Q1. Write in Latex the following linearized Differential Equations.

The local behavior of the solution to a differential equation near any point (t_c, y_c) can be analyzed by expanding $f(t, y)$ in a two-dimensional Taylor series:

$$f(t, y) = f(t_c, y_c) + \alpha(t - t_c) + J(y - y_c) + ...,$$

Where

$$\alpha = \frac{\partial f}{\partial t}(t_c, y_c), \quad J = \frac{\partial f}{\partial y}(t_c, y_c).$$

The most important term in this series is usually the one involving J, the Jacobian.
For a system of differential equations with n components,

$$\frac{d}{dt}\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, ..., y_n) \\ f_2(t, y_1, ..., y_n) \\ \vdots \\ f_n(t, y_1, ..., y_n) \end{bmatrix},$$

the Jacobian is an n by n matrix of partial derivatives:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_n} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial y_1} & \frac{\partial f_n}{\partial y_2} & \cdots & \frac{\partial f_n}{\partial y_n} \end{bmatrix}$$

The influence of the Jacobian on the local behavior is determined by the solution to the linear system of ordinary differential equations.

$$\dot{y} = Jy.$$

Let $\lambda_k = \mu_k + i\nu_k$ be the eigenvalues of J and $\Lambda = diag(\lambda_k)$ the diagonal eigenvalue matrix. If there is a linearly independent set of corresponding eigenvectors V, then

$$J = V\Lambda V^-1$$

The linear transformation

$$Vx = y$$

transforms the local system of equations into a set of decoupled equations for the individual components of x:

$$\dot{x_k} = \lambda_k x_k$$

. The solutions are

$$x_k(t) = e^{\lambda_k(t-t_c)}x(t_c)$$

A single component $x_k(t)$ grows with t if μ_k is positive, decays if μ_k is negative, and oscillates if ν_k is nonzero. The components of the local solution $y(t)$ are linear combinations of these behaviors.

Q2.Solve the following integrand in Matlab

MATLAB has several different ways of specifying the function to be integrated by a quadrature routine. The anonymous function facility is convenient for a simple, one-line formula. For example,

$$\int_0^1 \frac{1}{\sqrt{1+x^4}}dx$$

can be computed with the statements

f = @(x)1./sqrt(1 + x^4)

Q = quadtx(f, 0, 1).

If we want to compute

$$\int_0^\pi \frac{\sin x}{x}dx,$$

we could try

f = @(x)sin(x)./(x)

Q = quadtx(f ,0 ,pi).

Unfortunately, this results is a division by zero message when f(0) is evaluated and, eventually, a recursion limit error. One remedy is to change the lower limit of integration from 0 to the smallest positive floating-point number, realmin.

Q = quadtx(f,realmin,pi).

The error made by changing the lower limit is many orders of magnitude smaller than roundoff error because the integrand is bounded by one and the length of the omitted interval is less the 10^{-300} Another remedy is to use an M-file instead of an anonymous function. Create a file named **sinc.m** that contains the text

function f = sin(x)

```
if    x == 0

      f = 1

else

     f = sin(x)/x

end
```

Then the statement

```
Q = quadtx(@sinc,0,pi).
```

uses a function handle and computes the integral with no difficulty.
Integrals that depend on parameters are encountered frequently. An example is the *beta* function, defined by

$$\beta(z,w) = \int_0^1 t^{z-1}(1-t)^{w-1}dt.$$

MATLAB already has a **beta** function, but we can use this example to illustrate how to bundle parameters. Create an anonymous function with three arguments.

```
F =@(t,z,w)t^(z − 1) ∗ (1 − t)^(w − 1)
```

Or use an M-file with a name like **betaf.m**.

```
function    f = betaf(t,z,w)

f = t^(z − 1) ∗ (1 − t)^(w − 1)
```

Q3 Write the following Algorithm in Latex to generate the same algorithm:

Algorithm 2: LSHDE44 algorithm for constrained problems

1: initialization: N_{init}, N_{max}, circle memories, probabilities
. $q_l = 1/4$, $l = 1, ..., N$, archive $A = \emptyset$
2: $N := N_{init}$
3: generate an initial population $P = (x_1, x_2, ..., x_N)$
4: evaluate $f(x_s)$, $s = 1, 2, ..., N$
5: evaluate all constrains and $\bar{v}(x_s)$, $s = 1, 2, ..., N$
6: **while** stopping condition not reached **do**
7:     set all S_F and S_C empty: $Q = \emptyset$

for $s = 1$ to N **do**
choose the strategy according to the q_l, $l = 1, 2, 3, 4$
generate F and CR, use respective circle memories
generate a trial vector y_s
evaluate $f(y_s)$
evaluate all constrains and $\bar{v}(x_s)$, $s = 1, 2, ..., N$
if $(\bar{v}(y_s) = 0 \wedge \bar{v}(x_s) = 0 \wedge f(y_s) \leq f(x_s)) \vee$ $\bar{v}(y_s) \leq \bar{v}(x_s)$ **then**
if $\bar{v}(y_s) = 0 \wedge \bar{v}(x_s) = 0$ **then**
store difference $f(x_s)$ - $f(y_s)$
else
store difference $\bar{v}(x_s)$ - $\bar{v}(y_s)$
else if
save F and CR into respective S_F and S_C
increase respective count of successful trial points
increase n_s (for competition of strategies)
insert x_s into archive A
insert y_s into new generation Q
else
insert x_s into new generation Q
end if
end for
P := Q
modify circle memories if needed, use respective S_F and S_C for each pair of memories
$N_(old) := N$, recompute size of population N, eq. (13)
if $N < N_(old)$ **then**
remove superfluous points from population
end if
end while

Table 1: **Q4.Generate the Following Table in Latex to generate the same table**

Table 2: The ICD-metric values found by (a) MOEA/D[3], (c)NSGA-II [22] and AMALGAM [70] on CEC09 test instances[71] for UF1-UF10, in their 30 independent runs.

(a)MMTD,(b)MOEA/D[3],NSGA-II[22]and(d)AMALGAM[70]

CEC'09	Min	Median	Mean	Std	Max	MOEAs
UF1	0.011467	0.015462	0.015364	0.002001	0.020610	a
	0.004499	0.073061	0.078707	0.051734	0.193602	b
	0.051996	0.106873	0.096076	0.024862	0.128739	c
	0.029425	0.059633	0.057992	0.008557	0.070121	d
UF2	0.013187	0.016649	0.017700	0.003114	0.025038	a
	0.018233	0.057095	0.062814	0.032670	0.142833	b
	0.016012	0.019849	0.020050	0.001407	0.023589	c
	0.011432	0.013029	0.013217	0.001367	0.016769	d
UF3	0.023253	0.075123	0.072199	0.021670	0.117247	a
	0.027292	0.238281	0.220884	0.084442	0.319024	b
	0.066353	0.098234	0.097065	017958	0.134235	c
	0.091044	0.135348	0.136503	0.022927	0.199235	d
UF4	0.041134	0.041119	0.040188	0.000576	0.043219	a
	0.062797	0.070280	0.070560	0.004064	0.077784	b
	0.052199	0.054388	0.054551	0.001274	0.056679	c
	0.040359	0.041061	0.041020	0.000332	0.041678	d
UF5	0.321300	0.700783	0.600963	0.137507	0.0719319	a
	0.210113	0.428818	0.426645	0.131033	0.707106	b
	1.523087	1.671735	1.676288	0.099452	1.844279	c
	0.166357	0.171420	0.171810	0.002873	0.178301	d
UF6	0.071023	0.174925	0.176185	0.037850	0.291790	a
	0.244165	0.456803	0.508164	0.142289	0.193602	b
	0.705834	0.762023	0.762271	0.024862	0.831784	c
	0.068589	0.079046	0.078552	0.005998	0.089807	d
UF7	0.010327	0.012968	0.013058	0.008266	0.017348	a
	0.007096	0.057095	0.062814	0.022670	0.142833	b
	0.016012	0.114403	0.020050	0.012407	0.023589	c
	0.103736	0.013029	0.017795	0.001354	0.016769	d
UF8	0.071106	0.089740	0.089411	0.008266	0.103749	a
	0.027292	0.238281	0.220884	0.084442	0.319024	b
	0.088857	0.098234	0.120433	0.030475	0.134235	c
	0.091044	0.135348	0.136503	0.022927	0.199235	d
UF9	0.059715	0.0117414	0.118395	0.045441	0.0179117	a
	0.062797	0.070280	0.070560	0.004064	0.077784	b
	0.088857	0.054388	0.160832	0.001274	0.056679	c
	0.040359	0.041061	0.041020	0.085662	0.041678	d
UF10	0.203804	0.241985	0.260353	0.064216	0.458983	a