} E^{(e)}(\xi_m) A^{(e)}(\xi_m) \frac{\mathrm{d}N_q(\xi_m)}{\mathrm{d}\xi} \frac{1}{J^{(e)}(\xi_m)} w_m$$

Solo aparecen derivadas de primer orden, por lo que las funciones de forma lineales son suficientes.

Requisitos **necesarios** para la convergencia de la solución (2/3)

Condición de integrabilidad:

Las funciones de forma deben ser tales que las integrales que aparecen en el PTV sean integrables. La derivada de orden *m* de una función es integrable si sus *m-1* primeras derivadas también lo son. Por lo tanto, si en el PTV aparecen derivadas de los desplazamientos de orden *m*, dichos desplazamientos, y por consiguiente las funciones de forma utilizadas para aproximarlos, deben tener continuidad de clase C_{m-1}.

$$\int_{x_1^{(1)}}^{x_2^{(1)}} \left(\frac{dN_1^{(1)}}{dx}(EA)^{(1)} \frac{dN_1^{(1)}}{dx} u_1^{(1)} + \frac{dN_1^{(1)}}{dx}(EA)^{(1)} \frac{dN_2^{(1)}}{dx} u_2^{(1)} \right) dx - \\ - \int_{x_1^{(1)}}^{x_2^{(1)}} N_1^{(1)} t_x \, dx - F_{x_1}^{(1)} = 0$$

Observe que en nuestro caso existen derivadas de orden 1, por lo tanto la función de forma debe ser al menos de orden C_0.

Requisitos **necesarios** para la convergencia de la solución (3/3)

- **Condición de parcela:**
  - Condición de deformación constante: A medida que la malla de elementos finitos se refina, las condiciones dentro de cada elemento se aproximarán más a las de un estado de deformación constante.
  - Condición de sólido rígido: Si el elemento se mueve como un sólido rígido, el desplazamiento es el mismo para todos los puntos y la deformación dentro del elemento debe ser nula en su interior. Esta condición se satisface cuando

$$\sum_{i=1}^{n} N_i(x) = 1 \text{ para todo } x \text{ al interior del elemento}$$

$$\sum_{i=1}^{n} \frac{\mathrm{d}N_i(x)}{\mathrm{d}x} = 0 \text{ para todo } x \text{ al interior del elemento}$$

Requisitos **deseables** para la convergencia de la solución (1/5)

Condición de compatibilidad:

Los movimientos característicos del problema, tales como desplazamientos en problemas de elasticidad, desplazamientos y giros en placas y láminas deben ser continuos entre elementos (los elementos que satisfacen este principio se llaman **elementos compatibles** o **conformes** y si no lo hacen **elementos incompatibles** o **no conformes**). Se sugiere refinar progresivamente la malla.

Requisitos **deseables** para la convergencia de la solución (2/5)

Condición del polinomio completo:

La aproximación por elementos finitos depende del polinomio completo de mayor grado contenido en las funciones de forma. Así, la aproximación será óptima si todos sus términos forman un polinomio completo y no lo será en caso contrario, ocurriendo entonces que los términos adicionales a los del polinomio completo introducen variables que no contribuyen de manera significativa a una mayor aproximación del elemento.

Es deseable que las funciones de forma del elemento sean polinomios completos y, en caso de que esto no sea factible, como sucede con frecuencia, que el número de términos adicionales a los del polinomio completo sea el menor posible.

Requisitos **deseables** para la convergencia de la solución (3/5)

Condición de estabilidad:

- El rango correcto de la matriz de rigidez de un elemento aislado y sin vinculaciones externas debe ser igual al número de movimientos de sólido rígido de un elemento.
- Rigid body mode = modos de deformación de energía nula: son los correspondientes a los valores propios nulos de la matriz de rigidez del elemento

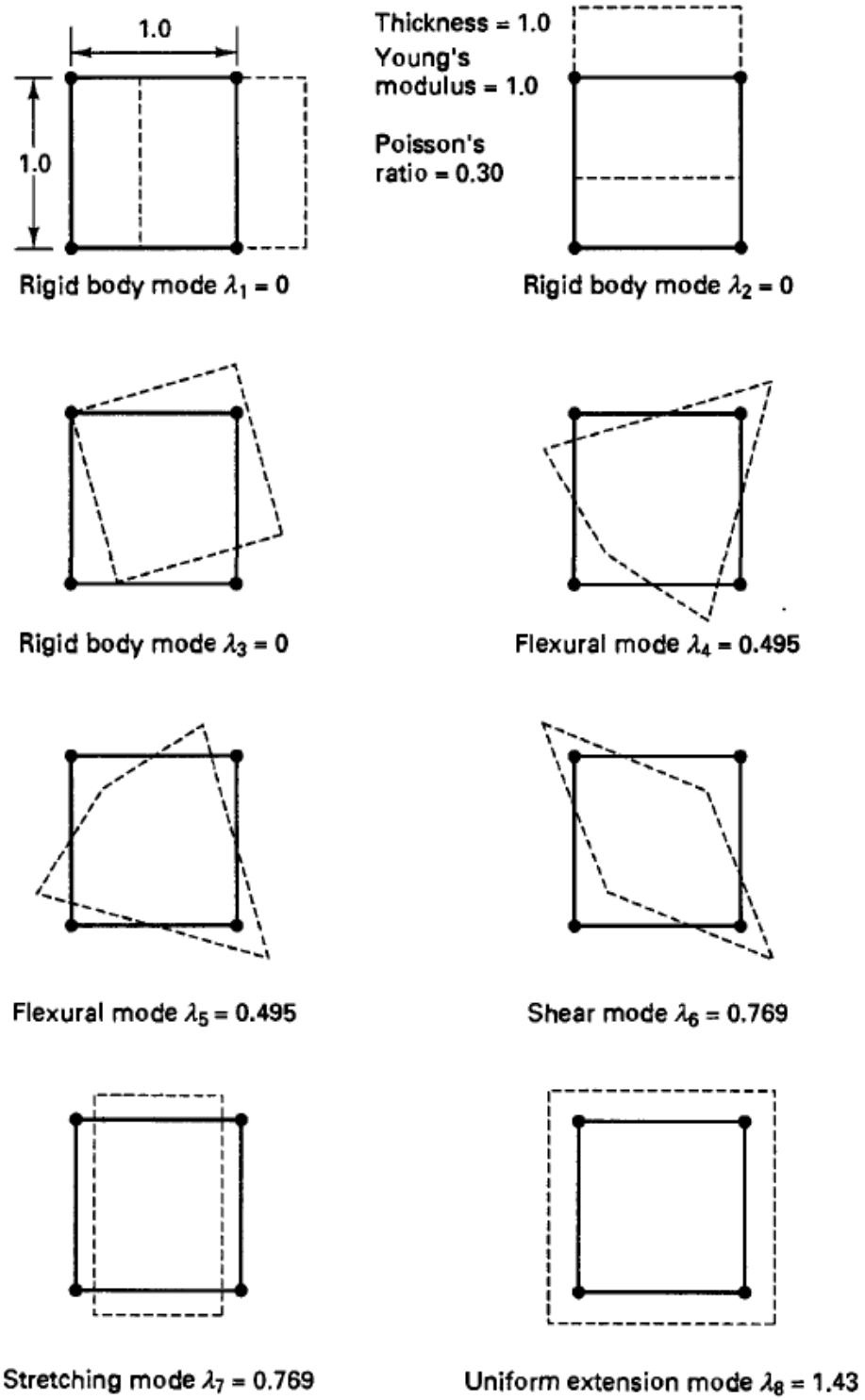


Figure 4.10 Eigenvalues and eigenvectors of four-node plane stress element

Requisitos **deseables** para la convergencia de la solución (4/5)

En la práctica sucede con frecuencia que debido a la evaluación inexacta de algunos términos de $\mathbf{K}^{(e)}$ (por ejemplo por medio de técnicas de integración reducida), se introducen en el elemento mecanismos internos adicionales a los de sólido rígido. Dichos mecanismos no son deseables y un elemento “óptimo” debería estar libre de ellos.

Requisitos **deseables** para la convergencia de la solución (5/5)

Condición de invarianza:

- Un elemento no debe tener direcciones preferentes
- La invarianza de un elemento está garantizada si sus funciones de forma son polinomios completos.

Consideraciones sobre la compatibilidad y equilibrio de la solución

La solución por el MEF es aproximada y, por consiguiente, en general no satisface los requisitos de equilibrio y compatibilidad que serían exigibles a la solución exacta. Sin embargo, en el MEF:

- La solución es compatible dentro de los elementos
- La solución puede ser o no ser compatible a lo largo de los contornos interelementales
- La compatibilidad se satisface siempre en los nodos
- El equilibrio de fuerzas y momentos se satisface siempre en los nodos.
- Normalmente no existe equilibrio de tensiones entre elementos. Por lo tanto al final del análisis se debe realizar una interpolación entre los esfuerzos del elemento.
- Los esfuerzos no están en equilibrio en el interior del elemento.

Condiciones para la convergencia de los elementos isoparamétricos

Los elementos isoparamétricos satisfacen:

- la condición de continuidad del campo geométrico de desplazamientos
- el patch test (criterio de parcela)
- la condición de invarianza
- la condición de compatibilidad de desplazamientos

Tipos de error en la solución de elementos finitos

- Error de discretización
- Error de aproximación de la geometría
- Error en el cálculo de las integrales del elemento
- Errores en la solución del sistema de ecuaciones
- Errores asociados a la ecuación constitutiva

Bibliografía

- Oñate (2009). *Structural Analysis with the Finite Element Method - Linear Statics*, Volume 1. Springer. https://doi.org/10.1007/978-1-4020-8733-2