## Least squares

First step of the method is the definition of an error metric. Different definitions are possible, and are not necessarily equivalent leading to different results both in algorithmic terms as well in terms of solutions. Usually we ask for a positive definite real function (for evident reasons, like avoidance of error cancelation). Error metric functions are usually non linear henceforth. A couple of very common and almost universally adopted types of are MSE (Mean Squared Error), MAE (Mean Absolute Error) and  weighted error versions of. Nevermore other positive definite functions are possible.

$$\epsilon = \epsilon\left(f, x, obs\right)$$

$$MSE = \epsilon = \sqrt{\left(obs - f\left(x\right)\right)^2} \Leftrightarrow \epsilon^2 = \left(obs - f\left(x\right)\right)^2$$

$$MAE = \epsilon = \left|obs - f\left(x\right)\right|$$

As by the the name recall us, original LS method rely on squared error, we'll focus on now later. Error is then function of independent variables we've to find out.

$$\epsilon^2 = \| \ obs - f\left(x\right) \ \|^2 = \left(obs - f\left(x\right)\right)^2$$

$$\epsilon^2 = \| \ y - \hat{y} \ \|^2 = \left(y - \hat{y}\right)^2$$

Once defined a residuals function, we need to identify which value of parameters x our model depends on, allow for minimizing residuals over given observations. Please note we didn't up to now any assumption about the nature od our parameters vector x, they can be real, integre, discrete, rational, continuous or whatever. That's of the most impotence we'll see in the following. In mathematical term we write:

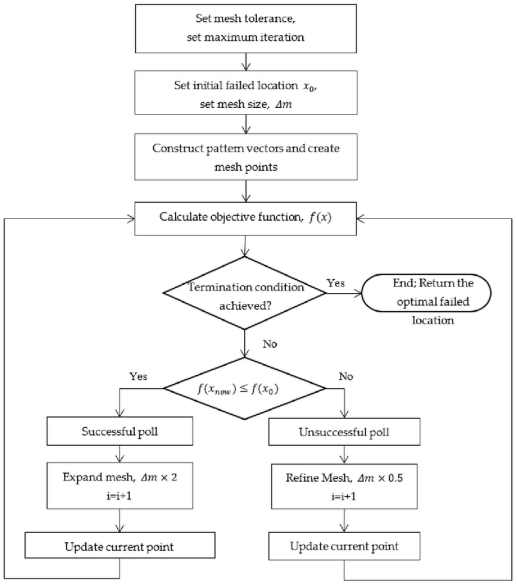
$$x = arg\left(\min_{x}\left(\epsilon^2\right)\right)$$

Use of squared errors sums (SSE) is just one of possible objective (or cost) function possible choice. Alternatively to assure more robust solutions we can use the sum of absolute error (SAE) or any other desired custom objective function.

Minimization problem solution can be achieved in many different ways depending on problem nature (continuous, discrete, piecewise continuous, mixed, linear, non linear,) and approach desired to solve them.

A very general procedure of solution, suitable for discrete values and continuous (after a domain discretization) is direct search and pattern search. Direct search simply defines set

of points where to evaluate cost function and then find minimal values. Similarly des pattern search using a different criteria of domain exploration. These methods have intensive computational burden and consequently slow and inefficient. Other methods exists assuring faster and efficient solutions, but for all of them some cautions shall be used for correctly judging findings, beyond time, computational load, and numerical errors issues.




A further caution shall be arised about the fact, in presence of substantial noise or unmodeled contributes, those vector minimizing is the best choice since it could be biased from unpredicted errors. Usually when dealing with continuous parameters errors spread across different vector elements following their sensitivity respect variations of parameters (partial derivatives). But when parameters are discrete, we can in one case be less prone to noise, on the other case, when large errors occurs, lead to wrong solutions.

Another relevant aspect is the nature of model. it could be linear or non linear.

When dealing with a model that is differentiable respect parameters, we can use estrema condition given by Fermat principle:

$$\frac{\partial \epsilon^2}{\partial x} = \frac{\partial \epsilon^T \epsilon}{\partial x} = \epsilon^T \frac{\partial \epsilon}{\partial x} + \frac{\partial \epsilon}{\partial x}^T \epsilon = 2\epsilon^T \frac{\partial \epsilon}{\partial x} = 0$$

Using linearity relations comes out easily

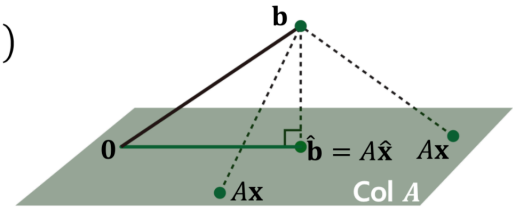
$$2(y - Ax)^T A = 0 \Leftrightarrow \left(A^T A\right) = A^T y$$

This conditions, known also for evident reasons as Normal Equations, is easily seen to be equivalent to a geometrical condition of orthogonality of residuals respect linear plane describing variety of our model on parameters. Indeed the projection on hyperplane of

observations vector gives the minimal distance of them from it, or in other words minimize residual vector.

## Geometric Interpretation of Least Squares

- $\mathbf{b} - A\hat{\mathbf{x}} \perp (x_1\mathbf{a}_1 + x_2\mathbf{a}_2 \cdots + x_p\mathbf{a}_n)$ for any vector $\mathbf{x}$




- Or equivalently,

$$\begin{array}{l} (\mathbf{b} - A\hat{\mathbf{x}}) \perp \mathbf{a}_1 \\ (\mathbf{b} - A\hat{\mathbf{x}}) \perp \mathbf{a}_2 \\ \vdots \\ (\mathbf{b} - A\hat{\mathbf{x}}) \perp \mathbf{a}_m \end{array} \quad \begin{array}{l} \mathbf{a}_1^T(\mathbf{b} - A\hat{\mathbf{x}}) = 0 \\ \mathbf{a}_2^T(\mathbf{b} - A\hat{\mathbf{x}}) = 0 \\ \vdots \\ \mathbf{a}_m^T(\mathbf{b} - A\hat{\mathbf{x}}) = 0 \end{array} \quad \Rightarrow \quad A^T(\mathbf{b} - A\hat{\mathbf{x}}) = 0$$

100

Observe error norm is a strictly positive quantity and a gradient descent method could be applied.

$$x^T J^T J x = \epsilon^2$$

Sensitivity and comparative analysis

The envelope theorem describes how the value of an optimal solution changes when an underlying parameter changes. The process of computing this change is called comparative statics.

The maximum theorem of Claude Berge (1963) describes the continuity of an optimal solution as a function of underlying parameters.

## Optimization problems and constrained optimization

As just seen, any problem of fitting or regression could be seen as an optimization problem, where residual error minimizations is required. What seen so fare regards a free optimization, where in many practical case domain is not free or constraints are imposed on parameters relations. In these case we use the concept of COP (Constrained Optimization Problems). Constraints are of equality or inequality, and they can be linear or non linear. Generally we write

$$\min_x f(x)$$
$$subject\ g(x) = c$$
$$h(x) \geqslant d$$

Equality constraints are also known as hard constraints, where inequality constraints are sometimes known as soft constraints. Solution methods include:

- Substitution (equality constraints)
- Lagrange multipliers (equality constraints only)
- Linear programming
- Non linear programming
- Quadratic programming
- KKT conditions
- Branch and bound

## Other methods of optimization

Other methods of optimization exist, like particle swarm methods, Genetic Algorithms, Surrogate models, Annealing and others. Their use in research applications but also in industrial practices, has become more common in recent years, because of its promising tradeoff between performances and accuracy. Differently from traditional methods, many of them, or most, can effectively exploit advantages of parallelization using computers in networks to speed up processing search.
Particle Swarm Optimization algorithm is an evolutionary, Bio-inspired, useful when dealing with not regular functions when pattern search fails, is Swarm-intelligence-based algorithm that simulates the collective behavior of a swarm of insects/animals, in searching for food. It was first developed by Eberhart and Kennedy in 1995, and since then, it has been modified and enhanced to fit a wide range of engineering and scientific problems, therefore there are many variants of PSO algorithm. However, the Standard PSO algorithm is still the origin from which all variants have been developed.

A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution.

You can apply the genetic algorithm to solve problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear.

The genetic algorithm differs from a classical, derivative-based, optimization algorithm in two main ways, as summarized in the following table.

| Classical Algorithm | Genetic Algorithm |
| --- | --- |
| Generates a single point at each iteration. The sequence of points approaches an optimal solution. | Generates a population of points at each iteration. The best point in the population approaches an optimal solution. |
| Selects the next point in the sequence by a deterministic computation. | Selects the next population by computation which uses random number generators. |

*Simulated annealing (SA)* is a method for solving unconstrained and bound-constrained optimization problems. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy.

At each iteration of the simulated annealing algorithm, a new point is randomly generated. The distance of the new point from the current point, or the extent of the search, is based on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective, but also, with a certain probability, points that raise the objective. By accepting points that raise the objective, the algorithm avoids being trapped in local minima in early iterations and is able to explore globally for better solutions.

### *Surrogate optimization*

*Surrogate optimization* (SO) is an optimization methodology applied with *black-box models* that are *computationally expensive* to evaluate. By iteratively constructing *surrogate models* which can be evaluated quickly compared to the black-box model, the optimizer can perform a wider search with more evaluations in less time, increasing the chance of finding a global optimum. This approach is widely used in applications such as **design optimization**.

A black-box model relates inputs to outputs without exposing the inner workings of the model. Engineers use surrogate models to identify optimal parameters for black-box models that require running a simulation, training a machine learning model, or solving FEA/CFD models where the parameters are both continuous and discrete.

An alternative approach is to build a surrogate model, such as a **reduced order model (ROM)**, and then apply optimization to the surrogate model. If a ROM model is close enough to real model chances to find an optimal solution for original problem relying on it, are good.

ROM: Reduced Order Models

Use of reduced order models is a widely diffused practice in engineering since dawn of applied physics and applied computer science. Different approaches are possible

- Discarding states that do not contribute to the system dynamics, such as structurally disconnected states or canceling pole-zero pairs.
- Discarding low-energy states that contribute relatively little to system dynamics.
- Focusing on a particular frequency region and discarding dynamics outside that region. For example, if your control bandwidth is limited by actuator dynamics, discard higher-frequency dynamics.
-

| Approach | Command Line | Model Reducer App and Reduce Model Order Live Editor Task |
|---|---|---|
| **Simplification** — Reduce model order exactly by canceling pole-zero pairs or eliminating states that have no effect on the overall model response | • `sminreal` — Eliminate states that are structurally disconnected from the inputs or outputs.<br>• `minreal` — Eliminate canceling or near-canceling pole-zero pairs from transfer functions. Eliminate unobservable or uncontrollable states from state-space models.<br>• `xelim` — Eliminate states explicitly. | Pole-Zero Simplification method — Eliminate:<br>• Structurally disconnected states<br>• Unobservable or uncontrollable states from state-space models<br>• Canceling or near-canceling pole-zero pairs from transfer functions |
| **Approximation** — Compute a lower-order approximation of your model. | `reducespec` — Create a model order reduction task for ordinary LTI and sparse LTI models. | Balanced Truncation method — Discard states that have relatively low effect on the overall model response. |
| **Modal Decomposition** — Eliminate poles and zeros that fall outside a specific area of interest. | • `freqsep` — Separate model into slow and fast dynamics around a specified cutoff frequency.<br>• `stabsep` — Separate model into stable and unstable dynamics. | Mode Selection method — Select frequency range of interest and discard dynamics outside that range. |

| | | |
|---|---|---|
| | • `modalsep` – Separate model into modal components. | |

Other techniques of model reduction include

- Ignoring degree of freedom
-

### Are least squares better and model finding a worthy?

In spite of large set of available tools for global optimization today available, most of them have a computational burden not negligible, making LSQ method, when applicable and available, best choice in terms of solution reliability and rapidity of resolution, keeping it as main reference for most of applicative problems in research and industrial filed. Key concerns about regards the availability of a model or at least a suitable knowledge of model nature (mainly linearity) respect desired estimation parameters. As nowadays happens more and more frequently, lack or dropping of this point usually lead to the adoption of alternative methods, sometimes more inspired to machine learning methods, whose in case of linearity would be equivalent, but in more general case are expected to be, maybe not as much performing, but likely more robust.

### Problem feasibility

The *satisfiability problem*, also called the *feasibility problem*, is just the problem of finding any feasible solution at all without regard to objective value. This can be regarded as the special case of mathematical optimization where the objective value is the same for every solution, and thus any solution is optimal. Some problems require you to find a point that satisfies all constraints, with no objective function to minimize. If constraints cause a problem to be infeasible, you might want to find a subset of the constraints that is infeasible, but removing any member of the subset makes the rest of the subset feasible. The name for such a subset is *Irreducible Infeasible Subset of Constraints,* abbreviated IIS. Conversely, you might want to find a maximum cardinality subset of constraints that is feasible. This subset is called a *Maximum Feasible Subset,* abbreviated MaxFS. The two concepts are related, but not identical. A problem can have many different IISs, some with different cardinality.

Many optimization algorithms need to start from a feasible point. One way to obtain such a point is to relax the feasibility conditions using a slack variable; with enough slack, any starting point is feasible. Then, minimize that slack variable until the slack is null or negative.

### Tracking

Another usually arising problem type coming up naturally when dealing with time dependent or dynamic problem is related to tracking conditions.

*Equality constraints*

Substitution is one of most traditional used methods, and one of most easy, but less general and difficult to manage. Indeed often even a linear problem after substitution becomes non linear and harder to solve.
The other traditional method is the use of lagrange multipliers, converting a constrained problem into a unconstrained one by introduction of additional variables (multipliers). Both them apply only in case of equality constraints for evident reasons.

*Inequality constraints*

All other mentioned methods apply to inequality constraints (then to equality as limit case).

## Non linear problems

When the function is nonlinear, we need to solve a nonlinear system that could converge to local solution and requiring an initial value. Local solution at which one converge is not necessarily a global minima.

- A *local* minimum of a function is a point where the function value is smaller than or equal to the value at nearby points, but possibly greater than at a distant point.
- A *global* minimum is a point where the function value is smaller than or equal to the value at all other feasible points.

The region of points that gradient makes converge to is usually called basin of attraction. This occurs bot in one as well in many dimensions. Usually this is done using some iterative numerical method like gauss-Newton or Levenberg-Marquardt illustrated below.

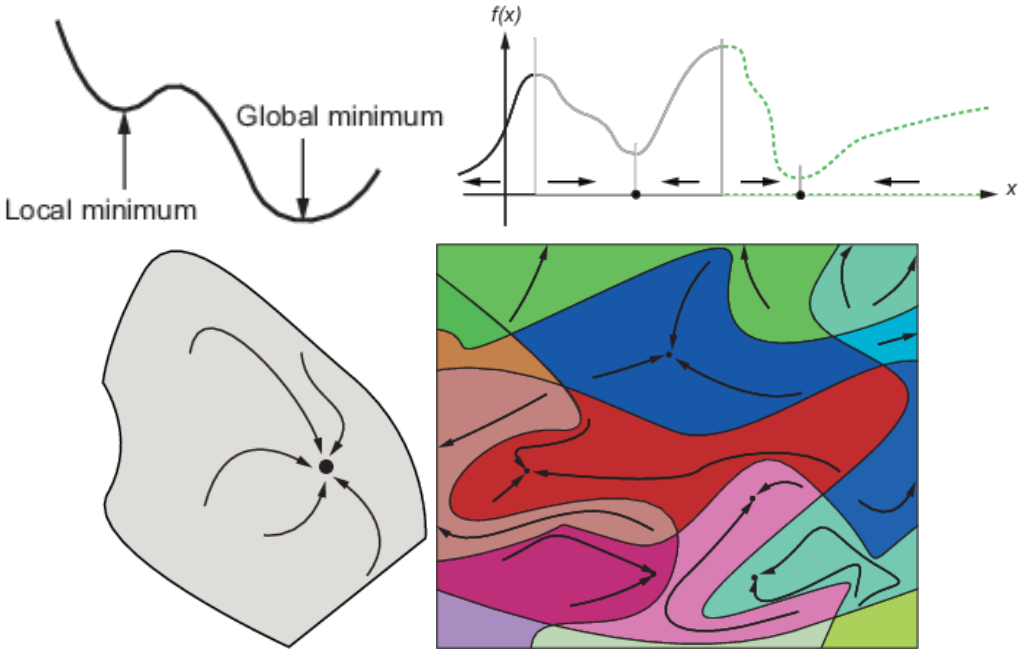
$$\left(J^T J\right) x = J^T \left(y - f\left(x\right)\right)$$

or its damped version

$$\left(J^T J + \lambda I\right) x = J^T \left(y - f\left(x\right)\right)$$
$$\left(J^T J + \lambda \cdot diag\left(J^T J\right)\right) x = J^T \left(y - f\left(x\right)\right)$$

$$\Delta x = -\left(J^T J\right) J^T r$$

In all these case we assume is possible to get partial derivatives of model over parameters. We can see it as a linearized model approximation, where so called design matrix is the jacobian of model over parameters and applied and updated iteratively at each step. As any numerical and approximated method risk of divergence exist if initial point is not close enough.




In case model is linear the solution is unique and obtainable in one step, since we just deal with a overdetermined linear system of equations, solvable using some numerical technique, like normal equations or using pseudo inverse or with some decomposition technique like SVD decomposition or QR decomposition. in literature usually we see the use of normal equations is prominent, when conditioning number and observation matrix rank are satisfactory

$$Ax = b$$

$$A^T Ax = A^T b$$

or

$$x = \text{pinv}\ (A)\, b$$

### Constrained least square

What we've seen so far do not account for any constraint, so parameters are free to vary in their full range of definition. Nonetheless there are several case this is not desirable and constrains shall be imposed to respect all problem boundary conditions. Furthermore constraints could be of linear or non linear nature, as well as they can be of equality or inequality. One obvious case is the positiveness constraint that last for some positive definite parameter like geometrical distance (not coordinates), like radii, mass, principal inertia values, and many others depending on the kind of problem under study. While non-linear case may be difficult to manage, linear equality constraints could be easily included in our formalism just presented for linear models. Indeed, a general constraint can be written as

$$g\left(x\right) \geqslant 0$$

under linearity assumptions reduces to

$$Bx \geqslant h$$

Consequently we can write all as a linear system, after introducing lagrange multipliers to our equations

$$2A^TAx - 2A^Tb + \lambda Bx = 0$$
$$Bx - h = 0$$

in matrix form

$$\begin{pmatrix} A^TA & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ h \end{pmatrix}$$

Could be managed in a similar way.

## Rank deficiency issue

Sometimes when collinearity exist between rows or columns of A, we need to discard some in order to get a non singular problem. That happens often in identification of inertial parameters since they can play same role in determining observed quantities, or just don't play any, leading to null columns.

Is possible to use QR factorization in order to eliminate these and getting a non singular solvable problem over a reduced set of parameters.

### Recursive Least Squares (RLS)

Least squares as seen so far are used collecting observations and performing parameters estimate, online or offline. Nonetheless is easy to prove that observations and equations stacking could be written in iterative form

This is an advantage when dealing with large number of observations and equations, yielding to less demanding calculations, suitable also for real time applications or low resources systems, as well for a continuous process in palace of one time only estimate, resulting in a monitoring process, precious when parameters are expected potentially to change over time.

In case of sensor fusion thi is even more interesting allowing for a continuous update of estimated parameters over time.

### Instrument variable

A variant of OLS method is the IVM (Instrumental Variable Method) that modify slightly normal equations for giving the following.

From a theoretical point of view, the LS assumptions are violated in practical applications. In the equation, the observation matrix depends on the joint position q which is measured and on $\dot{q}$ , $\ddot{q}$ which is computed by pass band filtering q. Therefore the observation matrix is noisy. Moreover the identification process is carried out with a feedback control of the robot. It is known that these violations of assumption imply theoretical biased LS solution. Indeed, from, it comes:

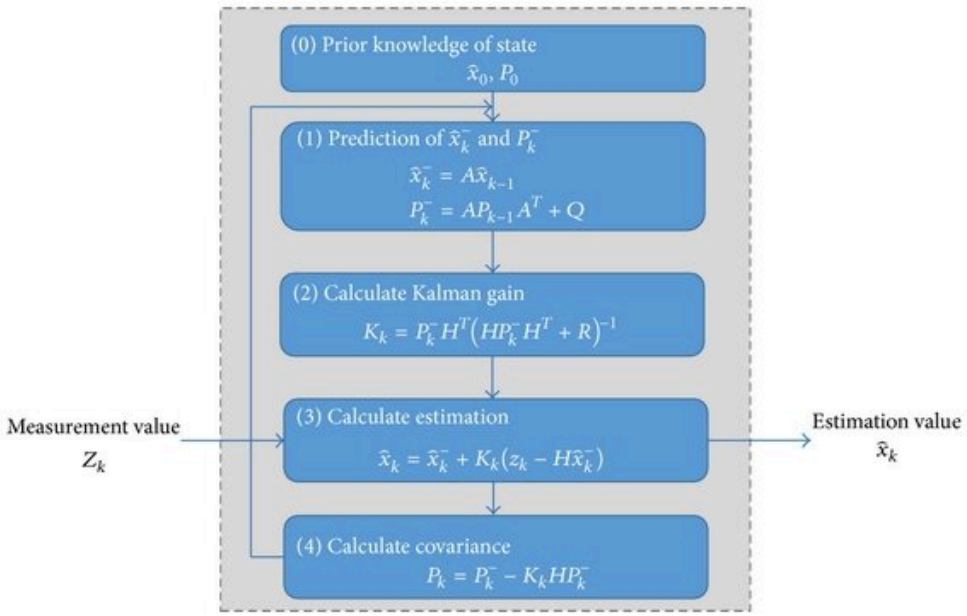
The SRIV (Simple Refined Instrumental Variable) approach deals with this problem of noisy observation matrix and can be statistically optimal (Young, 1979). The Instrumental Variable Method proposes to remove the bias on the solution by building the instrument matrix V such as $E\left[V^T \rho\right] = 0$ and $V^T W$ is invertible. Normal Equation becomes:

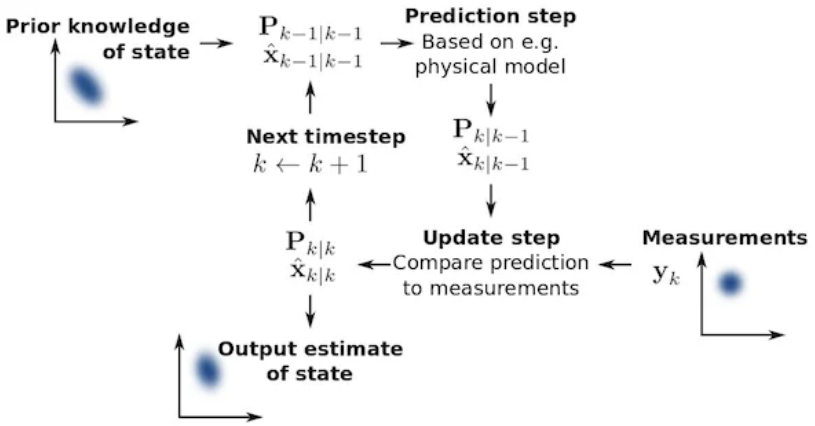
$$x = \left(Z^T J\right)^{-1} Z^T r$$

The first problem is to find an instrument matrix V. A classical solution is to build an observation matrix from *simulated data* instead of measured data. In this way, V is a deterministic matrix and uncorrelated to ρ. So, if we assume that ρ is a zero-mean additive independent noise, then we have $E\left[V^T \rho\right] = 0$ . The simulated states are denoted $q_s$ (simulated joint position), $\dot{q}_s$ (simulated joint velocity) and $\ddot{q}_s$ (simulated joint acceleration). Also in this case arise naturally the usefulness of a digital twin. The remaining of the process, including in case QR factorization, is similar to traditional method.

## Kalman Filtering

Recursive least squares aforementioned lead in a natural way, when minimal covariance condition is imposed, to Kalman filter concept. Equations structure are almost the same, but gains are put in function of signal covariance.






One of main difference respect OLS and KF is last one directly refer to an underlying dynamic system (in original theory a linear continuous state system) working as model.
This is also our case although in most of identification problem dynamics is assumed to be known and reconducted to LS problem, avoiding state propagation issues. Nonetheless one can imagine to include dynamic models equation in a full KF filter scheme using current best estimate of system parameters.
In that case links parameters are expected to be not varying over time (for self evident reasons), so for them system dynamic parameters are

$$\dot{\chi} = 0$$

As well as system dynamics can be easily seen to be non linear in inputs and current state since, given standard state vector

$$\begin{pmatrix} \dot{q} \\ \ddot{q} \\ \dot{\chi} \end{pmatrix}$$

$$\begin{pmatrix} \dot{q} \\ \ddot{q} \\ \dot{\chi} \end{pmatrix} = \begin{pmatrix} \dot{q} \\ M(q)^{-1}\left(\tau - C\left(q, \dot{q}\right)\dot{q} - G\left(q\right)\right) \\ 0 \end{pmatrix}$$

remembering linear relation of dynamics with inertial parameters.

### Connection between Kalman filtering and inertia matrix factorization

Some works have proven a close relationship between Kalman filtering coming from optimal estimation theory and inertia mass factorization as well forward and inverse robot dynamics, using spatial operators notation, as illustrated below (originated from Rodriguez work in 1988).

$$x\left(k\right) = \phi\left(k, k-1\right)x\left(k-1\right) + M\left(k\right)\lambda\left(k\right) + b\left(k\right)$$

$$b = \begin{pmatrix} \omega \times I\omega + m\,p\, \times \left[\omega \times v\right] - T - p \times F \\ m\omega \times p + m\omega \times v - F \end{pmatrix}$$

This analogy, mostly relying in equations shape, is very interesting from physical point of view since could highlight some deep meaning. Following table illustrate better correspondence between two domains entities

IN OPTIMAL ESTIMATION AND RECURSIVE ROBOT DYNAMICS

| Estimation | | Robot Dynamics |
|---|---|---|
| States | $x(k)$ | spatial forces |
| Costates | $\lambda(k)$ | spatial accelerations |
| Measurements | $\tau(k)$ | active joint moments |
| Transition matrix | $\phi(k, k-1)$ | spatial jacobian |
| Process error covariance | $M(k)$ | spatial inertia |
| Known input | $b(k)$ | spatial bias force |
| State-to-output map | $H(k)$ | state-to-joint-axis map |

The key of this analogy is the sequential and iterative nature of forces and torques acting on each link of a serial robot, corresponding to system state and observations correspond to joint actuation. Spatial inertia matrix then play a role of process error covariance where inputs correspond to bais force, coming from each link inertial forces and torques that acts as inputs to the process, modifying system state. Then pivotal step is to cast the system dynamics and kinematics as a two-point boundary-value problem

The use of this remarkable observation lead rather naturally to adopting inversion lemma used in Kalman filtering derivation for the evaluation of inertia matrix inverse carrying out required for forward dynamics problem treatment, usually a not easy task. This lead to following expressions for forward dynamics

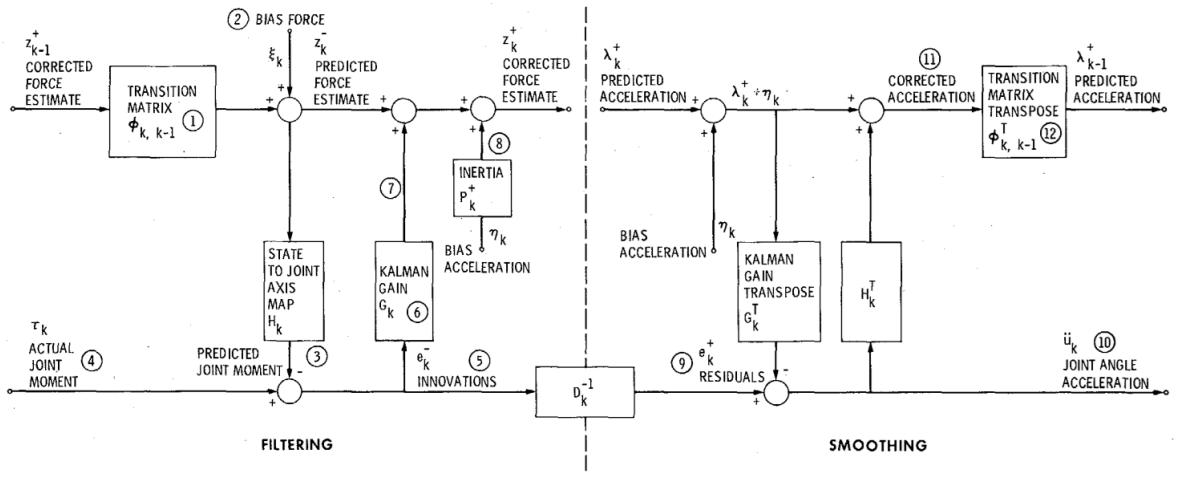
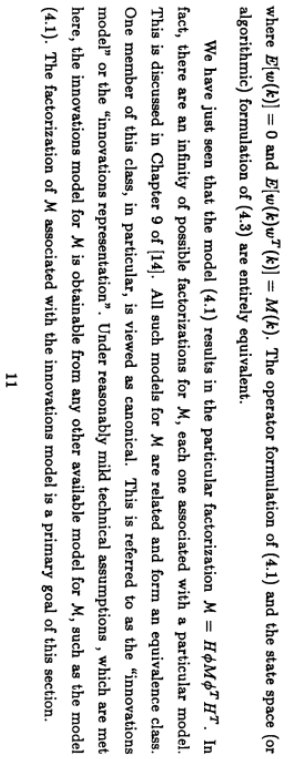



Fig. 3. Filtering and smoothing architecture.

https://ntrs.nasa.gov/19870008018.pdf
ntrs.nasa.gov/api/citations/19880019944.pdf



4. A FORWARD DYNAMICS ALGORITHM
BY MASS MATRIX/OPERATOR FACTORIZATION

Note that the operator $\mathcal{M}$ in the bias-free robot dynamics equations (3.22) is symmetric positive definite. This follows from (3.20). If such an operator can be modeled as the covariance of an output from a known, causal, and finite-dimensional linear system driven by white noise, then the operator can be factored and inverted efficiently by the use of standard techniques from filtering, detection, and estimation theory [8-14]. This factorization and inversion leads to a solution of (3.22). With $\mathcal{M} = H\phi M\phi^T H^T$, such a model is immediately at hand. Indeed, taking

$$Y = H\phi W \tag{4.1}$$

$$E(W) = 0; \quad E(WW^T) = M,$$

where $Y^T = [y_1^T, \ldots, y_N^T]$ and $W^T = [w_1^T, \ldots, w_N^T]$, results in $E(Y) = 0$ and

$$E(YY^T) = H\phi M\phi^T H^T = \mathcal{M}. \tag{4.2}$$

Here, $E(\cdot)$ is the statistical expectation operator [14]. Note that (4.1) is just a succinct way of saying

$$z(0) = 0;$$

for $k = 1, \ldots, N$ loop

$$z(k) = \phi(k, k-1)z(k-1) + w(k); \tag{4.3}$$

$$y(k) = H(k)z(k);$$

end loop;

where $E[w(k)] = 0$ and $E[w(k)w^T(k)] = M(k)$. The operator formulation of (4.1) and the state space (or algorithmic) formulation of (4.3) are entirely equivalent.

We have just seen that the model (4.1) results in the particular factorization $\mathcal{M} = H\phi M\phi^T H^T$. In fact, there are an infinity of possible factorizations for $\mathcal{M}$, each one associated with a particular model. This is discussed in Chapter 9 of [14]. All such models for $\mathcal{M}$ are related and form an equivalence class. One member of this class, in particular, is viewed as canonical. This is referred to as the "innovations model" or the "innovations representation". Under reasonably mild technical assumptions, which are met here, the innovations model for $\mathcal{M}$ is obtainable from any other available model for $\mathcal{M}$, such as the model (4.1). The factorization of $\mathcal{M}$ associated with the innovations model is a primary goal of this section.

11

Thai fact lead to a very interesting analogy between spatial motion dynamics and Kalman filtering.

## Spatial Kalman filtering

### Kalman canonical decomposition

As any linear problem, is possible to apply a linear transformation to the system (variables) leading to a modified system structure (https://en.wikipedia.org/wiki/Kalman_decomposition )

$$A' = TAT^{-1}$$
$$B' = TB$$
$$C' = CT^{-1}$$
$$D' = D$$

Choosing T such that is composed of submatrixes related to a partitioning of states are:

- reachable and unobservable
- reachable and observable

Consequently there are many columns potentially zero.

$$T = [T_{r\bar{o}} \quad T_{ro} \quad T_{\bar{r}\bar{o}} \quad T_{\bar{r}o}]$$

$$\begin{pmatrix} x_a \\ x_b \end{pmatrix} = \begin{pmatrix} A_a & A_{ab} \\ 0 & A_b \end{pmatrix} \begin{pmatrix} x_a \\ x_b \end{pmatrix} + \begin{pmatrix} B_a \\ B_b \end{pmatrix} u(t)$$

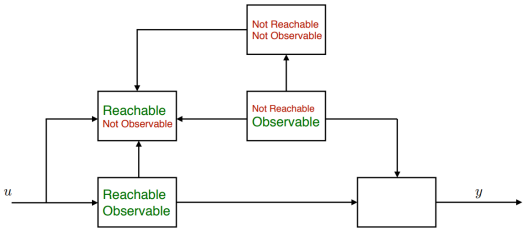
$$\begin{bmatrix} A_a & A_{ab} & A_{ac} & A_{ad} \\ 0 & A_b & 0 & A_{bd} \\ 0 & 0 & A_c & A_{cd} \\ 0 & 0 & 0 & A_d \end{bmatrix} \begin{pmatrix} x_a \\ x_b \\ x_c \\ x_d \end{pmatrix}$$

$$\begin{pmatrix} x_a \\ x_b \\ x_c \\ x_d \end{pmatrix} \begin{pmatrix} B_a \\ B_b \\ 0 \\ 0 \end{pmatrix} (0 \quad C_b \quad 0 \quad C_d)$$

**Kalman Decomposition**

$$\begin{bmatrix} \dot{\bar{x}}_{co} \\ \dot{\bar{x}}_{\bar{c}o} \\ \dot{\bar{x}}_{c\bar{o}} \\ \dot{\bar{x}}_{\bar{c}\bar{o}} \end{bmatrix} = \left[\begin{array}{cc|cc} \bar{A}_{co} & \bar{A}_{12o} & O & O \\ O & \bar{A}_{\bar{c}o} & O & O \\ \hline * & * & \bar{A}_{c\bar{o}} & \bar{A}_{*} \\ O & * & O & \bar{A}_{\bar{c}\bar{o}} \end{array}\right] \begin{bmatrix} \bar{x}_{co} \\ \bar{x}_{\bar{c}o} \\ \bar{x}_{c\bar{o}} \\ \bar{x}_{\bar{c}\bar{o}} \end{bmatrix} + \left[\begin{array}{c} \bar{B}_{co} \\ O \\ \hline \bar{B}_{c\bar{o}} \\ O \end{array}\right] u$$

$$y = \left[\begin{array}{cc|cc} \bar{C}_{co} & \bar{C}_{\bar{c}o} & O & O \end{array}\right] \begin{bmatrix} \bar{x}_{co} \\ \bar{x}_{\bar{c}o} \\ \bar{x}_{c\bar{o}} \\ \bar{x}_{\bar{c}\bar{o}} \end{bmatrix} + Du$$



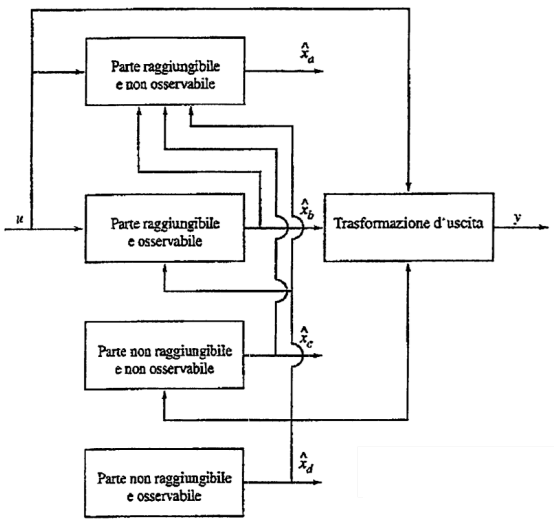




**Figura 3.18**
Scomposizione canonica.

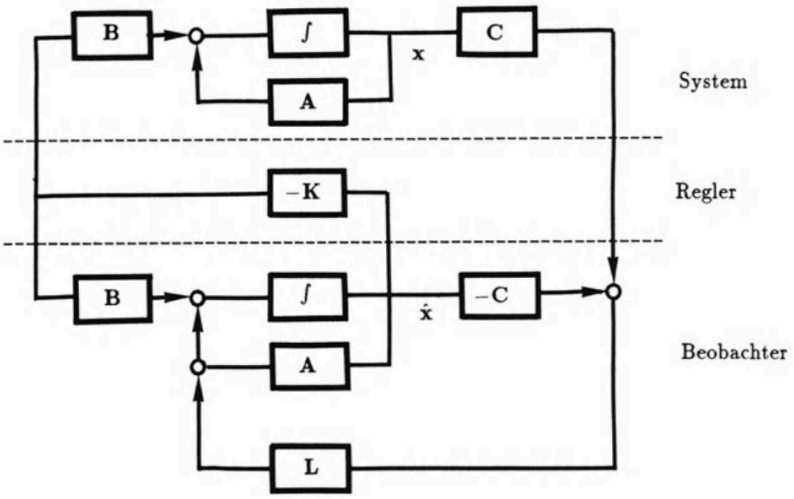
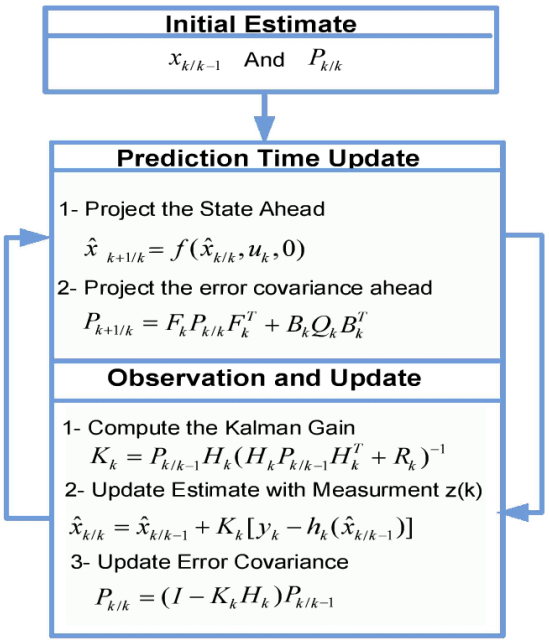



Bild 7.3: Blockschaltbild des vollständigen Beobachters

## Non linear systems extension

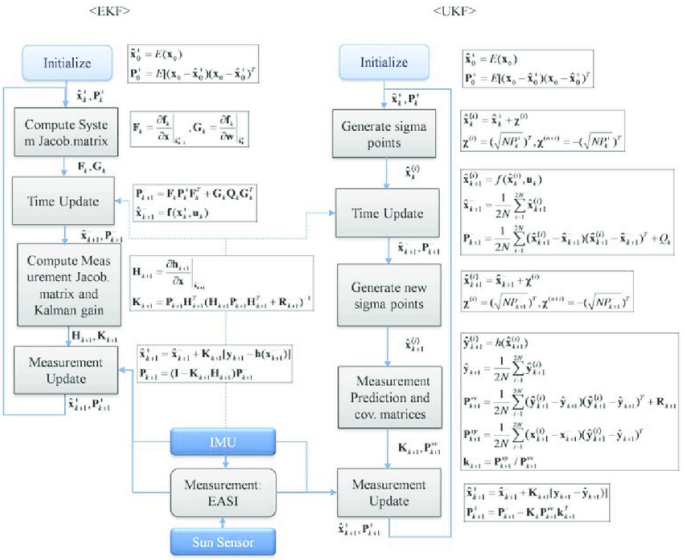
Most of dynamics systems present in nature and in mechanics are only approximately representable as linear dynamic systems. That's especially true for robotics and robots dynamics since almost all robots (excluding pure cartesian robots) present a highly non linear dynamics, with non negligible contributes coming from nonlinear terms present both in inertia matrix and centrifugal and coriolis terms, as well as from non linear friction contributes (for instance coulomb friction is not linear as well as quadratic and exponential terms related to joint speed). Hence the use of linear dynamics model assumed in standard Kalman filtering are not applicable to robotics mechanism dynamics case, as for many other applicative case. Other possible concerning is about the presence of a non linear observation equation. On the other side these problems, commonly coming out in mechanical systems (for instance even single rigid body dynamics in body frame is non linear in angular speeds) as in many other fields, have already been managed by the use of some approximations or extensions of like well known in literature EKF and UEKF. Let's give a brief presentation of the method and consider some consequence. The first matter is the use of dynamic system to predict system state provided current state estimate and known (assumed to be) parameters. Then we can imagine to replace linear dynamic model with non linear one, in case is known. That's easy, but we need to remember dynamic matrix is also involved in covariance propagation. Common approach in this case is the use of a linearization of the system around current estimated state point. One could also imagine to use linearization also for state propagation and results should be pretty close for small enough propagation steps. Resulting modified algorithm looks as follow, quite similar to original version:




Key trouble in using EKF is tied to the lack of proof of convergence and optimality of linear case (would to be honest also require gaussian white noise, often lacking in real applicative case). When non linear measurement models are involved we can similarly use directly non linear model for observation and jacobian of for Kalman gain calculation.
Nevermore EKF has been employed extensively into application for academic and industrial purpose with effective results in a large variety of problems from navigation to many others. What presented is a first order extension, but also higher order extension have been proposed in literature. Is of most importance indeed to keep non linearities as small as possible to keep filter performances acceptable.

Alternatively a common and performing (better) choice is the use of UEKF. UKF uses deterministic sampling named Unscented Transformations (UT) to pick a minimal set of sample points (called sigma points) around the mean.
The sigma points are then propagated through the nonlinear functions, from which a new mean and covariance estimate are then formed. The resulting filter depends on how the transformed statistics of the UT are calculated and which set of sigma points are used. It should be remarked that it is always possible to construct new UKFs in a consistent way
This technique removes the requirement to explicitly calculate Jacobians, which for complex functions can be a difficult task in itself (i.e., requiring complicated derivatives if done analytically or being computationally costly if done numerically), if not impossible (if those functions are not differentiable).

A comparison between two algorithms is proposed in following picture




So different extension of original Kalman filter has been developed over years in order to manage a variety of cases originally out of scope, like non gaussian noise and non linear systems. Principal part of them are reported in following tables (from [Urrea-Kalman Filtering]

TABLE 2: Comparison between KF and its modifications.

| State estimator | Model | Assumed distribution | Computational effort |
|---|---|---|---|
| Kalman filter | Lineal | Gaussian | Low |
| Extended Kalman filter | Locally linear | Gaussian | Low or medium |
| Unscented Kalman filter | Nonlinear | Gaussian | Medium |
| Particle filter | Nonlinear | Non-Gaussian | High |

TABLE 1: Modifications to the KF.

| KF<br>EKF | UKF | Others |
|---|---|---|
| IEKF | AUKF | KKF |
| AEKF | DUKF | CDKF |
| AREKF | SRUKF | DKF |
| Q-EKF | | EKF-SLAM |
| OC-EKF | | UKF-SLAM |
| SO-EKF | | CDKF-SLAM |
| RI-EKF | | CV-SLAM |
| | | K-mean clustering KF |

In same piece of work a list of key use in robotics of KF and its extensions is given

Table 3: Main filters used in robotics.

| Filter | Wheels and legs | Aquatic | Aerial | Manipulators |
|---|---|---|---|---|
| EKF | Localization<br>Positioning<br>Trajectory<br>Detection | Localization<br>Trajectory | Control<br>Navigation and guidance<br>Parameter identification | Control<br>Parameter estimation |
| UKF | Localization<br>Positioning<br>Trajectory<br>Detection | Position estimation<br>Positioning | Control<br>Parameter identification | Parameter estimation<br>Control |
| AEKF | Localization<br>Estimation | Navigation<br>Localization | Navigation and guidance | Control<br>Trajectory tracking<br>Position estimation |
| AUKF | Trajectory tracking | Position and direction estimation<br>Navigation | Control | Attitude estimation |
| IEKF | State estimation (position and speed) | | | State estimation (position and speed) |

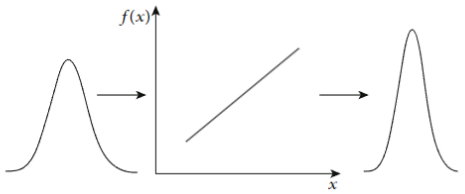



Figure 6: Representation of the preservation of the Gaussian features after linear transformation.

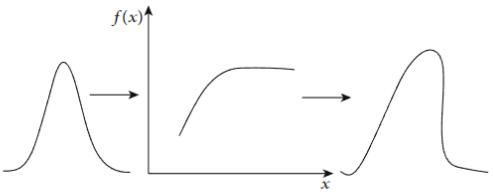
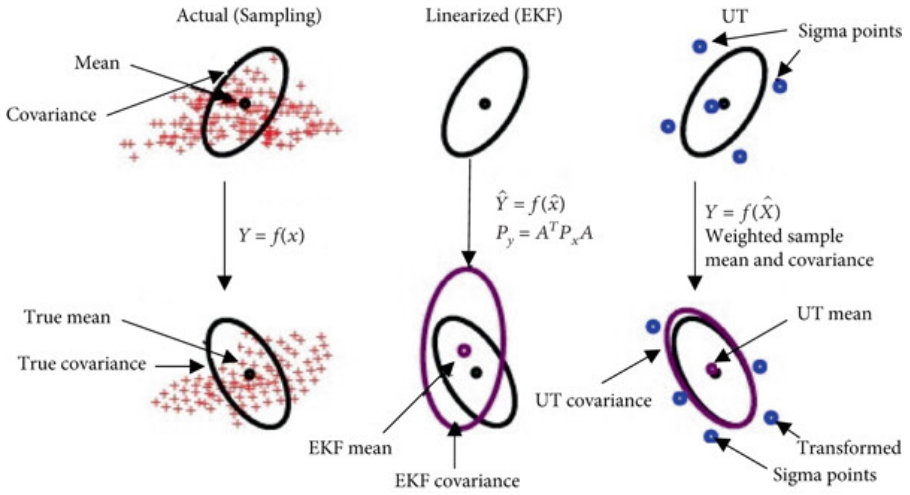



Figure 7: Representation of lost Gaussian features when function $f(x)$ is nonlinear.




## Use of symbolic methods in EKF

As seen EKF would need even more than KF of mathematical operations like differentiation and taking jacobian out of model. Is possible to perform this directly with numerical method,

but the use of symbolic approach could be more effective and accurate, at least when model is formed of differentiable equations.

### Applications of Kalman filtering and extended Kalman filtering in robotics

In robotics field EKF and KF has been applied for a variety of applications, from obvious signal smoothing, time series analysis and signal processing, to sensor fusion,  system parameters identification and estimate, state identification, parameters and state identification, positioning  down to monitoring including adaptive control use, to solution of inverse kinematics, motion planning and control, trajectoryoptimization.

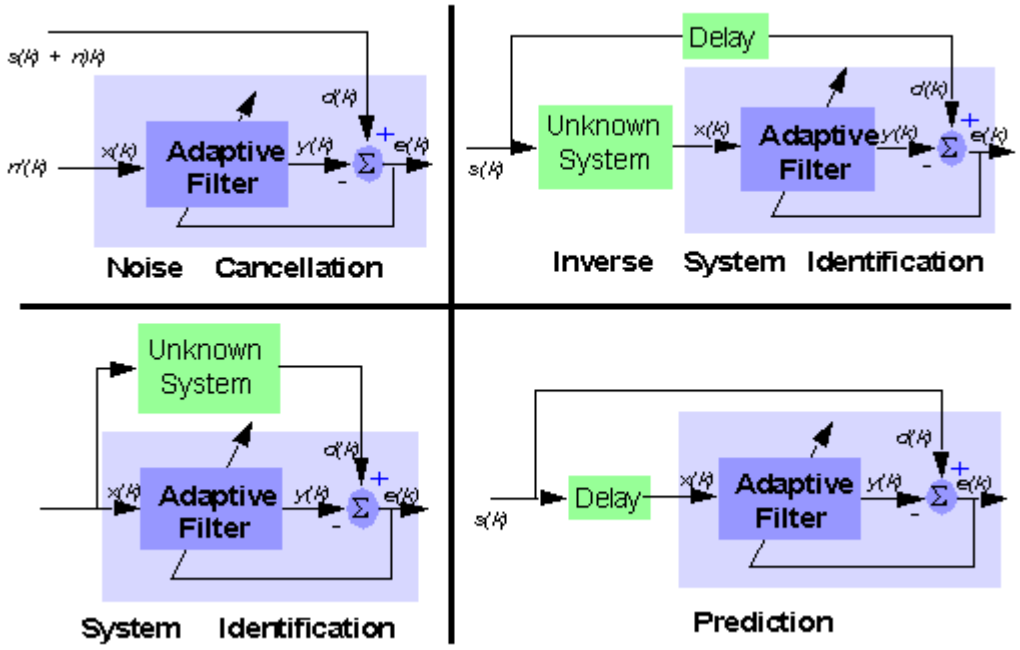
### Comparison of EKF vs WLS for identification

: EKF algorithm estimates both the velocities and the parameters while WLS estimation needs the joint velocity and acceleration to be calculated separately by pass band filtering. However, it does not appear to be an advantage for EKF. Estimations of the parameters are very close for both methods, but EKF algorithm is very sensitive to the initial conditions and the convergence speed is slower. Moreover, recursive calculations are time consuming, and symbolic calculation of the Jacobian matrix is very tedious for the EKF method. The conclusion is that the WLS method with the inverse dynamic model appears to be better than EKF for off line identification. Future work will concern the analysis of the on line behavior with a priori knowledge given by WLS and parameter tracking with EKF algorithm.

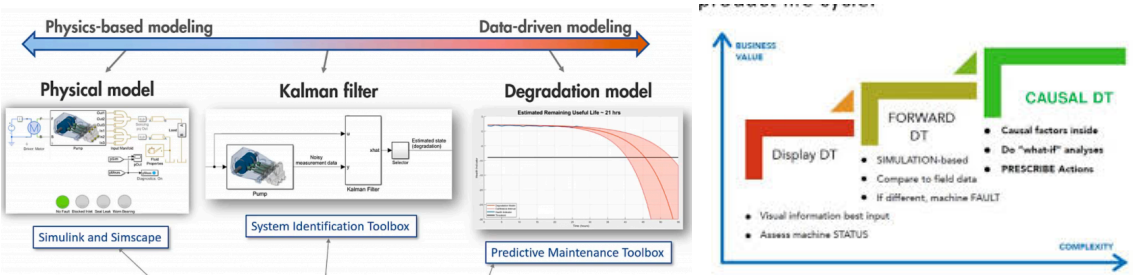
Source:https://www.diag.uniroma1.it/~batti/papers_ifsd/1.pdf

### Beyond Kalman Filtering

LS, RLS, WLS are just some of examples of a larger set of mathematical methods coming from estimation theory and filtering theory. For a deepening on the subject see chapter [Filtering and identification]. Reasons why we wanted to treat this subject on a digital twin work is motivated by its essentiality for DT functioning itself. Filters are required for systems identification, and a correct system identification is required for a proper working of a DT. When DT model and real system start diverging one other is a key moment when a DT take a strategic importance role. Secondly DT filtering is able to lead to observation of not directly measurable systems parameters and states, and alert predictively undesired, anomalous, or alert conditions.




Use of filtering is one of traditional and mostly well consolidate approach, locating at intermediate level between data driven methods and pure physical modeling. In addition almost no DT can work on physical measurement data without a preprocessing usually consisting of a smoothing, filtering in frequency or any other, including KF filter on data in order to cancel noise, or reconstruct signal desired content. Relevant process can be located inside data processing block of digital twin architecture.




## Optimization problems: a connection with dynamics

We've seen as many kind of mathematical problems could be reduced to optimization problems. There's a further application pertaining dynamics. A direct problem of dynamics, as well as inverse dynamics can be reformulated in the sense of functional optimization. In dynamics consequent principle is named "variational principle" and associated functional are Lagrangian function (already seen on relative chapter) and Hamiltonian function.
Variational principle states effective dynamics problem solutions are those satisfying

$$\delta \int_{t_0}^{t_1} L\left(q, \dot{q}, \ddot{q}, t\right) dt = 0$$

over functions space of trajectories $q\left(t\right)$.

Another form is by using relativistic condition of geodetic motion in a non flat differential manifold used to describe acting accelerations and forces on a particle (usually used only for gravitational forces, because of mass equivalence principle allowing for its cancellation into equations set). Using tensorial notation:

$$\delta s = \delta \int \sqrt{g_{\mu\nu} dx^{\mu} dx^{\nu}} = 0 = \delta \int \sqrt{g_{\mu\nu} \dot{x}^{\mu} \dot{x}^{\nu}} d\tau = 0$$

It can be proved that after some linearization approximation on metric tensor for low speed motions this minimal distance principle lead to same variational principle of dynamics in classic mechanics. We should then consider geodetic equations in differential form

$$\frac{\partial x^{\mu}}{\partial s} + \Gamma^{\mu}_{kh} \frac{\partial x^{k}}{\partial s} \frac{\partial x^{h}}{\partial s} = 0$$

are then a relativistic generalization (under proper assumptions) of classic mechanics equations. Is possible demonstrate mechanics equations have effectively that shape. In spite of most of mechanics be tensors based, this expression is not since $\Gamma$ is a pseudo-tensor.

### Optimal control

Another traditional problem related to optimization is the one of optima control, can be solved using Pontryagin optimality criterion, equivalent to Bellmann principle.
It states, by means of a settling similar to lagrange multiplier method, that optimal control problems can be solved applying system dynamics equations as a constraint and introducing a functional cost to be minimized. Introducing then a functional J

$$J = \Psi\left(T\right) + \Phi\left(0\right) + \int_{t_0}^{t_1} f\left(u, g, t\right) dt + \lambda^{T} g\left(x, t\right)$$

where constraint equations is a dynamic (generally non linear or even linear) system, whose dynamics depend on input control function u.

$$\dot{x} = g\left(x, u, t\right)$$

Observe as in the equation other terms can be introduced in order penalize initial and fnal condition at trajectory end and/or beginning.

Minimization of the function J over all possible controls $u(t)$ satisfying required conditions and dynamics $x(t)$ respecting constraint function $g(x,t) = \dot{x}$, for a set of multipliers $\lambda(t)$ depending on time, lead to a solution of both dynamics constraint and minimization of cost function. Solutions turns out to be a set of Hamilton like equation, named in fact Hamilton-Jacobi-Bellman (BHJ) equations,

$$V(x,t) = \min\left(\int_0^T C(x(t),u(t),t)dt + D(x(T))\right)$$

Solved by solutions of a Partial Differential Equation (PDE):

$$\frac{\partial V(x,t)}{\partial t} + \min_u \left\{ \frac{\partial V(x,yt)}{\partial x} \cdot F(x,u) + C(x,u) \right\} = 0$$

under terminal conditions

$$V(x,T) = \Psi(T)$$

These equations are closely similar and connected to Hamilton Jacobi equations found in mechanics.  Indeed one Alternatively one can consider Halmiton equations system, where associated

$$\begin{cases} \dot{x} = -\frac{\partial J}{\partial \lambda} \\ \dot{\lambda} = \frac{\partial J}{\partial x} \end{cases}$$

This approach is known also as optimal control problem

Some simple applications are

- Optimal time controls
- Minimal energy control
- LQR regulators
- LQG regulators

And many others. Optima control is very interesting for automation applications since allows to find out controls having best possible performance. On the contrary we should remark some assumptions are made on this treatment regarding control functions, that are unbounded in magnitude and derivative, as well as resulting system trajectory in phase space. Real control design should then consider additionally also physical realizability and compatibility with system constraints in terms of maximum applicable control loads, speeds, accelerations, material strengths, actuators limit performances, bandwidth and so on.

### Riccati equations

Solution of optimal control for linear dynamic system under infinite horizon conditions of type

$$J = \int_0^\infty x^T Q x dt + u^T R u$$

lead to Algebraic Riccati equations, whose solution is not hard to find, algebraically

$$0 = -\left(PA + A^T P - PBR^{-1}B^T P + Q\right)$$

$$u = -R^{-1}B^T Px$$

$$u = -R^{-1}B^T P\left(x - x_e\right)$$

### Dynamic programming

Dynamic programming exploits the *Bellman principle of optimality*, which states that in an optimal sequence of decisions, each subsequence must also be optimal. Dscretzing then a problem into a finite number of subproblems is then dynamic programming method, whose optimal solution is found by solving for optimal solution/s of a sequence of smaller and usually simpler subproblems.

### From optimization to reinforced learning policy: a connection bridge

Reinforced learning models use a policy to define reward a system get for a behavior under determined conditions. Best candidate for a learning reinforcement are behaviors that improves reward.

Robustness and flexibility of reinforced learning allows applications to different field, from learning to control, even where traditional systems fails or hardly found a solution that works.

All this have a price of a training time usually taking many hours of attempts and evaluations that in a general sense are not a symptomatic of efficiency, but are quite effect do require little intervention from user other than choosing a suitable policy to apply to the system in order to make it works. For instance we can make a robot walk without imposing directly any command law, but letting the system define by itself most suitable actions to take in order to achieve objective superimposed.