Massachusetts Institute of Technology
16.412J/6.834J - Cognitive Robotics
Spring 2016

Iterative Risk Allocation

A Gentle Introduction by Ben Ayton and Steve Levine

Introduction

Iterative Risk Allocation (IRA) is a method of solving Robust Model Predictive Control (RMPC) problems with a joint chance constraint. In this context, this means a Model Predictive Control problem with a specified maximum risk of constraint violation.

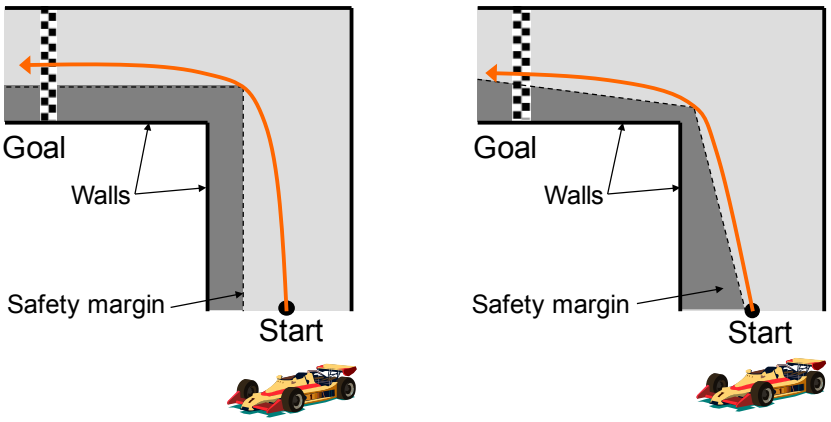


Figure 1: This racecar example can be framed as an RMPC.

Figure 1 shows an example of an RMPC with a racecar. Our goal is to find a set of control inputs (steering commands for the racecar) that will achieve our goal (crossing the finish line) with the greatest possible utility (fastest possible time). However, there is uncertainty in this situation; our vehicle has noisy dynamics. If we steer our vehicle in a certain direction, it may end up actually going in a slightly different direction due to tire wear, road conditions, etc. We must avoid driving off the road, in which case we crash and lose the race. However, we wish to be *robust* to our uncertainty and achieve our goals anyway (hence the word *robust* in RMPC). We quantify what we mean by "robust" with a *chance constraint*, which intuitively describes a maximum acceptable probability of failure (in this case, driving off the road and crashing). Suppose we wish to get to the finish line as quickly as possible, but are willing to accept a 0.1% chance that our racecar may crash; this is a chance constraint. Note that there is often a direct tradeoff between risk and reward. In the left image, the racecar takes a longer path because it uses its risk inefficiently (and hence has lower utilty). At right, the rarecar daringly takes a shorter path close to the corner - thus using most of its risk at once but also having higher utility.

IRA solves an RMPC this by *allocating* the total risk (denoted Δ) between all individual

constraints (each is allowed to take risk up to δ_i) and allowing a solution to be found under that risk allocation. Once a solution is found, it detects where risk is being used effectively, and reallocates it to those constraints from where it is not being used. The problem is *iteratively* solved this way until the risk allocations converge.

In the racecar Figure 1, the risk allocation is visualized in the safety margin. Higher risk allocations are associated with "cutting it closer" to the edge of the track, since a slight slip up there is more risky than when in the middle of the road. Imagine discretizing the racecar's trajectory to five or so timepoints. You could associate each timepoint i with a maximum risk δ_i that is allowed to be taken at that timestep. At left in the diagram, the risk may be distributed equally amongst all the δ_i. However, by allocating risk differently (at right), the racecar could reach the goal faster, but still take the same total risk Δ. The racecar would like to take the most risk where it would be the most beneficial to its utility; in this case, that occurs near the corner.

Note that IRA itself does not specify how a RMPC problem is solved once its risk has been specified. In principle, IRA can be used with any optimal solver.

Basic Idea: Allocating Your Risk to Different Constraints

We'll get to the full specification of a RMPC problem in a bit, but for now, we will simply say that it requires assignment to a finite set of **deterministic** control variables $U = \{\boldsymbol{u}_0, ..., \boldsymbol{u}_{T-1}\}$ that optimizes some objective function and satisfies a set of constraints.

The control variables determine the distributions of a finite set of **potentially random** state variables $X = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_T\}$. **Note that all control and state variables are vector quantities.** Each of the constraints may be deterministic, acting on either the deterministic variables or a deterministic function of random variables (such as their mean), or the constraints can be probabilistic, which we will refer to as **chance constraints**. We restrict our attention to RMPC problems where the chance constraints are limited to a single **joint chance constraint**. (Note that IRA can be applied to multiple joint chance constraints with few extensions, but we will not cover the details here.)

Let us denote the set of N individual constraints, which together compose a joint chance constraint, as $\{C_1, ..., C_N\}$. For abbreviation, we write the probability that C_i is true as $P(C_i)$ and the probability that it is false as $P(\overline{C_i})$. Specification of a joint chance constraint means specifying a single **risk** Δ that indicates the maximum probability that any constraint C_i is violated,

$$P\left(\bigvee_{i=1}^{N} \overline{C_i}\right) < \Delta$$

(the notation $\bigvee$ means the "or" or disjunction of a bunch of things, and $\bigwedge$ means the "and" or conjunction of a bunch of things).

Put another way, if the probability of at least one of the constraints being violated is $< \Delta$, then the probability that all of the constraints are satisfied is $\geq 1 - \Delta$:

$$P\left(\bigvee_{i=1}^{N} \overline{C_i}\right) < \Delta \iff P\left(\bigwedge_{i=1}^{N} C_i\right) \geq 1 - \Delta.$$

The approach is to appeal to a basic principle of joint distributions,

$$P\left(\overline{C_1} \vee \overline{C_2}\right) \leq P(\overline{C_1}) + P(\overline{C_2}).$$

It follows that if we have N constraints, we split the total risk into N **risk allocations** $\{\delta_1, ..., \delta_N\}$, and specify N **individual chance constraints** which satisfy

$$P(\overline{C_i}) < \delta_i \iff P(C_i) \geq 1 - \delta_i$$

$$\sum_{i=1}^{N} \delta_i \leq \Delta,$$

then we have satisfied the joint chance constraint, because

$$P\left(\bigvee_{i=1}^{N} \overline{C_i}\right) \leq \sum_{i=1}^{N} P(\overline{C_i}) < \sum_{i=1}^{N} \delta_i \leq \Delta.$$

We have now formulated a RMPC problem from one with a single, joint chance constraint over all of the constraints, to another with only individual chance constraints. The solution to this new problem satisfies the joint chance constraint RMPC, and is, in most cases, easier to solve. Note that this is method of allocating risk may be conservative, even if $\sum_{i=1}^{N} \delta_i = \Delta$ is satisfied exactly. But normally it will be advantageous to use all the risk we are allowed.

Basic Idea - Iterative Risk Allocation

For now, let's postpone the discussion of how we solve the RMPC with individual chance constraints above, and assume we have some method of doing so. The solution will specify the values of $P(C_i)$ (perhaps indirectly, but they will be computable).

Intuitively, if $P(\bar{C}_i)$ is much less than δ_i, we have allocated more risk to the constraint C_i than we may need, because there is some available that the solution has not used. This constraint is "inactive." We could take some risk away and define a $\delta_i' < \delta_i$ such that $P(\bar{C}_i) \leq \delta_i'$ is still satisfied and it would not affect the solution.

Similarly, if $P(\bar{C}_j)$ is very close to δ_j, almost all of the allocated risk is being used for that constraint. It's an "active" constraint. If we allocate more risk to that constraint with a larger $\delta_j' > \delta_j$, there may be a better solution that makes use of that risk. Intuitively, this is because our constraint that $P(\bar{C}_j) < \delta_j$ may be "limiting" our solution, in that it's what could be

preventing us from achieving a higher utilty. This is a key idea of IRA! By taking a little more risk at this constraint, maybe we could improve our solution utility (and cross the finish line sooner, for example). Even if not, we certainly will not make the solution any *worse* by allocating more risk to such a constraint.

Thus, the key idea of IRA is to re-allocate risk from constraints that don't need it (are inactive), and give it to those constraints that may be able to use it (are active), in an attempt to improve our utility. We keep iteratively doing this (hence the name, IRA).

We define some tolerance η, and then distinguish between active and inactive constraints:

$$\text{Active: } |P(C_i) - 1 + \delta_i| \leq \eta$$

$$\text{Inactive: } |P(C_i) - 1 + \delta_i| > \eta.$$

At each iteration, IRA takes some risk from all the inactive constraints. Then, it divides that collected risk uniformly between the active constraints, and adds it to those active constraints. That's it! The innovation in IRA comes from how we can intelligently decide how much risk to take from the inactive constraints. To do that, we need to look in more detail at the form of constraint we expect.

Working with Linear Chance Constraints

We consider all our chance constraints to be expressible as thresholds on linear combinations of our state variables. This means that we define some vector $\boldsymbol{h}_i$ and some scalar g_i such that

$$C_i = \text{True} \iff \boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i$$

$$\Rightarrow P(C_i) \geq 1 - \delta_i \iff P\left(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i\right) \geq 1 - \delta_i.$$

We focus on this form of constraint because it is simple, but simultaneously powerful and expressive. $\boldsymbol{x}_i$ can be composed of any subset of elements of X arranged in a single vector, so any linear combination of state variables can be considered. However, in most cases, $\boldsymbol{x}_i$ will be one element of X, meaning the constraint applies to the state at a single timestep. Many types of constraints can be encoded as linear constraints, and the representation can be naturally encoded in a linear program, which can be used to solve the RMPC problem. Multiple limits on $\boldsymbol{h}_i^T \boldsymbol{x}_i$ can be encoded through multiple constraints, each with their own individual risk.

Note that the product $\boldsymbol{h}_i^T \boldsymbol{x}_i$ is a **scalar** random variable, which has its own cumulative distribution function. In some cases, it turns out to be more convenient to work with $\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i$, where $\bar{\boldsymbol{x}}_i$ is the mean of $\boldsymbol{x}_i$. Note that $\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i$ is also a scalar random variable, with **zero mean** and the **same variance** as $\boldsymbol{h}_i^T \boldsymbol{x}_i$.

We denote the cumulative distribution function of $\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i$ by $\text{cdf}_i(\cdot)$. By definition:

$$P\left(\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i \leq a\right) = \text{cdf}_i(a),$$

from which we can immediately write

$$P\left(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i\right) = \text{cdf}_i\left(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i\right).$$

The above is an expression for the **probability a constraint is satisfied**, which must be compared to the allocated risk to determine if a constraint is active or inactive. The above implies

$$P(C_i) \geq 1 - \delta_i \iff \delta_i \geq 1 - \text{cdf}_i\left(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i\right).$$

The significance of this result is that we have derived a **lower bound** on the acceptable values of the risk allocation, $\delta_{i,min}$, through a deterministic and known function of deterministic variables. For any inactive constraint, we may reduce δ_i to no less than this value. In other words, we reformulate the test for whether a constraint is active or inactive as:

$$\delta_{i,min} = 1 - \text{cdf}_i\left(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i\right)$$

$$\text{Active: } |\delta_i - \delta_{i,min}| \leq \eta$$

$$\text{Inactive: } |\delta_i - \delta_{i,min}| > \eta.$$

It is important to understand that the value of $\bar{\boldsymbol{x}}_i$ is solved for in the RMPC problem. Therefore, the lower bound is **not a global bound for the problem**, but a lower bound for the acceptable risk allocation for a given solution. So long as $\delta_i \geq \delta_{i,min} \, \forall \, i$, the current solution is valid, but if it is violated for any i, a new but worse solution may still exist.

RMPC Formulation with Gaussian State Variables

We consider a discrete-time linear time-invariant problem posed over $T+1$ timesteps $\{0, ..., T\}$, where the system is subject to a conjunction of N chance constraints. The state at timestep $k+1$ is a linear combination of the state and control variables at timestep k, plus a zero-mean Gaussian noise with known covariance. The control variables at each timestep are limited between a minimum and maximum. The initial state is specified, also as a Gaussian random variable, and we attempt to minimize the expectation of some function of the state and control variables. In total, we have

$$
\begin{aligned}
\min \quad & E[J(X,U)] \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w}_k \\
& \bar{\boldsymbol{x}}_{min} \leq \bar{\boldsymbol{x}}_k \leq \bar{\boldsymbol{x}}_{max} \\
& \boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max} \\
& \boldsymbol{w}_k \sim \mathcal{N}(0, \Sigma_w) \\
& \boldsymbol{x}_0 \sim \mathcal{N}(\bar{\boldsymbol{x}}_0, \Sigma_{x,0}) \\
& P\left(\bigwedge_{i=1}^{N} \boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i\right) \geq 1 - \Delta \\
& \text{plus other deterministic constraints.}
\end{aligned}
$$

Note that the above is a stochastic optimization - it involved random variables and their distributions. What we will show shortly is that, given a risk allocation for each of the individual chance constraints, we can convert this stochastic optimization into a deterministic optimization - something like a linear program, or a MILP, that will allow us to solve for an optimal solution easily under that risk allocation.

The approach here is to build a linear program that considers only the **means of the state variables** and the control variables, which can be solved for under a given risk allocation. In this way, the linear program only considers deterministic variables. We define $\bar{X} = \{\bar{\boldsymbol{x}}_0, ..., \bar{\boldsymbol{x}}_T\}$, and assume we can formulate the objective function using $\bar{X}$ instead of X,

$$E[J(X,U)] = J(\bar{X}, U).$$

We also note the following two identities, from the properties of Gaussian random variables:

$$\bar{\boldsymbol{x}}_{k+1} = A\bar{\boldsymbol{x}}_k + B\boldsymbol{u}_k$$

$$\Sigma_{x,k+1} = A\Sigma_{x,k}A^T + \Sigma_w.$$

The above result is very helpful! It says that the covariance $\Sigma_{x,k}$ does not depend at all on the control values $\boldsymbol{u}_k$ that we'll ultimately be optimizing over. In fact, we can compute all $\Sigma_{x,k}$ at the start because we know $\Sigma_{x,0}$, Σ_w, and A. Furthermore, although we can't guarantee what exactly will happen to $\boldsymbol{x}_k$ because of the random noise, we can note that its mean $\bar{\boldsymbol{x}_k}$ is directly influenced by the control inputs $\boldsymbol{u}_k$.

We use the previously detailed expansion to decompose the joint chance constraint involving Δ into individual chance constraints of the form $\Pr(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i) \geq 1 - \delta_i$. We also express the distribution of $\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i$ as

$$\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i \sim \mathcal{N}(0, \boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i),$$

which means we can express cdf_i as the cumulative distribution function of a Gaussian with zero mean and variance $\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i$. But how can we represent the cumulative distribution function of a Gaussian? It turns out we can do so with the erf function. For a Gaussian with mean μ and variance σ^2, its cumulative distribution function is

$$\text{cdf}(a) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{a-\mu}{\sigma\sqrt{2}}\right)$$

(note that the erf function can be readily evaluated, just like other functions like cos).

So, since $\boldsymbol{h}_i^T \boldsymbol{x}_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i \sim \mathcal{N}(0, \boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i)$, we can write its cdf as

$$\text{cdf}_i(a) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{a}{\sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i}}\right).$$

Earlier, we saw that the probability $\Pr(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i) = \text{cdf}_i\left(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i\right)$. If we plug this result in the above cdf_i for a Gaussian by substituting a, we can compute **the probability that a linear constraint is satisfied** when the state is Gaussian:

$$\Pr(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i}{\sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i}}\right)$$

Previously, we considered the statement $\delta_i \geq 1 - \text{cdf}_i\left(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i\right)$ as a lower bound on δ_i to make a solution with $\bar{\boldsymbol{x}}_i$ satisfy $P(C_k^i) \geq 1 - \delta_i$. But we can also view this as a constraint that must be placed on $\bar{\boldsymbol{x}}_i$ so that $P(C_k^i) \geq 1 - \delta_i$ is satisfied for a fixed value of δ_i. This is valid, because risk can be treated as fixed for a given RMPC problem once it has been allocated within IRA. We can therefore define a set of linear constraints on $\bar{\boldsymbol{x}}_i$ such that all risk bounds are satisfied:

$$\delta_i \geq \frac{1}{2} - \frac{1}{2}\text{erf}\left(\frac{g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i}{\sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i}}\right)$$

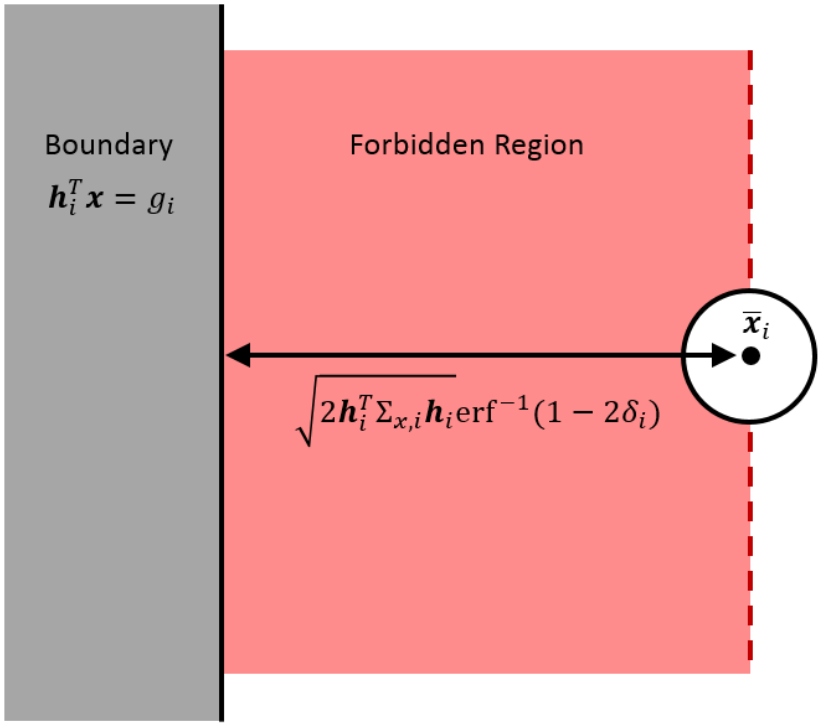
$$\Rightarrow \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i \leq g_i - \sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i}\text{erf}^{-1}(1 - 2\delta_i).$$

This is a key result! We have "determinized" our chance constraint. We have taken an individual chance constraint, specified in the form $\Pr(\boldsymbol{h}_i^T \boldsymbol{x}_i \leq g_i) \geq 1 - \delta_i$ and involving the random variable $\boldsymbol{x}_i$, and converted it to a regular, linear constraint only involving its mean $\bar{\boldsymbol{x}}_i$ (remember that δ_i is constant, so that long, scary-looking expression is also just a constant). We can determinize all our individual chance constraints in this way, under a constant risk allocation, and transform our problem into a linear program involving $\bar{\boldsymbol{x}}_i$.

The equation above has a nice intuitive interpretation. $\sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i}\text{erf}^{-1}(1 - 2\delta_i)$ is a positive function that pushes $\boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i$ further from g_i, so that $\boldsymbol{h}_i^T \boldsymbol{x}_i$ is less likely to cross the boundary.

In this way, $\sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i} \text{erf}^{-1}(1-2\delta_i)$ acts as a "forbidden region" that grows with larger $\Sigma_{x,i}$ and with smaller δ_i. So as the uncertainty in $\boldsymbol{x}_i$ increases, and the risk allocated to a constraint decreases, the mean must be further from the boundary to ensure that that the constraint is satisfied.

The above explains the behavior of the racecar example at the beginning of this document. Note that at the corner, the safety region / forbidden region is smaller - it is tighter around that area. That's because IRA is allocating more of it's total risk there, causing the boundary to be pushed away just a little bit.



To summarize, the constraint RMPC problem can then be framed as the following linear program for a given risk allocation. **This is the "determinized" version of the problem; it is no longer stochastic, as all of the probabilistic / chance constraints have been converted to linear constraints (as all δ_i are constant).** All the constraints below are linear, because all δ_i are specified once inside IRA, and all $\Sigma_{x,k}$ are computable before the problem is solved using known constants A and Σ_w:

$$
\begin{aligned}
\min \quad & J(\bar{X}, U) \\
\text{s.t.} \quad & \bar{\boldsymbol{x}}_{k+1} = A\bar{\boldsymbol{x}}_k + B\boldsymbol{u}_k \\
& \bar{\boldsymbol{x}}_{min} \leq \bar{\boldsymbol{x}}_k \leq \bar{\boldsymbol{x}}_{max} \\
& \boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max} \\
& \Sigma_{x,k+1} = A\Sigma_{x,k}A^T + \Sigma_w \\
& \bigwedge_{i=1}^{N} \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i \leq g_i - \sqrt{2\boldsymbol{h}_i^T \Sigma_{x,i} \boldsymbol{h}_i} \text{erf}^{-1}(1 - 2\delta_i) \\
& \text{plus other deterministic constraints.}
\end{aligned}
$$

We're now prepared to walk through the IRA algorithm in full! Let's take a look.

The IRA Algorithm

The IRA algorithm is presented below. But first, we present a simpler version in English:

1. Initialize our risk allocation uniformly
2. Determinize the problem and solve using a sub-solver
3. Find which individual chance constraints are active
4. Reallocate risk:
   (a) Remove some risk from any inactive chance constraint δ's
   (b) Add unused risk to active chance constraints δ's
5. Go back to 2

The more detailed IRA algorithm is presented below. The algorithm also makes use of a convergence tolerance ϵ on the objective function, and a free parameter $0 < \alpha < 1$. $\bar{J}^*$ refers to the utility value for the optimized problem.

Line 1 allocates the total risk Δ uniformly between all constraints. We then enter the do-while loop, which continues until the objective value $\bar{J}^*$ converges. Line 5 takes the RMPC problem and our risk allocations δ_i and constructs the determinized version of the problem that is a normal (and not probabilistic) optimization (the δ_i's are constants in this version). Line 6 solves this determinized version, and retrieves the objective value. Line 7 partitions each of the chance constraints into two sets: an active set of constraints (those that are active), and an inactive set. We note the number of active constraints in Line 8.

Lines 9 through 11 end the algorithm if all constraints are active or all constraints are inactive, because in those cases there is nothing to allocate risk from or to respectively. Lines 12 through 14 reassign the risk allocated to all inactive constraints such that the new allocation is less than

its previous value but greater than $\delta_{i,min}$, which is the lower bound for the solution found in line 4 to still be valid.

This allows the $\delta_{residual}$ on line 15 to increase, which is roughly the "unused" risk that can be re-allocated to the active constraints, which happens uniformly on lines 16-18.

Algorithm 1: The IRA Algorithm

1 $\delta_i \leftarrow \Delta/N$ for all i
2 $\bar{J}^* \leftarrow \infty$
3 **do**
4 $\quad \bar{J}^*_{prev} \leftarrow \bar{J}^*$
5 $\quad$ Determinize the problem given all the δ_i
6 $\quad$ Solve the determinized version and get the new $\bar{J}^*$
7 $\quad$ Partition the chance constraints into two sets, $\mathcal{C}_{active}$ and $\mathcal{C}_{inactive}$
8 $\quad N_{active} \leftarrow |\mathcal{C}_{active}|$
9 $\quad$ **if** $N_{active} = 0$ *or* $N_{active} = N$ **then**
10 $\quad\quad$ **Break**
11 $\quad$ **end**
12 $\quad$ **forall the** *constraints* i *in* $\mathcal{C}_{inactive}$ **do**
13 $\quad\quad \delta_i \leftarrow \alpha\delta_i + (1-\alpha)\left(1 - \mathrm{cdf}_i(g_i - \boldsymbol{h}_i^T \bar{\boldsymbol{x}}_i)\right)$
14 $\quad$ **end**
15 $\quad \delta_{residual} = \Delta - \sum_{i=0}^{N} \delta_i$
16 $\quad$ **forall the** *constraints* i *in* $\mathcal{C}_{active}$ **do**
17 $\quad\quad \delta_i \leftarrow \delta_i + \delta_{residual}/N_{active}$
18 $\quad$ **end**
19 **while** $|\bar{J}^* - \bar{J}^*_{prev}| > \epsilon$

And that's it! In the remainder of this document, we'll present how IRA can be used with disjunctive constraints, which is useful in many circumstances but won't be tested in the problem set.

Disjunctive Constraints

Note! While this section is cool, it is not required to do the problem set.

We frequently encounter disjunctions of constraints, one of which must hold. We consider that the constraint C_i is actually a disjunction of M constraints, each of which act on the same vector $\boldsymbol{x}_i$,

$$C_i = \bigvee_{j=1}^{M} C_{i,j}.$$

In this case, we could write the joint chance constraint as

$$P\left(\bigwedge_{i=1}^{N}\bigvee_{j=1}^{M} C_{i,j}\right) \geq 1-\Delta.$$

As before, we create an individual chance constraint for each conjunction, producing

$$P\left(\bigvee_{j=1}^{M} C_{i,j}\right) \geq 1-\delta_i.$$

Now, we use a principle of disjunctive distributions,

$$P\left(\bigvee_{j=1}^{M} C_{i,j}\right) \geq P\left(C_{i,j}\right) \; \forall \, j \in \{1,...,M\}.$$

Using this principle, we can satisfy the disjunctive constraint by encoding a disjunction of individual constraints, each with the same risk:

$$\bigvee_{j=1}^{M} \left[P\left(C_{i,j}\right) \geq 1-\delta_i\right].$$

So long as one of these constraints is satisfied, the joint constraint will also be satisfied. If multiple are satisfied, the risk has been allocated conservatively. This encoding can be achieved in a mixed integer linear program. Through the use of binary variables, we can state that at least one constraint must be satisfied. For the Gaussian RMPC problem considered above, we encode:

$$\begin{aligned} & P\left(\bigwedge_{i=1}^{N}\bigvee_{j=1}^{M} \boldsymbol{h}_{i,j}^T \boldsymbol{x}_i \leq g_{i,j}\right) \geq 1-\Delta \\ \Rightarrow \quad & \bigwedge_{i=1}^{N}\bigvee_{j=1}^{M} \left[\boldsymbol{h}_{i,j}^T \bar{\boldsymbol{x}}_i \leq g_{i,j} - \sqrt{2\boldsymbol{h}_{i,j}^T \Sigma_{x,i} \boldsymbol{h}_{i,j}}\,\mathrm{erf}^{-1}(1-2\delta_i)\right]. \end{aligned}$$

References

Ono, Masahiro, and Brian C. Williams. "Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint." Decision and Control, 2008. CDC 2008. 47th IEEE Conference on. IEEE, 2008.

Ono, Masahiro. "Robust, goal-directed plan execution with bounded risk." Diss. Massachusetts Institute of Technology, 2012.

16.412J / 6.834J Cognitive Robotics

Spring 2016