

# Gauss quadrature

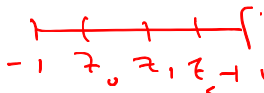
# introduction

- ▶ here we consider integrals of the form

$$\int_{-1}^1 f(x) dx$$

- ▶ quadrature rule

$$Q(f) = \sum_{k=0}^n \underline{w_k} f(\underline{z_k})$$



with quadrature points  $z_k$  and weights  $w_k$

- ▶ compute weights *and* quadrature points such that for all polynomials  $p(x)$  of degree  $2n+1$  the quadrature is exact, i.e.,

$$Q(p) = \int_{-1}^1 p(x) dx$$

- ▶ the Gaussian rules require thus the determination of  $2n+2$  parameters  $z_k, w_k$  for  $k = 0, \dots, n$

## method of unknown coefficients

- ▶ determine rule which is exact for all monomials  $p(x) = x^j$  for  $j = 0, \dots, 2n + 1$

$$\sum_{k=0}^n \underline{w_k z_k^j} = \int_{-1}^1 x^j dx = \frac{2}{j+1}, \quad j = 0, \dots, 2n+1$$

- ▶ this is a polynomial system of equations for  $w_k, z_k$
- ▶ solution of polynomial systems of equations is a topic of algebraic geometry.
- ▶ general approach: Gröbner bases which use a combination of the Euclid and Gauss algorithms
- ▶ here we use a method based on orthogonal polynomials

## example $n = 0$ – midpoint rule

- ▶ general form

$$\underline{Q(f) = w_0 f(z_0)}$$

- ▶ method exact for  $p(x) = 1$  and  $p(x) = x$  leads to two equations

$$w_0 = 2$$

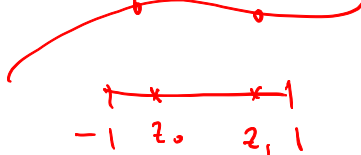
$$w_0 z_0 = 0$$

$$\int_{-1}^1 1 dx = 2$$
$$\int_{-1}^1 x dx = 0$$

- ▶ solution  $w_0 = 2$  and  $z_0 = 0$  which leads to rule

$$\boxed{Q(f) = 2f(0)}$$

example  $n = 1$



- ▶ general form

$$Q(f) = w_0 f(z_0) + w_1 f(z_1)$$

- ▶ method exact for polynomials  $p(x) = 1, x, x^2, x^3$  leads to

$$w_0 + w_1 = 2$$

$$w_0 z_0 + w_1 z_1 = 0$$

$$w_0 z_0^2 + w_1 z_1^2 = \frac{2}{3}$$

$$w_0 z_0^3 + w_1 z_1^3 = 0$$

## solving the equations

- ▶ idea: eliminate 4 unknowns  $w_k$  and  $z_k$  using first 2 equations and introducing two (unknown) parameters  $t$  and  $s$
- ▶ solution of  $w_0 + w_1 = 2$

$$w_0 = 1 + t, \quad w_1 = 1 - t$$

- ▶ solution of  $w_0 z_0 + w_1 z_1 = 0$  (orthogonality of  $w$  and  $z$ )

$$\underline{z_0} = -s w_1 = -s(1 - t), \quad \underline{z_1} = s w_0 = \underline{s(1 + t)}$$

$$+s^2(1+t)(1-t)^2 + s^2(1-t)(1+t)^2$$

- ▶ substituting  $w_k$  and  $z_k$  in third equation  $w_0 z_0^2 + w_1 z_1^2 = \frac{2}{3}$

$$\underline{w_0 z_0^2 + w_1 z_1^2 = 2s^2(1-t^2) = \frac{2}{3}}$$

thus  $s \neq 0$  and  $t^2 \neq 1$

- ▶ substituting  $w_k, z_k$  in fourth equation  $w_0 z_0^3 + w_1 z_1^3 = 0$

$$w_0 z_0^3 + w_1 z_1^3 = \underline{4s^3(1-t^2)t = 0}$$

thus  $t = 0$

- ▶ substitute  $t = 0$  into third equation to get  $s = 1/\sqrt{3}$

- ▶ solution

$$w_0 = w_1 = 1, z_0 = -1/\sqrt{3}, z_1 = 1/\sqrt{3}$$

# composite Gauss rules

$$m=7$$



- quadrature points

$$x_{k+jn} = \frac{z_k + 1}{2}h + jh$$

where  $h = (b - a)/m$

- use Gauss weights  $w_k$  for the interval  $[-1, 1]$

$$Q(f) = \frac{h}{2} \sum_{j=0}^m \sum_{k=0}^n w_k f(x_{k+jn})$$



example  $n = 2$

$$\int_{-1}^{+1} e^{-x} dx$$

```
f = lambda x : np.exp(-x) # integrand
for m in (1,2,4,8,16):
    h = 1.0/m
    Q = 0.0;
    for j in range(m):
        Q += h/2*(f(h*((-1.0/math.sqrt(3)+1)/2 + j)) \
                  + f(h*((1.0/math.sqrt(3)+1)/2 +j)))
    print("m = {:2d},  Q = {:7.6e},  Error = {:4.2e}" \
          .format(m,Q,Q-1 + 1.0/np.e))
```

```
m = 1,  Q = 6.319788e-01,  Error = -1.42e-04
m = 2,  Q = 6.321115e-01,  Error = -9.07e-06
m = 4,  Q = 6.321200e-01,  Error = -5.70e-07
m = 8,  Q = 6.321205e-01,  Error = -3.57e-08
m = 16, Q = 6.321206e-01,  Error = -2.23e-09
```

computing  $n + 1$  quadrature points and weights for larger  $n$

- ▶ for larger  $n$  one could use
  - ▶ Newton's method
  - ▶ algebraic approaches

but these approaches typically take a long time and/or are complicated to implement

- ▶ in the following we discuss an approach based on *Legendre polynomials*  $p(x)$  defined on  $[-1, 1]$

## orthogonality and polynomials



- ▶ recall that two vectors are orthogonal, if their scalar product is zero
- ▶ example:  $v = [1, 2]$  and  $u = [-2, 1]$  are orthogonal
- ▶ we can define orthogonality for polynomials if we have a scalar product

**Definition (scalar product for real polynomials):**

$$(p, q) = \int_{-1}^1 p(x)q(x) dx$$

**Definition (orthogonality for polynomials):**

$p$  and  $q$  are orthogonal if their scalar product  $(p, q) = 0$

- ▶ example:  $p(x) = x$  and  $q(x) = x^2$  are orthogonal as

$$\int_{-1}^1 p(x)q(x) dx = \int_{-1}^1 x \cdot x^2 dx = 0$$

# Legendre polynomials

- ▶ Legendre polynomials  $q_k$  are of the form

$$q_k(x) = x^k + c_{k-1}x^{k-1} + \cdots + c_0$$

- ▶ they are pairwise orthogonal, i.e., if  $k \neq j$  one has

$$\int_{-1}^1 q_k(x) q_j(x) dx = 0$$

$q_k \perp q_j$

- ▶ the first four Legendre polynomials  $q_k$

$$q_0(x) = 1, \quad q_1(x) = x, \quad q_2(x) = x^2 - \frac{1}{3}, \quad q_3(x) = x^3 - \frac{3}{5}x$$

## zeros of Legendre polynomials

### Proposition:

The Legendre polynomial  $q_n$  of degree  $n$  has exactly  $n$  real zeros  $z_k$  satisfying

$$-1 < z_0 < z_1 < \cdots < z_{n-1} < 1$$

### Proof.

- ▶ as degree of  $q_n$  equals  $n$ ,  $q_n$  has  $\leq n$  real zeros ✓
- ▶ as  $q_n$  orthogonal to all  $q_k$  with  $k < n$ ,  $q_n$  orthogonal to any polynomial of degree  $k < n$
- ▶ assume  $x_0, \dots, x_k$  are the zeros (excluding the ones without sign change)
- ▶ then the following integral is either positive or negative:

$$\int_{-1}^1 \prod_{i=0}^{k-1} (x - x_i) q_n(x) dx$$

it cannot be zero

- ▶ thus  $q_n$  is not orthogonal to  $\prod_{i=0}^k (x - x_i)$  contrary to assumption

**Proposition:**

No quadrature formula with  $n + 1$  quadrature points  $z_k$  can be exact for all polynomials of degree  $2n + 2$ .

**Proof**

- ▶ consider  $p(x) = \prod_{k=0}^n (x - z_k)^2$
- ▶ then

$$Q(f) = \sum_{k=0}^n w_k p(x_k) = 0$$

- ▶ but

$$\int_{-1}^1 p(x) dx > 0$$

## Gauss quadrature rules

$$Q(f) = \sum_{k=0}^n w_k f(z_k)$$

- ▶ let  $n + 1$  *quadrature points*  $z_k$  to be the zeros of the Legendre polynomial  $q_{n+1}$
- ▶ select the *quadrature weights*  $w_k$  such that for all polynomials  $p$  of degree up to  $n$

$$Q(p) = \int_{-1}^1 p(x) dx$$

- ▶ compute the weights using either the Lagrange interpolation formula or the method of unknown coefficients

## accuracy of Gauss quadrature

### Proposition:

Gauss quadrature with  $n + 1$  points is exact for all polynomials  $p(x)$  of degree up to  $2n + 1$ , i.e.,

$$Q(p) = \int_{-1}^1 p(x) dx$$

### Proof

- ▶ by construction  $Q(p)$  is exact for all polynomials up to degree  $n$
- ▶ for  $p$  of degree (at most)  $2n + 1$  there exist  $q, r$  of degree  $n$  s.t.

$$p(x) = q(x)q_{n+1}(x) + r(x)$$

- ▶ by linearity and choice of the quadrature points and weights

$$Q(p) = \underline{Q(r)} = \int_{-1}^1 r(x) dx$$

- ▶ as  $q$  is orthogonal to  $q_{n+1}$  one has

$$\int_{-1}^1 p(x) dx = \int_{-1}^1 r(x) dx$$



## construction of Legendre polynomial $q_{n+1}$

- ▶ do this recursively, starting with  $q_0(x) = 1$
- ▶ multiply  $q_k(x)$  with  $x$  and Gram-Schmidt orthogonalisation

$$q_{k+1}(x) = \underline{xq_k(x)} - \sum_{j=0}^k c_j q_j$$

where

$$c_j = \frac{\int_{-1}^1 xq_k(x)q_j(x) dx}{\int_{-1}^1 q_j(x)^2 dx}$$

- ▶ the  $q_k$  are either even or odd, in any case

$$\int_{-1}^1 xq_k(x)^2 dx = 0 \Rightarrow c_k = 0$$

- ▶ if  $j < k - 1$  then the degree of  $xq_j(x)$  is less than  $k$  and thus

$$\int_{-1}^1 xq_k(x)q_j(x) dx = 0$$

$c_j = 0$  for  
 $j \leq k-1$

- ▶ it follows that

$$q_{k+1}(x) = xq_k(x) - c_{k-1}q_{k-1}(x)$$

where

$$c_{k-1} = \frac{\int_{-1}^1 xq_k(x)q_{k-1}(x) dx}{\int_{-1}^1 q_{k-1}^2(x) dx}$$

```
# computing Legendre polynomials qk(x)
```

```
n = 4
```

```
x = sy.Symbol('x')
```

```
qkm1 = 1
```

```
qk    = x
```

```
for k in range(n):
```

```
    qkp1 = sy.simplify(x*qk - sy.integrate(x*qk*qkm1, \
```

```
    (x,-1,1))/sy.integrate(qkm1**2, (x,-1,1))*qkm1)
```

```
    qkm1 = sy.expand(qk)
```

```
    qk    = qkp1
```

```
    print("q{:1d}(x) = {}".format(k+1,qkm1))
```

```
q1(x) = x
```

```
q2(x) = x**2 - 1/3
```

```
q3(x) = x**3 - 3*x/5
```

```
q4(x) = x**4 - 6*x**2/7 + 3/35
```

## computing the quadrature points

```
# compute the Gauss quadrature points  
c = sy.Poly(qkm1).all_coeffs() # Legendre coefficients  
z = np.roots(c) # zeros Legendre fct = quad. pts  
z.sort() # sort by size
```

## computing the quadrature weights

```
# use Lagrange polynomials and sympy
n = z.shape[0]-1
w = np.zeros(n+1)
x = sy.Symbol('x')
print("\n n = {}".format(n))
for j in range(n+1):
    lj = 1
    for k in range(n+1):
        if (k!=j): lj *= (x-z[k])/(z[j]-z[k])
    w[j] = float(sy.integrate(lj,(x,-1,1)))
    print("w{} = {:.4f}".format(j,w[j]),end='    ')
```

*Lagrange*

```
n = 3:
w0 = 0.3479    w1 = 0.6521    w2 = 0.6521    w3 = 0.3479
```

---

## example $n = 3$

```
f = lambda x : np.exp(-x) # integrand
print("n = {}".format(n))
for m in (1,2,4,8,16):
    h = 1.0/m
    Q = 0.0;
    for j in range(m):
        for k in range(n+1):
            Q += h/2*w[k]*(f(h*((z[k]+1)/2 + j)))
    print("m = {:2d},   Q = {:7.6e},   Error = {:4.2e}" \
          .format(m,Q,Q-1 + 1.0/np.e))
```

n = 3

m = 1,	Q = 6.321206e-01,	Error = -3.43e-10
m = 2,	Q = 6.321206e-01,	Error = -1.38e-12
m = 4,	Q = 6.321206e-01,	Error = -5.33e-15
m = 8,	Q = 6.321206e-01,	Error = 1.11e-16
m = 16,	Q = 6.321206e-01,	Error = 2.22e-16



## sign of Gauss weights

### Proposition:

All Gauss weights  $w_k$  are positive.

### Proof

- ▶ let  $p_i(x) = \prod_{k \neq i} (x - z_k)^2$ , a polynomial of degree  $2n - 1$
- ▶ Gauss quadrature with points  $z_k$  is exact for  $p_i$

$$\int_{-1}^1 p_i(x) dx = Q(p_i) = \sum_{k=0}^n w_k p_i(z_k) = w_i p(z_i)$$

- ▶ the integral and  $p(z_i)$  are positive and one gets  $w_i > 0$

see Wikipedia [[https://en.wikipedia.org/wiki/Gaussian\\_quadrature](https://en.wikipedia.org/wiki/Gaussian_quadrature)]

## performance of Gauss rules

**Theorem:**  $Q(f)$  converges to the exact integral for  $n \rightarrow \infty$  and any continuous function  $f$ .

- ▶ Gauss quadrature rules are very reliable and highly accurate
- ▶ composite rules using Gauss weights may be more convenient and, in the case of less smooth functions, may require fewer function evaluations



## convergence rate for Gaussian Quadrature

The basis for the convergence rate is the error formula for any Gaussian quadrature formula:

The error in Gaussian Quadrature is given by:

$$\int_a^b f(x)dx - \sum_{i=0}^n A_i f(x_i) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b w(x)dx$$

where  $\xi$  is some point in the domain of integration and

$$w(x) = \prod_{i=0}^n (x - x_i)^2$$

- ▶ **composite rules** have error  $O(h^{2n+2})$  in this case
- ▶ for  $n = 9$  doubling  $m$ 
  - ▶ doubles the computational effort
  - ▶ reduces error by factor  $2^{20} \approx 10^6$