Rayleigh-Ritz method for SCSF and SSSF plate bending with patch loads

Both SCSF and SSSF plates are **Lévy-type boundary conditions**, meaning exact single-series solutions exist— ScienceDirect often making full double-series Rayleigh-Ritz unnecessary. However, Ritz remains valuable for complex loadings and provides a general framework. This report delivers explicit beam functions, stiffness matrix formulations, Python implementation, and validation benchmarks for practical engineering use.

Beam characteristic functions and their eigenvalues

The Rayleigh-Ritz method approximates plate deflection as $\mathbf{w(x,y) = \Sigma_m \Sigma_n A_{mn} X_m(x) Y_n(y)}$, where X_m and Y_n are beam functions satisfying boundary conditions on each edge. Academia.edu ScienceDirect These functions come from beam vibration theory, with normalized coordinate $\boldsymbol{\xi = x/L}$.

Simply supported (SS) beam functions take the simplest form:

$\varphi_n(\xi) = \sin(n\pi\xi)$, with eigenvalues $\beta_n L = n\pi$ (n = 1, 2, 3,...)

Clamped-free (cantilever) beam functions satisfy $\cos(\beta L)\cosh(\beta L) + 1 = 0$:

$\varphi_n(\xi) = [\cosh(\beta_n L\xi) - \cos(\beta_n L\xi)] - \sigma_n[\sinh(\beta_n L\xi) - \sin(\beta_n L\xi)]$

where $\sigma_n = (\sinh \beta_n L - \sin \beta_n L)/(\cosh \beta_n L + \cos \beta_n L)$. The first five eigenvalues are $\mathbf{\beta_1 L = 1.8751, \beta_2 L = 4.6941, \beta_3 L = 7.8548, \beta_4 L = 10.996, \beta_5 L = 14.137}$. vibrationdata ScienceDirect For n > 3, the approximation $\beta_n L \approx (2n-1)\pi/2$ holds within 0.5%.

Simply supported-free (S-F) beam functions satisfy $\tan(\beta L) = \tanh(\beta L)$:

$\varphi_n(x) = A_n[\sin(\beta_n x) - (\sin \beta_n L/\sinh \beta_n L)\cdot\sinh(\beta_n x)]$

The eigenvalues are $\mathbf{\beta_1 L = 3.9266, \beta_2 L = 7.0686, \beta_3 L = 10.210, \beta_4 L = 13.352, \beta_5 L = 16.493}$. These identical roots also apply to clamped-simply supported (C-S) beams, though with different eigenfunctions.

Clamped-simply supported (C-S) beam functions share the same characteristic equation:

$\varphi_n(x) = [\sinh(\beta_n x) - \sin(\beta_n x)] - \sigma_n[\cosh(\beta_n x) - \cos(\beta_n x)]$

with $\sigma_n = (\sinh \beta_n L - \sin \beta_n L)/(\cosh \beta_n L - \cos \beta_n L)$.

Boundary condition	Characteristic equation	$\beta_1 L$	$\beta_2 L$	$\beta_3 L$	Asymptotic formula
Simply supported	$\sin(\beta L) = 0$	3.142	6.283	9.425	$n\pi$
Clamped-free	$\cos(\beta L)\cosh(\beta L) = -1$	1.875	4.694	7.855	$(2n-1)\pi/2$
Clamped-clamped	$\cos(\beta L)\cosh(\beta L) = 1$	4.730	7.853	10.996	$(2n+1)\pi/2$
Clamped-pinned / Pinned-free	$\tan(\beta L) = \tanh(\beta L)$	3.927	7.069	10.210	$(4n+1)\pi/4$

Stiffness matrix formulation from strain energy

The strain energy for a Kirchhoff thin plate is:

$$\mathbf{U = (D/2) \iint [(\partial^2 w/\partial x^2)^2 + (\partial^2 w/\partial y^2)^2 + 2\nu(\partial^2 w/\partial x^2)(\partial^2 w/\partial y^2) + 2(1-\nu)(\partial^2 w/\partial x \partial y)^2]\, dA}$$

where $D = Eh^3/12(1-\nu^2)$ is flexural rigidity. ScienceDirect Substituting the series form and applying the Ritz minimization $\partial U/\partial A_{mn} = 0$ yields the linear system **K·A = F**.

The stiffness coefficient connecting modes (m,n) and (p,q) is:

$$\mathbf{K_{mn,pq} = D[I''_{mp} J^0_{nq}/a^3b + I^0_{mp} J''_{nq}/ab^3 + \nu(I''_{mp} J^0_{nq}/ab + I^0_{mp} J''_{nq}/ab)/ab + 2(1-\nu)\, I'_{mp} J'_{nq}/a^2b^2]}$$

where the fundamental integrals are:

- $I^0_{mp} = \int_0^a X_m(x)X_p(x)\, dx$ (orthogonality integral)
- $I'_{mp} = \int_0^a X'_m(x)X'_p(x)\, dx$ (first derivative product)
- $I''_{mp} = \int_0^a X''_m(x)X''_p(x)\, dx$ (curvature product)

For **simply supported functions**, orthogonality gives $I^0_{mp} = (a/2)\delta_{mp}$, and the stiffness matrix becomes **diagonal**—a major computational simplification. For clamped or free boundaries, cross-coupling integrals are non-zero when (m+p) is even, creating a sparse but non-diagonal structure.

Tabulated integral values from Young & Felgar (1949) and Blevins for clamped-clamped beams Wiley Online Library (normalized to unit length):

m	$\int \varphi_m^2 d\xi$	$\int (\varphi''_m)^2 d\xi$	$\int (\varphi'_m)^2 d\xi$
1	1.0	500.6	12.30
2	1.0	3803.5	46.05
3	1.0	14617	118.9
4	1.0	39944	231.4

Cross-coupling integrals $\int \varphi''_m \varphi''_n d\xi$ for clamped beams: modes (1,3) give −228.7, modes (2,4) give −1869.4. These non-zero off-diagonal terms create the coupling that requires solving the full system.

Handling rectangular and circular patch loads

For a rectangular patch load of intensity q_0 over region $[x_1,x_2] \times [y_1,y_2]$, the generalized force vector separates as:

$$\mathbf{F_{mn} = q_0 \cdot \int_{x1}^{x2} X_m(x)dx \cdot \int_{y1}^{y2} Y_n(y)dy}$$

For simply supported functions, the integral has closed form: $\int_{x1}^{x2} \sin(m\pi x/a)dx = (a/m\pi)[\cos(m\pi x_1/a) - \cos(m\pi x_2/a)]$.

Circular patch loads require numerical integration. The most efficient approach uses polar coordinates centered on the patch: $x = xc + r\cdot\cos(\theta)$, $y = yc + r\cdot\sin(\theta)$, with Jacobian $r\, dr\, d\theta$. Integration bounds become $0 \leq r \leq R$, $0 \leq \theta \leq 2\pi$, clipped to plate boundaries.

For quick estimates, an equivalent-area square approximation works: side length $s = \sqrt{\pi}\cdot R$ gives a square with identical area, centered at the same point.

Python implementation for Ritz plate bending

```python
```

```python
import numpy as np
from scipy.integrate import dblquad
from scipy.linalg import solve

class RitzPlateSS:
    """Rayleigh-Ritz solver for simply supported rectangular plate."""

    def __init__(self, a, b, h, E, nu, M, N):
        self.a, self.b, self.h = a, b, h
        self.E, self.nu = E, nu
        self.M, self.N = M, N
        self.D = E * h**3 / (12 * (1 - nu**2))
        self.ndof = M * N

    def idx(self, m, n):
        """Map (m,n) starting at 1 to linear index."""
        return (m - 1) * self.N + (n - 1)

    def mode(self, i):
        """Recover (m,n) from linear index."""
        return i // self.N + 1, i % self.N + 1

    def stiffness_matrix(self):
        """Diagonal stiffness for SSSS plate."""
        K = np.zeros((self.ndof, self.ndof))
        for i in range(self.ndof):
            m, n = self.mode(i)
            alpha = m * np.pi / self.a
            beta = n * np.pi / self.b
            # Analytical result exploiting orthogonality
            K[i, i] = self.D * (self.a * self.b / 4) * (alpha**2 + beta**2)**2
        return K

    def load_vector_rect_patch(self, q0, x1, y1, x2, y2):
        """Load vector for rectangular patch."""
        F = np.zeros(self.ndof)
        for i in range(self.ndof):
            m, n = self.mode(i)
            # Closed-form integrals of sine functions
            Ix = (self.a / (m * np.pi)) * (
                np.cos(m * np.pi * x1 / self.a) -
                np.cos(m * np.pi * x2 / self.a))
            Iy = (self.b / (n * np.pi)) * (
```

```
                np.cos(n * np.pi * y1 / self.b) -
                np.cos(n * np.pi * y2 / self.b))
            F[i] = q0 * Ix * Iy
        return F

    def load_vector_circular_patch(self, q0, xc, yc, R):
        """Load vector for circular patch via numerical integration."""
        F = np.zeros(self.ndof)
        for i in range(self.ndof):
            m, n = self.mode(i)
            def integrand(theta, r):
                x = xc + r * np.cos(theta)
                y = yc + r * np.sin(theta)
                if 0 <= x <= self.a and 0 <= y <= self.b:
                    return r * np.sin(m*np.pi*x/self.a) * np.sin(n*np.pi*y/self.b)
                return 0.0
            result, _ = dblquad(integrand, 0, R, 0, 2*np.pi, epsabs=1e-8)
            F[i] = q0 * result
        return F

    def solve(self, F):
        """Solve for mode coefficients."""
        return solve(self.stiffness_matrix(), F)

    def displacement(self, A, x, y):
        """Compute deflection at point."""
        w = 0.0
        for i, Ai in enumerate(A):
            m, n = self.mode(i)
            w += Ai * np.sin(m*np.pi*x/self.a) * np.sin(n*np.pi*y/self.b)
        return w

# Example: 1m × 1m steel plate, 10mm thick, central patch load
plate = RitzPlateSS(a=1.0, b=1.0, h=0.01, E=200e9, nu=0.3, M=10, N=10)
F = plate.load_vector_rect_patch(q0=10000, x1=0.4, y1=0.4, x2=0.6, y2=0.6)
A = plate.solve(F)
w_center = plate.displacement(A, 0.5, 0.5)
print(f"Central deflection: {w_center*1000:.4f} mm")
```

For **clamped or mixed boundaries**, replace the sine functions with appropriate beam functions and compute stiffness integrals numerically:

```python
```

```
def beam_CF(xi, m):
    """Clamped-free beam function."""
    beta = [1.8751, 4.6941, 7.8548, 10.996, 14.137]
    b = beta[m-1] if m <= 5 else (2*m - 1) * np.pi / 2
    sigma = (np.sinh(b) - np.sin(b)) / (np.cosh(b) + np.cos(b))
    return (np.cosh(b*xi) - np.cos(b*xi) -
            sigma * (np.sinh(b*xi) - np.sin(b*xi)))

def beam_SF(xi, m):
    """Simply supported-free beam function."""
    beta = [3.9266, 7.0686, 10.210, 13.352, 16.493]
    b = beta[m-1] if m <= 5 else (4*m + 1) * np.pi / 4
    return np.sin(b*xi) - (np.sin(b)/np.sinh(b)) * np.sinh(b*xi)
```

Convergence behavior and practical term counts

Convergence depends strongly on both boundary conditions and load type:

Configuration	Load type	Terms needed (M=N)	Relative error
SSSS	Uniform	3–5	< 1%
SSSS	Central patch	8–12	< 1%
SSSS	Point load	15–25	< 2% at distance
CCCC	Uniform	5–8	< 2%
SCSF/SSSF	Patch	10–15	< 2%

Key insight: Deflection converges faster than moments. For accurate stress results, double the term count. Series diverge at concentrated load points—always model point loads as small patches (R ≈ h for a "practical point load").

A convergence study should track relative change:

```python
```

```
for N in [3, 5, 8, 12, 16, 20]:
    plate = RitzPlateSS(a, b, h, E, nu, N, N)
    w_max = compute_max_deflection(plate, load)
    if N > 3:
        rel_change = abs(w_max - w_prev) / w_prev
        if rel_change < 0.01:  # 1% convergence
            break
    w_prev = w_max
```

Simpler alternatives to full Ritz for SCSF and SSSF plates

Both SCSF and SSSF belong to the six **Lévy-type boundary conditions** (two opposite edges simply supported), allowing exact single-series solutions that converge faster than double-series Ritz.

Lévy method assumes $w(x,y) = \Sigma_m Y_m(y)\cdot\sin(m\pi x/a)$, automatically satisfying simply supported edges at x = 0 and x = a. The remaining boundary conditions on y = 0 and y = b reduce to solving a fourth-order ODE for each $Y_m(y)$:

$$Y_m(y) = A_m \cosh(\alpha_m y) + B_m \sinh(\alpha_m y) + C_m \cos(\gamma_m y) + D_m \sin(\gamma_m y)$$

Coefficients come from four boundary equations. For SSSF: simply supported at y = 0 (w = 0, Myy = 0) and free at y = b (Myy = 0, Vy = 0). This yields algebraic equations solvable term-by-term.

Finite integral transform method applies Fourier transforms directly to the governing equation and boundary conditions, converting the PDE to linear algebra without assuming trial functions. Springer Li Rui's group (Dalian University) has published extensive solutions for plates with free edges, though **500+ terms may be needed** for accurate results at free corners.

Gorman's superposition method decomposes the problem into Lévy-type "building blocks"—auxiliary problems with simpler boundaries. Superimposing solutions satisfies all original conditions. Google Books This converges faster than Ritz (40×40 matrices vs. 100×100) and provides upper or lower bounds depending on block selection. Academia.edu Reference: D.J. Gorman, *Vibration Analysis of Plates by the Superposition Method* (World Scientific, 1999).

Recommendation: For SCSF and SSSF under patch loads, start with Lévy solution. Only use full Ritz if edges have elastic restraints or non-standard conditions.

Validation benchmarks for implementation verification

SSSS square plate under uniform load (Timoshenko & Woinowsky-Krieger, Table 35):

Parameter	Formula	Value (a/b=1, ν=0.3)
Max deflection	$w_{max} = \alpha \cdot pa^4/D$	$\alpha = 0.00406$
Max moment Mx	$M_{x,max} = \beta \cdot pa^2$	$\beta = 0.0479$
Max moment My	$M_{y,max} = \gamma \cdot pa^2$	$\gamma = 0.0479$

SSSS square plate under central point load:

- $w_{max} = 0.01160 \cdot Pa^2/D$ ijaers

SSSF plate under uniform load (Roark's Formulas, free edge ratio = 1.0):

- Deflection coefficient: $\mathbf{c_1 = 0.140}$ in $w_{max} = c_1 \cdot pa^4/(Eh^3)$
- Stress coefficient: $c_2 = 0.67$ in $\sigma_{max} = c_2 \cdot pa^2/h^2$

SCSF plate (Gao et al., 2019): For hydrostatic loading, dimensionless deflection $w' = wD/(q_0 b^4)$ at the free edge corners provides benchmark values. FEM validation with ABAQUS shows series truncation dominates error. Wiley Online Library

Convergence verification for SSSS uniform load:

Terms	$w_{max} \times D/(pa^4)$	Error
1×1	0.00416	+2.5%
3×3	0.00406	+0.1%
5×5	0.004062	exact

These values should match your implementation output. Discrepancies beyond 1% indicate coding errors in stiffness assembly or load vector computation.

Conclusion and implementation guidance

The Rayleigh-Ritz method provides a versatile framework for plate bending analysis, ScienceDirect Wiley Online Library but **SCSF and SSSF are special cases with exact Lévy solutions**—consider these first. For general implementation:

- Use **sine functions for simply supported edges** (diagonal stiffness, fast convergence)
- Implement **beam functions numerically** for clamped/free boundaries using tabulated eigenvalues

- Model patch loads with **separable integrals** for rectangles, **polar integration** for circles
- Start with **M = N = 10** terms and verify convergence to 1%
- Validate against Timoshenko tables before tackling novel geometries

The Python code provided handles simply supported plates directly. For SCSF, use $X_m(x) = \sin(m\pi x/a)$ in x and $Y_n(y)$ from C-S beam functions in y; for SSSF, use S-F beam functions in y. Numerical integration of the stiffness integrals via scipy.integrate.quad replaces the closed-form diagonal results.