Metodi numerici Cheat Sheet

by Unobia via cheatography.com/216860/cs/47508/

Sistemi lineari

Jacobi

```
n = size(A,1);
D = spdiag s(d iag (A) ,0, n,n);
N = A-D;
for k = 1:kmax
       x = D\(b-N*x);
        res_rel = norm(b -A* x)/ nor m(b);
       if res_rel <= toll
                break
       end
end
```

Gauss-Seidel

```
M = tril(A);
N = A-M;
for k = 1:kmax
       x = M\(b-N*x);
        res_rel = norm(b -A* x)/ nor m(b);
       if res_rel <= toll
                break
       end
end
```

Gradiente

```
r = b-A*x;
for k = 1:kmax
       z = A*r;
        alpha = r()r'(r'z);
       x = x + alpha*r;
       r = r - alpha*z;
        res_rel = norm(r )/n orm(b);
       if res_rel <= tol
                break
       end
end
```

Quadratura

Trapezi composita

```
x = linspace(a,b,n+1);
y = f(x);
h = (b-a)/n;
Itrap = (h/2) (y(1) + 2sum(y( 2:e nd-1)) + y(end));
```

Simpson composita

```
x = linspace(a,b,2*n+1);
y = f(x);
h = (b-a)/n;
Isimpson = (h/6) (y(1) + 4sum(y( 2:2 :en d-1)) +
2*sum( y(3 :2: end-2)) + y(end));
```

PDE

Differenze finite DD

```
N = length(x) - 1;
h = x(2) - x(1);
xM = (x(1:e nd- 1)+ x(2 :en d))/2;
muVec = mu(xM);
d0 = 1/h^2* (mu Vec (1: end-1) + muVec( 2:e nd));
d1 = -1/h^2 *mu Vec (2: end);
d_1 = -1/h^2 *mu Vec (1: end-1);
A = spdiag s([d_1, d0, d1], [-1, 0, 1], N-1, N-1);
b = f(x(2: end -1));
b(1) = b(1) + (1/h^2 )*m uVec(1) *BC(1);
b(end) = b(end) + (1/h^2 )*m uVe c(e nd) *BC(2);
u = A\b;
u = [BC(1); u; BC(2)];
```

Differenze finite ND

```
N = length(x) - 1;
h = x(2) - x(1);
xM = (x(1:e nd- 1)+ x(2 :en d))/2;
d0 = mu/h^2 *([1; 2*ones (N- 1,1)]);
d1 = -mu/h^ 2*o nes (N,1);
d_1 = -mu/h^ 2*o nes (N,1);
A = spdiag s([d_1, d0, d1], [-1, 0, 1], N, N);
b = f(x(1: end -1));
b(1) = 1/2*b( 1)- BC( 1)/h;
b(end) = b(end) + 1/h^2* mu* BC(2);
u = A\b;
u = [u; BC(2)];
```

Elementi finiti DD

```
N = length(x) - 1;
h = diff(x);
xM = x(1:N) + 0.5*h;
muVec = mu(xM)./h;
d = muVec( 1:N-1) + muVec( 2:N);
d_1 = -muVec (2:N);
d1 = -muVec (1: N-1);
A = spdiags([d_1 d d1], [-1 0 1], N-1, N-1);
b = f(x(2: N)).*( h(1 :N-1) + h(2:N) )*0.5;
b(1) = b(1) + muVec( 1)* BC(1);
b(N-1) = b(N-1) + muVec( N-1 )*B C(2);
uDof = A\b;
u = [BC(1); uDof; BC(2)];
```

Elementi finiti ND

```
N = length(x) - 1;
h = diff(x);
xM = x(1:N) + 0.5*h;
muVec = mu(xM)./h;
d = [muVec(1); muVec( 1:N-1) + muVec( 2:N)];
d_1 = -[muVe c(1); muVec( 2:N)];
d1 = -[muVe c(1); muVec( 1:N -1)];
A = spdiags([d_1 d d1], [-1 0 1], N, N);
b = f(x(2: N)).*( h(1 :N-1) + h(2:N) )*0.5;
b = [0.5*f (x( 1)) *h(1) - BC(1); b];
b(end) = b(end) + muVec( end )*B C(2);
uDof = A\b;
u = [uDof; BC(2)];
```

Equazioni non lineari

Newton

```
for k=1:kmax
       x = x0 - m*f(x0 )/d f(x0);
        errx(k) = abs(x- x0) /ab s(x);
        val_f = f(x);
       if val_f< =tolf
                break
       end
       if errx(k )<=tolr
                break
       end
       x0 = x;
```

Newton (cont)

```
> end
```

Punto fisso

```
x_vec(1) = x0;
for k=1:kmax
       x = phi(x0);
        x_v ec(k+1) = x;
        errx(k) = abs(x- x0)/x;
       if errx(k )<=tolr
                break
       end
        val_f = abs(ph i(x )-p hi( x0));
       if val_f< =tolf
                break
       end
       x0 = x;
end
```

Attenzione in versione 2014 l'aggiunta di elementi in un vettore

ODE

Eulero esplicito

```
t = t0:dt:T;
u = zeros( len gth (t),1);
u(1) = u0;
for k = 1:leng th(t) - 1
        u(k+1) = u(k) + dt*f(t (k) ,u(k));
end
t = t';
```

Eulero implicito

```
t = t0:dt:T;
u = zeros( len gth (t),1);
u(1) = u0;
for k = length (t)-1
        u(k+1) = 1/(1 -lambda)u(k);
end
t = t';
```

Eulero sistemi

```
t = t0:dt:T;
dim_u = size(u 0,1);
u = zeros( len gth (t) ,di m_u);
u(1,:) = u0;
for k = 1:leng th(t) - 1
        u(k +1,:) = u(k,:) + dt*f(t (k) ,u( k,:))';
end
t = t';
```

By **Unobia**
cheatography.com/unobia/

Not published yet.
Last updated 16th December, 2025.