Ellipsoidal Toolbox (ET)

Alex A. Kurzhanskiy and Pravin Varaiya

***Abstract*— Ellipsoidal Toolbox is the first free MATLAB package that implements the operations of ellipsoidal calculus: geometric (Minkowski) sums and differences of ellipsoids, intersections of ellipsoids, and ellipsoids with hyperlanes and polyhedra. The toolbox uses ellipsoidal methods to compute forward and backward reach sets of continuous- and discrete-time piecewise affine systems. Forward and backward reach sets can be also computed for continuous-time piece-wise linear systems with disturbances.**

I. INTRODUCTION

Computation of reach sets of controlled linear systems with convex bounds on control and initial conditions boils down to performing set-valued operations — unions, intersections, geometric sums and differences — of convex sets. Two basic objects are used as convex approximations: various kinds of polytopes (general polytopes, zonotopes, parallelotopes, rectangular polytopes), and ellipsoids.

The Multi Parametric Toolbox (MPT) for Matlab implements reachability analysis for general polytopes [8], [9]. MPT computes the reach set at every time step as the geometric sum of two polytopes. The procedure consists in finding the vertices of the resulting polytope and calculating their convex hull. Polytopes can give arbitrarily close approximations to any convex set, but the number of vertices can grow prohibitively large and the computation of a polytope by its convex hull becomes intractable for large number of vertices in high dimensions.

For a given polytope of initial conditions, d/dt computes the evolution in time of this polytope's extreme points [18], and approximates the resulting set by unions of rectangular polytopes [17]. Rectangular polytopes are easy to represent, but a large number of rectangles must be used to assure the approximations are accurate enough.

Using zonotopes for external approximation of reach sets [19], [20] brings the benefit of a more compact representation than general polytopes. But the benefit appears to diminish greatly if one needs to plot the reach set, compute its intersection with with a given object, because the zonotope must first be converted to a polytope on which the required operations are performed.

Research supported by NSF Grants CCR-00225610 and ECS-0424445. The authors are with the EECS Department at the University of California, Berkeley. They can be reached at `(akurzhan,varaiya)@eecs.berkeley.edu`

CheckMate [16] is a Matlab toolbox to evaluate specifications for trajectories of an autonomous (uncontrolled) system starting from the set of initial (continuous) states corresponding to the parameter values at the vertices of the parameter set. The method of oriented rectangluar polytopes for external approximation of reach sets is introduced in [15].

The level set method [13], [14] deals with general nonlinear controlled systems and gives exact representation of their reach sets, but requires solving the HJB equation, which makes it impractical for systems of dimension higher than three.

Requiem [22] is a Mathematica notebook which, given a linear system, the set of initial conditions and control bounds, symbolically computes the exact reach set, using the experimental quantifier elimination package. Quantifier elimination is the removal of all quantifiers (the universal quantifier $\forall$ and the existential quantifier $\exists$) from a quantified system. Each quantified formula is substituted with quantifier-free expression with operations $+$, $\times$, $=$ and $<$. It is proved in [21] that if A is constant and nilpotent or is diagonalizable with rational real or purely imaginary eigenvalues, the quantifier elimination package returns a quantifier free formula describing the reachable set.

Ellipsoidal Toolbox (*ET*) implements in MATLAB the ellipsoidal calculus [3] and its application to the reachability analysis of continuous-time [4], discrete-time [6] affine systems, and closed-loop control of linear systems with disturbances [5]. *ET* offers these benefits: the complexity of the reach set representation grows quadratically with the dimension of the state space and linearly with the number of time steps; the reach set can be approximated with any accuracy through external and internal ellipsoids; analytical expressions for the control that steers the system to a desired target state.

ET can be downloaded from [1]. Some routines of the toolbox solve optimization problems. YALMIP [10], [11] is used as an interface to external optimization package SeDuMi [12]. Both, YALMIP and SeDuMi are included in the *ET* distribution, so the user does not need to download them separately.

II. ELLIPSOIDAL CALCULUS

The support function of a set $\mathcal{X} \subseteq \mathbf{R}^n$ is

$$\rho(l \mid \mathcal{X}) = \sup_{x\in\mathcal{X}} \langle l, x\rangle, \quad l \in \mathbf{R}^n;$$

here $\langle\cdot,\cdot\rangle$ denotes inner product.

The ellipsoid $\mathcal{E}(q,Q)$ with center $q \in \mathbf{R}^n$ and shape matrix $Q \in \mathbf{R}^{n\times n}$, $Q = Q^T \geq 0$, is the set with

$$\rho(l \mid \mathcal{E}(q,Q)) = \langle l, q\rangle + \langle l, Ql\rangle^{1/2}, \ \forall l \in \mathbf{R}^n.$$

The affine transformation of an ellipsoid is an ellipsoid:

$$A\mathcal{E}(q,Q) + b = \mathcal{E}(Aq + b, AQA^T), \ A \in \mathbf{R}^{m\times n}, \ b \in \mathbf{R}^m.$$

If $b = 0$ and the rows of matrix A are orthonormal, the transformation projects the ellipsoid onto the subspace whose basis is specified by the rows of A.

ET defines class `ellipsoid` with fields specifying the center and the shape matrix. Objects of type `ellipsoid` can be concatenated into two-dimensional arrays. Most operations with ellipsoids implemented as methods of the `ellipsoid` class work with ellipsoidal arrays as well as single objects.

```
>> E1 = ellipsoid([1; 7], [5 0; 0 36]);
>> E2 = ell_unitball(2);
>> EE = [E1 E2 (3*E1 - [4; 1])]; % array
```

The geometric (Minkowski) sum of k ellipsoids is

$$\mathcal{E}(q_1,Q_1)\oplus\cdots\oplus\mathcal{E}(q_k,Q_k) = \{x = \sum_{i=1}^{k} y_i \mid y_i \in \mathcal{E}(q_i,Q_i)\},$$

which is not in general an ellipsoid, but can be approximated by families of external and internal ellipsoids parametrized by vector $l \in \mathbf{R}^n$:

$$\bigcup_l \mathcal{E}_l^- = \mathcal{E}(q_1,Q_1) \oplus \cdots \oplus \mathcal{E}(q_k,Q_k) = \bigcap_l \mathcal{E}_l^+.$$

As the number of directions l increases, the more accurate will be the approximation.

```
>> L = [1 0; 1 2; 0 1]'; % 3 values of l
>> EA = minksum_ea(EE, L);
>> IA = minksum_ia(EE, L);
```

`EA` is an array of three ellipsoids that externally approximate the geometric sum of ellipsoids in `EE`, and array `IA` contains three internal approximating ellipsoids.

The geometric (Minkowski) difference of two ellipsoids is

$$\mathcal{E}(q_1,Q_1)\dot{-}\mathcal{E}(q_2,Q_2) = \{x \mid x + \mathcal{E}(q_2,Q_2) \subseteq \mathcal{E}(q_1,Q_1)\}.$$

This set is nonempty iff $\mathcal{E}(0,Q_2) \subseteq \mathcal{E}(0,Q_1)$. It is not in general an ellipsoid but, as in the case of geometric sum, it can be approximated by parametrized families of external and internal ellipsoids:

$$\bigcup_l \mathcal{E}_l^- = \mathcal{E}(q_1,Q_1)\dot{-}\mathcal{E}(q_2,Q_2) = \bigcap_l \mathcal{E}_l^+.$$

Unlike geometric sum, however, ellipsoidal approximations for the geometric difference do not exist for every direction l. A vector for which the approximation does not exist, is called a *bad direction*. Given two ellipsoids $\mathcal{E}(q_1,Q_1)$ and $\mathcal{E}(q_2,Q_2)$ with $\mathcal{E}(0,Q_2) \subseteq \mathcal{E}(0,Q_1)$, l is a bad direction if

$$\frac{\langle l, Q_1 l\rangle^{1/2}}{\langle l, Q_2 l\rangle^{1/2}} > \lambda,$$

where λ is the minimal root of polynomial $\det(Q_1 - \lambda Q_2)$.

```
>> EA = minkdiff_ea(E1, E2, L);
>> IA = minkdiff_ia(E1, E2, L);
```

Array `EA` contains one external and array `IA` contains one internal ellipsoid, because in this example only one of the three directions specified by matrix `L` is not bad.

The hyperplane $H(v,c)$ with normal $v \in \mathbf{R}^n$ and scalar c is

$$H(v,c) = \{x \in \mathbf{R}^n \mid \langle v, x\rangle = c\}.$$

ET defines class `hyperplane` with fields specifying the normal and the scalar. As with ellipsoids, hyperplanes can be concatenated into two-dimensional array, and all the functions that accept `hyperplane` object as parameter work with hyperplane array as well as a single hyperplane.

```
>> H = hyperplane([1; 1], 3);
>> HH = [H hyperplane([0; 1])]; % array
```

Hyperplane $H(v,c)$ defines two closed halfspaces

$$S_1 = \{x \mid \langle v, x\rangle \leq c\} \text{ and } S_2 = \{x \mid \langle v, x\rangle \geq c\}.$$

To avoid confusion, *ET* assumes that $H(v,c)$ defines S_1. In order to refer to S_2, the same hyperplane should be specified as $H(-v,-c)$.

The distance between ellipsoid $\mathcal{E}(q,Q)$ and point $a \in \mathbf{R}^n$ is

$$\mathbf{dist}(\mathcal{E}(q,Q), a) = \max_{\langle l,l\rangle=1} \left(\langle l, a\rangle - \langle l, q\rangle - \langle l, Ql\rangle^{1/2}\right).$$

The distance between ellipsoids $\mathcal{E}(q_1,Q_1)$ and $\mathcal{E}(q_2,Q_2)$ is

$$\mathbf{dist}(\mathcal{E}(q_1,Q_1), \mathcal{E}(q_2,Q_2)) = \max_{\langle l,l\rangle=1} \left(\langle l, q_1\rangle - \langle l, Q_1 l\rangle^{1/2} - \langle l, q_2\rangle - \langle l, Q_2 l\rangle^{1/2}\right).$$

The distance between ellipsoid $\mathcal{E}(q,Q)$ and hyperplane $H(v,c)$ is

$$\mathbf{dist}(\mathcal{E}(q,Q), H(v,c)) = \frac{|c - \langle v, q\rangle| - \langle v, Qv\rangle^{1/2}}{\langle v, v\rangle^{1/2}}.$$

The distance is negative if the point is inside the ellipsoid, or the two ellipsoids and the ellipsoid and hyperplane have a non-empty intersection.

```
>> DP = distance(EE, [0; 0]);
>> DE = distance(E1, EE);
>> DH = distance(E1, HH);
```

Array `DP` consists of three elements — distances from each of the ellipsoids in `EE` to the origin. Array `DE` contains distances from ellipsoid `E1` to each of the ellipsoids in `EE`, and `DH` contains distances from `E1` to each of the hyperplanes in `HH`.

If it is non-empty, the intersection of ellipsoid and hyperplane is an ellipsoid.

```
>> I = hpintersection(EE, H);
```

Array `I` contains three `ellipsoid` objects, two of which are empty, because only one ellipsoid in `EE` intersects with hyperplane `H`.

The intersection of two ellipsoids, or an ellipsoid and a half-space, is not generally an ellipsoid, but can be approximated by minimum volume external and maximum volume internal ellipsoids.

```
>> IE = intersection_ea(EE(1), EE(3));
>> II = intersection_ia(EE(1), EE(3));
>> HE = intersection_ea(E1, H);
>> HI = intersection_ia(E1, H);
```

`IE` is the minimum volume ellipsoid containing the intersection of the first and the third ellipsoids in array `EE`, `II` is the maximum volume ellipsoid that lies inside this intersection. Similarly, ellipsoids `HE` and `HI` are external and internal approximations of the intersection of ellipsoid `E1` with the halfspace defined by the hyperplane `H`.

III. REACHABILITY

Consider the dynamical system

$$\dot{x}(t) = A(t)x(t) + B(t)u(t, x(t)) + G(t)v(t), \qquad (1)$$

wherein $x \in \mathbf{R}^n$ is the state, $u \in \mathbf{R}^m$ is the control and $v \in \mathbf{R}^d$ is the disturbance. Matrices $A(t)$, $B(t)$ and $G(t)$ are continuous and take their values in $\mathbf{R}^{n\times n}$, $\mathbf{R}^{n\times m}$ and $\mathbf{R}^{n\times d}$ respectively. Control $u(t, x(t))$ and disturbance $v(t)$ are measurable functions restricted by ellipsoidal boundaries $\mathcal{E}(p(t), P(t))$ and $\mathcal{E}(q(t), Q(t))$. If matrix $Q(t) \equiv 0$, then the system (1) becomes ordinary affine system with known $v(t) = q(t)$. If matrix $G(t) \equiv 0$, then (1) reduces to linear system.

ET defines class `linsys` describing the dynamical system.

```
>> % define matrices A, B, G,
>> % control bounds U,
>> % disturbance bounds V
>> sys = linsys(A, B, U, G, V);
```

The same constructor is called with an additional parameter to create `linsys` object describing discrete-time system. Matrices `A`, `B`, `G` can be of type `double` if they are constant, or symbolic if they depend on time. Similarly, bounds `U` and `V` can be of type `ellipsoid` if they are constant, or be structures with the same fields as those of `ellipsoid` class but symbolic if they depend on time.

Given initial time t_0 and the set of initial conditions $\mathcal{E}(x_0, X_0)$, the *reach set* $\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0))$ of system (1) at time $t > t_0$ is the set of all states x for each of which there exist initial condition $x^0 \in \mathcal{E}(x_0, X_0)$ and control $u(\tau, x(\tau))$ that for every disturbance $v(\tau) \in \mathcal{E}(q(\tau), Q(\tau))$ assigns trajectory $x(\tau)$ satisfying

$$\dot{x}(\tau) \in A(\tau)x(\tau) + B(\tau)u(\tau, x(\tau)) + G(\tau)v(\tau),$$

where $t_0 \leq \tau \leq t$, such that $x(t_0) = x_0$ and $x(t) = x$.

Observe that when disturbances are present, the reach set is obtained through closed-loop control. The closed-loop reach set can be empty. This happens for example, if the set of initial conditions is reduced to a single state x_0 and control $u(t)$ is fixed, but the disturbance bound $\mathcal{E}(q(t), Q(t))$ is nondegenerate ellipsoid for all t. A sufficient condition for reach set $\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0)$ to be non-empty is

$$\mathcal{E}(0, G(\tau)Q(\tau)G^T(\tau)) \subseteq \mathcal{E}(0, B(\tau)P(\tau)B^T(\tau))$$

for $t_0 \leq \tau \leq t$.

In the absence of disturbance (when v is fixed), reach set $\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0))$ is the set of all states x to which the system can be steered at time t through all possible controls starting at any $x^0 \in \mathcal{E}(x_0, X_0)$ at time t_0. In this case, open-loop and close-loop control are the same, and the reach set is always non-empty.

$\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0))$ is compact convex set. If non-empty, it can be approximated by the parametrized families of external and internal ellipsoids, $\mathcal{E}(x_c(t), X_l^+(t))$ and $\mathcal{E}(x_c(t), X_l^-(t))$ respectively:

$$\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0)) = \bigcup_l \mathcal{E}(x_c(t), X_l^-(t)) = \bigcap_l \mathcal{E}(x_c(t), X_l^+(t)),$$

where $x_c(t)$ satisfies (1) with $u(t, x) = p(t)$ and $v(t) = q(t)$, and shape matrices $X_l^+(t)$ and $X_l^-(t)$ are the solutions of differential equations that depend on parameter $l \in \mathbf{R}^n$ (see [4], [5]).

ET implements class `reach` that describes the evolution in time of the reach set in terms of external and internal ellipsoidal arrays.

```
>> % define initial set - ellipsoid X0,
>> % initial time t0 and time t,
>> % matrix of directions L
>> RS = reach(sys, X0, L, [t0 t]);
```

At this point, variable `RS` contains the reach set approximations for the system `sys` and the set of initial conditions `X0` evolving in time from `t0` to `t`, computed for given directions `L`. By default, both, external and internal, approximations are computed.

The reach set approximation data can be extracted in the form of arrays of external and internal ellipsoids:

```
>> EA = get_ea(RS); % external
>> [IA, tt] = get_ia(RS); % internal
```

The number of columns in the ellipsoidal arrays `EA` and `IA` is defined by a configurable parameter. It is the number of time values in our time interval at which the approximations are evaluated. These time values are returned in the optional output parameter, array `tt`, whose length equals the number of columns in `EA` and `IA`. The intersection of ellipsoids in a particular column of `EA` gives an external ellipsoidal approximation of the reach set at the corresponding time. The union of ellipsoids in the same column of `IA` is an internal ellipsoidal approximation of this set at this time. Each row of `EA` and `IA` corresponds to the column of matrix `L`, specifying the value of parameter l.

In general, taking more different values of direction l gives a better reach set approximation. The computation complexity grows linearly with number of directions. There is no universal rule for the choice of l. For two- or three-dimensional systems, one may take vectors uniformly distributed on the unit sphere. For rough approximation, any single random vector in $\mathbf{R}^n$ works.

If the reach set data in the variable `RS` are not sufficiently accurate, they can be refined by computing additional approximations for some extra directions.

```
>> % define additional directions L2
>> RS = refine(RS, L2);
>> IA = get_ia(RS);
```

Now the number of rows in array `IA` has increased by the number of columns in the matrix `L2`.

Refinement may be useful, for instance, when one cannot determine if the actual reach set intersects a given object (ellipsoid, hyperplane or polytope), because its external approximation does intersect this object but the internal approximation does not.

```
>> e = intersect(RS, H, 'e');
>> i = intersect(RS, H, 'i');
```

Variable `e` equals 1 if the reach set external approximation intersects with hyperplane `H`, otherwise it equals 0. Variable `i` has a similar meaning for the internal approximation.

We may be interested in the reach set data for some smaller time interval than the one for which `RS` was computed, or in a snapshot of the reach set at given time.

```
>> % define t1, t2: t0 <= t1 <= t2 <= t
>> CT = cut(RS, [t1 t2]);
>> SS = cut(RS, t);
```

Both variables, `CT` and `SS`, are of type `reach`. `CT` contains reach set data for the time interval from `t1` to `t2`, and `SS` gives the snapshot of the reach set at time `t`.

The reach set satisfies the *semigroup* property:

$$\mathcal{X}(t, t_0, \mathcal{E}(x_0, X_0)) = \mathcal{X}(t, \tau, \mathcal{X}(\tau, t_0, \mathcal{E}(x_0, X_0)))$$

for $t_0 \leq \tau \leq t$.

Variable `RS` contains reach set data up to time `t`. It is possible to continue the reach set computation for the new time horizon, using `t` as new initial time and the reach set at time `t` as the new set of initial conditions.

```
>> T = t + 10; % new terminating time
>> RS2 = evolve(RS, T);
```

Variable `RS2` refers to the `reach` object with reachability data for the time interval from `t` to `T`.

It is possible that the dynamics and inputs of the system change at a certain time, and from that point on, the system evolves according to a new set of differential equations. The same function `evolve` is used in case of such switched systems.

```
>> % define new linsys object: sys2
>> RS2 = evolve(RS, T, sys2);
```

It is expected that `linsys` object `sys2` has the same state-space dimension as `sys`. Thus function `evolve` implements the semigroup property.

Remark. Refinement of the reach set approximation through `refine` function cannot treat `reach` objects that result from `cut` or `evolve` operations.

Given terminating time t_1 and target set $\mathcal{E}(y_1, Y_1)$, the *backward reach set* $\mathcal{Y}(t_1, t, \mathcal{E}(y_1, Y_1))$ of system (1) at time $t < t_1$ is the set of all states y for each of which there exist terminating state $y^1 \in \mathcal{E}(y_1, Y_1)$ and control $u(\tau, y(\tau))$ that for every disturbance $v(\tau) \in \mathcal{E}(q(\tau), Q(\tau))$ assigns trajectory $y(\tau)$ satisfying

$$\dot{y}(\tau) \in A(\tau)y(\tau) + B(\tau)u(\tau, y(\tau)) + G(\tau)v(\tau),$$

where $t \leq \tau \leq t_1$, $y(t) = y$ and $y(t_1) = y^1$.

Like the forward reach set, the backward reach set is convex and compact, and, when non-empty, can be over- and under-approximated by the ellipsoidal families parametrized by direction $l \in \mathbf{R}^n$.

In *ET*, backward reach sets are computed by the same constructor `reach` as forward reach sets. Only now, in place of the initial set there goes the target set, and the time interval is inverted, with terminating time first.

```
>> % define target set - ellipsoid Y
>> BRS = reach(sys, Y, L, [T t]);
```

Variable `BRS` refers to the reach set data computed for the target set `Y` backward in time from `T` to `t`.

The semigroup property holds for the backward reach set:

$$\mathcal{Y}(t_1, t, \mathcal{E}(y_1, Y_1)) = \mathcal{Y}(\tau, t, \mathcal{Y}(t_1, \tau, \mathcal{E}(y_1, Y_1)))$$

for $t \leq \tau \leq t_1$.

Function `evolve` works for backward reach sets.

```
>> BRS2 = evolve(BRS, t0); % t0 < t
```

Here, the new terminating time is t, and the new target set (the backward reach set at time t) are both obtained from `BRS`.

All methods of class `reach`, including functions `evolve`, `cut` and `refine`, work equally with objects describing forward and backward reach sets.

The functionality of methods in `linsys` and `reach` classes extends to discrete-time affine systems:

$$x[k+1] = A[k]x[k] + B[k]u[k] + G[k]v[k].$$

In discrete-time case, however, reachability for systems with disturbances is not implemented yet, and while controls $u[k]$ are bounded by $\mathcal{E}(p[k], P[k])$, values $v[k]$ are expected to be known and fixed. Reach set calculation for discrete-time systems is implemented following [6].

Remark. While *ET* computes forward reach sets for systems with any matrices $A[k]$, backward reach sets can be computed only for systems with nonsingular $A[k]$. This warning can be ignored if the discrete system is obtained by sampling of the continuous one.

IV. VISUALIZATION

ET implements several plotting routines to display ellipsoids, their geometric sums and differences, cointinuous- and discret-time reach set external and internal approximations.

Classes `ellipsoid` and `hyperplane` have overloaded method `plot`. Function `minksum` plots geometric sum of finite number of ellipsoids. Function `minkdiff` plots geometric difference of two ellipsoids if it is non-empty. Function `plot_ea` displays reach set external approximation, and for internal approximation function `plot_ia` is used.

All these plotting routines work with objects in $\mathbf{R}^2$ and $\mathbf{R}^3$. For `ellipsoid` and `reach` objects of larger dimensions `projection` methods are implemented.

```
>> % define dimension n > 3
>> Q = rand(n); % random matrix
>> R = ellipsoid(Q*Q');
>> [U, S, V] = svd(Q*Q');
>> BBB = U(:, 1:3); % subspace basis
>> P = projection(R, BBB);
```

Here `R` is a random large dimensional ellipsoid, and `P` is its three-dimensional projection onto the basis defined by its three main semiaxes.

```
>> BB = eye(n);
>> PRS = projection(RS, BB(:, 1:2));
```

Variable `PRS` contains the projection of the reach set `RS` onto (x_1, x_2) state subspace.

Remark. Functions `refine` and `evolve` do not work with `reach` objects that result from `projection` operation.

V. EXAMPLE

There is no explicit implementation of the reachability analysis for hybrid systems in *ET*. However, *ET*'s operations of intersection allow us to work with hybrid systems with affine continuous dynamics, whose guards are ellipsoids, hyperplanes or polyhedra. An example of this type of hybrid system can be found in [23]. Here we indicate how to use *ET* for the reach set computation for such a system.

Consider a hybrid system with two discrete *modes*, *1* and *2*. In *mode1* the continuous dynamics are described by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [k+1] = \begin{bmatrix} 0.98 & 0 & 0 & 0 \\ 0.02 & 0.99 & 0 & 0 \\ 0 & 0.01 & 0.99 & 0 \\ 0 & 0 & 0.01 & 0.98 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [k] + \begin{bmatrix} 0.02 & 0 & 0 \\ 0 & 0 & 0.01 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} [k],$$

and in *mode2* by

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [k+1] = \begin{bmatrix} 0.99 & 0.003 & 0 & 0 \\ 0 & 0.99 & 0.002 & 0 \\ 0 & 0 & 0.99 & 0.002 \\ 0 & 0 & 0 & 0.99 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} [k] + \begin{bmatrix} 0 & 0 & 0.003 \\ 0 & 0 & 0 \\ 0 & -0.003 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} [k] + \begin{bmatrix} 0 \\ 0 \\ -0.19 \\ 0.56 \end{bmatrix},$$

The system starts operating in *mode1* with the set of initial conditions $\mathcal{E}([170\ 180\ 175\ 170]^T, 100I)$ (I is the identity matrix) at $k = 0$, and may switch to *mode2* upon reaching the guard $H([0\ 1\ 0\ 0]^T, 200)$. In both modes, control $u[k] \in \mathcal{E}([180\ 150\ 50]^T, 25I)$.

```
>> % define systems: sys1, sys2
>> N = 100;
>> X0 = ellipsoid(100*eye(4));
>> X0 = X0 + [170 180 175 170]';
>> L = [1 0 0 0]';
>> H = hyperplane([0 1 0 0]', 200);
```

Variables `sys1` and `sys2` are `linsys` objects corresponding to *mode1* and *mode2*; `N` is the number of time steps for which we will compute the reach set; `X0` is the set of initial states; `L` contains a single direction for which the approximation will be computed; `H` is the guard.

We start by computing the reach set for *mode1* and finding its intersections with the guard.

```
>> RS1 = reach(sys1, X0, L, N);
>> EA = get_ea(RS1); % 101 ellipsoids
```

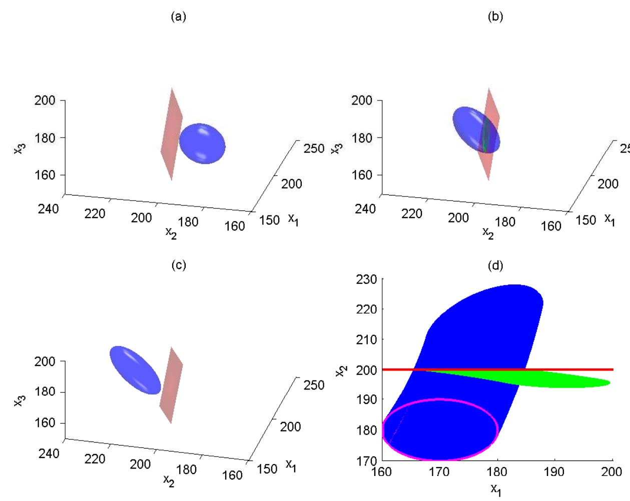
`EA` is an array of external ellipsoids whose first ellipsoid corresponds to step $k = 0$, and the last to $k = 100$. Now we can check which of the ellipsoids in `EA` intersect the guard.

```
>> HI = hpintersection(EA, H);
>> D = find(~isempty(HI));
```

Analyzing data in `D`, we find that *mode1* reach set has non-empty intersection with the guard for $18 \leq k \leq 68$. Figure (a) shows the *mode1* reach set projection (blue) onto (x_1, x_2, x_3) subspace at $k = 10$, before the guard (red) crossing, figure (b), at $k = 50$, during the guard crossing, and figure (c), at $k = 80$, after the guard was crossed.

Each non-empty ellipsoid in `HI` represents a new initial condition, and its index in the array minus one is the new initial time for the reach set computation according to the *mode2* dynamics.

```
>> RS2 = [];
```




```
>> for i = 1:size(D, 2)
     X0 = HI(D(i));
     k0 = D(i) - 1;
     rs = reach(sys2, X0, L, [k0 N]);
     RS2 = [RS2 rs];
   end
```

The union of reach sets in array `RS2` forms the reach set for *mode2*.

Figure (d) summarizes what was done. It shows the projection of the reach set trace onto (x_1, x_2) subspace. The system starts evolving in time in *mode1* with some set of initial conditions at $k = 0$. The reach set of *mode1* evolving from $k = 0$ to $k = 100$ is shown in blue, and a bound of the initial set in magenta. Between steps $k = 18$ and $k = 68$ *mode1* reach set crosses the guard. The guard is shown in red. For each of the non-empty intersections of the *mode1* reach set and the guard, *mode2* reach set starts evolving in time until $k = 100$. All *mode2* reach sets are shown in green.

VI. SUMMARY AND OUTLOOK

Ellipsoidal Toolbox is the first free MATLAB package that implements the ellipsoidal calculus and uses ellipsoidal methods for reachability analysis of continuous- and discrete-time affine systems, continuous-time linear systems with disturbances and switched systems, whose dynamics change at known times. The reach set computation for hybrid systems whose guards are hyperplanes or polyhedra is not implemented explicitely, but *ET* can be used for such computation.

More elaborate discussion on computational techniques used in *ET*, their accuracy and efficiency in comparison with other methods mentioned in the introduction, together with examples and MATLAB code can be found in [2].

In future versions of the toolbox we intend to implement additional basic operations with ellipsoids, discrete-time systems with disturbance, reachability analysis for state-constrained and obstacle problems; and ellipsoidal methods for stochastic control and estimation.

VII. ACKNOWLEDGMENTS

The authors would like to thank Alexander B. Kurzhanski, Manfred Morari, Johan Löfberg, Michal Kvasnica and Goran Frehse for their support of this work by useful advice and encouragement.

REFERENCES

[1] Ellipsoidal Toolbox homepage: *www.eecs.berkeley.edu/˜akurzhan/ellipsoids*
[2] A.A.Kurzhanskiy, P.Varaiya (2006). *Ellipsoidal Toolbox*. Technical Report EECS-2006-46, EECS Department, UC Berkeley.
[3] A.B.Kurzhanski, I.Vályi (1997). *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser, Boston, ser. SCFA.
[4] A.B.Kurzhanski, P.Varaiya (2000). *On ellipsoidal techniques for reachability analysis*. In: *Optimization Methods and Software*, Vol.17, pp.177-237, Taylor & Francis.
[5] A.B.Kurzhanski, P.Varaiya (2001). *Reachability analysis for uncertain systems - the ellipsoidal technique*. In: *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, Vol.9, pp.347-367.
[6] A.A.Kurzhanskiy, P.Varaiya (2005). *Ellipsoidal Techniques for Reachability Analysis of Discrete-Time Linear Systems*. Submitted to IEEE Transactions on Automatic Control.
[7] P.Varaiya (1998). *Reach set computation using optimal control*. In: *Proc. of KIT Workshop on Verification on Hybrid Systems*. Verimag, Grenoble.
[8] M.Kvasnica, P.Grieder, M.Baotić, M.Morari (2004). *Multi-Parametric Toolbox (MPT)*. In R.Alur and G.J.Pappas, editors, *Hybrid Systems: Computation and Control*, Vol.2993 of *Lecture Notes in Computer Science*, pp.448-462. Springer-Verlag.
[9] Multi-Parametric Toolbox homepage: *control.ee.ethz.ch/˜mpt*
[10] J. Löfberg (2004). *A Toolbox for Modeling and Optimization in MATLAB*. In: Proceedings of CACSD Conference, Taipei, Taiwan.
[11] YALMIP homepage: *control.ee.ethz.ch/˜joloef/yalmip.php*.
[12] SeDuMi homepage: *sedumi.mcmaster.ca*
[13] I.Mitchell, C.Tomlin (2000). *Level set methods for computation in hybrid systems*, In N.Lynch and B.H.Krogh, editors, *Hybrid Systems: Computation and Control*, Vol.1790 of *Lecture Notes in Computer Science*. Springer-Verlag.
[14] Level Set Toolbox homepage: *www.cs.ubc.ca/˜mitchell/ToolboxLS*
[15] O.Stursberg, B.H.Krogh (2003). *Efficient representation and computation of reachable sets for hybrid systems*, In O.Maler and A.Pnueli, editors, *Hybrid Systems: Computation and Control*, Vol.2623 of *Lecture Notes in Computer Science*, pp.482-497. Springer-Verlag.
[16] CheckMate homepage: *www.ece.cmu.edu/˜webk/checkmate*
[17] E.Asarin, O.Bournez, T.Dang, O.Maler (2000). *Approximate reachability analysis of piecewise linear dynamical systems*, In N.Lynch and B.H.Krogh, editors, *Hybrid Systems: Computation and Control*, Vol.1790 of *Lecture Notes in Computer Science*, pp.21-31. Springer-Verlag.
[18] d/dt homepage: *www-verimag.imag.fr/˜tdang/ddt.html*
[19] A.Girard (2005). *Reachability of uncertain linear systems using zonotopes*. In *Hybrid Systems: Computation and Control*, *Lecture Notes in Computer Science*. Springer-Verlag.
[20] MATISSE homepage: *www.seas.upenn.edu/˜agirard/Software/MATISSE*
[21] G.Lafferriere, G.J.Pappas, S.Yovine (2001). *Symbolic Reachability Computation for Families of Linear Vector Fields*. In: *Journal of Symbolic Computation* No.32, pp.231-253, Academic Press.
[22] Requiem homepage: *www.seas.upenn.edu/˜hybrid/requiem/requiem.html*
[23] L.Muñoz, X.Sun, R.Horowitz, L.Alvarez (2003). *Traffic density estimation with the cell transmission model*, In: *Proceedings of the American Control Conference*, Denver, Colorado, USA, pp.3750-3755.
[24] S.Boyd, L.Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.