Outline of Robot Control Algorithms

The dynamics of a robot manipulator can be written in the form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{1}$$

where τ is the vector of actuation forces/torques (forces for prismatic joints, torques for revolute joints).
Independent Joint Control: If we are given reference (i.e., desired) trajectories $q_d(t) = [q_{1,d}(t), q_{2,d}(t), \ldots, q_{n,d}(t)]^T$, the independent joint control method is to apply a "local" feedback loop at each joint ignoring coupling between joints:

$$\tau = -K_p\tilde{q} - K_d\dot{\tilde{q}} \tag{2}$$

where $\tilde{q}(t) = q(t) - q_d(t)$ and K_p and K_d are diagonal matrices. Note that $q_{i,d}(t)$ is the desired trajectory for the i^{th} joint variable. In general, reference trajectories such as $q_{i,d}(t)$ are found from inverse kinematics (based on desired positions and angular orientations of the end-effector) and trajectory planning (e.g., fitting cubic or LSPB trajectories between desired values of joint variables at discrete points in time). Since K_p and K_d are diagonal matrices, the controller in (2) for τ_i depends only on $q_i - q_{i,d}$ and does not depend on the joint variables or reference trajectories for other joints in the manipulator.

A variant of the independent joint controller is to add a $g(q)$ term to cancel out the gravity-dependent terms in (1), i.e., $\tau = -K_p\tilde{q} - K_d\dot{\tilde{q}} + g(q)$.
Inverse Dynamics Control: The inverse dynamics controller (also called the computed torque method) is based on converting the dynamics (1) into a linear system by defining

$$\tau = D(q)a_q + C(q,\dot{q})\dot{q} + g(q) \tag{3}$$

with a_q being a new control input (i.e., τ to a_q is a control input transformation). With (3), the dynamics (1) becomes simply $\ddot{q} = a_q$. Hence, we can define a controller in terms of a_q as

$$a_q = \ddot{q}_d - K_p\tilde{q} - K_d\dot{\tilde{q}} \tag{4}$$

with K_p and K_d being diagonal matrices (proportional and derivative gains in a PD controller). Then, the dynamics of the closed-loop system form a linear asymptotically stable system:

$$\ddot{\tilde{q}} + K_d\dot{\tilde{q}} + K_p\tilde{q} = 0. \tag{5}$$

The controller defined by (3) and (4) can be thought of as an inner-loop/outer-loop control architecture:

$$\text{Outer-Loop Controller} \quad a_q = \ddot{q}_d - K_p\tilde{q} - K_d\dot{\tilde{q}} \tag{6}$$

$$\text{Inner-Loop Controller} \quad \tau = D(q)a_q + C(q,\dot{q})\dot{q} + g(q) \tag{7}$$

Inverse Dynamics Controller in Task Space: If the reference trajectories are defined in the task space (i.e., in terms of the end-effector) rather than in the joint space (i.e., in terms of joint variables), then the inverse dynamics controller can be written in terms of the end-effector generalized position (i.e., position and angular orientation) X and its desired trajectory $X_d(t)$. If we write $\dot{X} = J(q)\dot{q}$ with a Jacobian matrix $J(q)$, then we have $\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$. With the application of the controller (3), we have $\ddot{q} = a_q$. Therefore, $\ddot{X} = J(q)a_q + \dot{J}(q)\dot{q}$. Now, if we define $a_q = J^{-1}(q)[a_X - \dot{J}(q)\dot{q}]$ with a_X being a new control input, we get $\ddot{X} = a_X$. Hence, we can design a controller in terms of a_X as $a_X = \ddot{X}_d - K_p\tilde{X} - K_d\dot{\tilde{X}}$ where $\tilde{X}(t) = X(t) - X_d(t)$. Thus, the inverse dynamics controller written in terms of task space reference trajectories is of the form:

$$a_X = \ddot{X}_d - K_p\tilde{X} - K_d\dot{\tilde{X}} \tag{8}$$

$$a_q = J^{-1}(q)[a_X - \dot{J}(q)\dot{q}] \tag{9}$$

$$\tau = D(q)a_q + C(q,\dot{q})\dot{q} + g(q) \tag{10}$$

Force Control: If the end-effector is in contact with the environment and experiences a generalized force given as a 6×1 vector F_e (which includes the force and torque on the end-effector), we know from the principle of virtual work that the resulting vector of generalized forces seen by the manipulator joints is $J^T(q)F_e$ where $J(q)$ is the manipulator Jacobian matrix. Therefore, the resulting dynamics when the manipulator is in contact with the environment is:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + J^T(q)F_e = \tau \tag{11}$$

In this case, we define the inverse dynamics controller to be

$$\tau = D(q)a_q + C(q,\dot{q})\dot{q} + g(q) + J^T(q)a_f \tag{12}$$

where a_q and a_f are new control inputs. Therefore, from the dynamics (11), we get:
$\ddot{q} = a_q + D^{-1}(q)J^T(q)(a_f - F_e)$.

Then, in terms of task space coordinates X as described above (under task space inverse dynamics controller), if we define $a_q = J^{-1}(q)[a_X - \dot{J}(q)\dot{q}]$, we get $\ddot{X} = J(q)\ddot{q} + \dot{J}(q)\dot{q} = a_X + J(q)D^{-1}(q)J^T(q)(a_f - F_e)$. The matrix $J(q)D^{-1}(q)J^T(q)$ is called the mobility tensor.

If we define $a_f = F_e$, then we get $\ddot{X} = a_X$. Therefore, we can design a controller in terms of a_X as $a_X = \ddot{X}_d - K_p\tilde{X} - K_d\dot{\tilde{X}}$. With this controller, the closed-loop dynamics written in task space is a linear asymptotically stable system: $\ddot{\tilde{X}} + K_d\dot{\tilde{X}} + K_p\tilde{X} = 0$; hence, the reference tracking control objective would be achieved. However, if we want the manipulator to interact with the environment taking into account the forces/torques that are applied on the environment, then we can do a trade-off of the reference tracking objective and a force control objective by prescribing "stiffness" properties of the end-effector; this is called impedance control. In this control method, we want to make the closed-loop dynamics to be of the form

$$M_d\ddot{\tilde{X}} + K_d\dot{\tilde{X}} + K_p\tilde{X} = -F_e. \tag{13}$$

where M_d, K_p, and K_d are matrices that can be used to set how the manipulator interacts with the environment (i.e., different values of these matrices can be used to interact with different types of objects in the environment). Physically, the matrices M_d, K_p, and K_d can be thought of as desired inertia, damping, and stiffness, respectively. The closed-loop dynamics in equation (13) can be achieved by defining

$$a_X = \ddot{X}_d - M_d^{-1}(K_p\tilde{X} + K_d\dot{\tilde{X}} + F_e). \tag{14}$$

If the stiffness properties of the environment are known, then a desired force on the environment can be applied by using a virtual trajectory that is defined to be some distance into the object in the environment. For example, consider a single direction of motion and a simple model of the environment as $|f_e| = K_e\delta_x$ where δ_x is the deformation of the object, K_e is a scalar coefficient, and f_e is the corresponding force that the environment would exert on the robot (which is numerically equal to the force that the robot would exert on the environment). If x_e denotes the undeformed boundary position of the object, then deformation represented by δ_x corresponds to the object's boundary position having been changed to $x_e + \delta_x$. From (13), we see that in steady state (i.e., when the velocity and acceleration variables are zero), we would have $K_p\tilde{X} = -F_e$. Therefore, considering for simplicity a single direction of motion, if we set a desired position x_v (which is a virtual point located inside the object in the environment), then at steady state, the end-effector would reach a position x where $K_p(x - x_v) = -K_e(x - x_e)$. Therefore, we get $x = \frac{K_p x_v + K_e x_e}{K_p + K_e}$. Hence, if we want to apply a desired force f_{des}, then we should have $f_{des} = K_e(x - x_e) = \frac{K_p K_e}{K_p + K_e}(x_v - x_e)$. Therefore, if we set $x_v = x_e + \frac{K_p + K_e}{K_p K_e} f_{des}$, we would apply the desired force f_{des}, i.e., to apply a desired force f_{des}, we should set the "virtual" reference trajectory to be a distance $\frac{K_p + K_e}{K_p K_e} f_{des}$ inside the object.