Consider a Sliding object with *n* points of contact

y⃗*f e*

⃗

*pe*

⃗*fi*

An external force acts upon an object,

***pe***= [*pex*, *pey*]′

at location , resulting in a

force and moment at origin. We can

write this as a unit external *wrench, w,*

*ρ*

with scaling factor :

[*fex*, *fey*, *pe* × *fe*] = *ρ*[*wex*,*wey*,*wem*]

x

As the external force magnitude

increases, the object will eventually start

to slide. It will have some (as yet

unknown) instantaneous *twist*

[*vx*, *vy*,*ωz*]′

**CoR**

which is equivalent to having an instantaneous CoR (center of rotation) at some location in the plane.

1

Sliding object with *n* points of contact

y⃗*f e*

⃗

*pe*

⃗

An external force acts upon an object,

***pe***= [*pex*, *pey*]′

at location , resulting in a

force and moment at origin. We can

write this as a unit external wrench with

scaling factor:

[*fex*, *fey*, *pe* × *fe*] = *ρ*[*wex*,*wey*,*wem*]

Each sliding point creates a friction

x

***f****i* = [*fxi*, *fyi*]

force, where the magnitude

*pi*

of the force is known:

| ***f****i*| = *μfni*

⃗*fi*

⃗

*rci*

*ρ*

**CoR**

*wex wey wem*

Thus the quasi-static equilibrium equations are:

∑ *μfnirciy*

|*rci*|

**=**

−∑ *μfnircix*

|*rci*|

|*rci*| *piy* +*rciy*

∑ *μfni*(−*rcix*

|*rci*| *pix*)

*What are the unknown quantities in these equations?*

friction forces and moment at origin for all sliding points, for a given (unkown) COR location

2

Sakurai (MIT 1990) example - used for assignment

***f****i*

***r****ci*

p3

y

***f****e*

x

⃗*f epe*

An external force acts upon object, pulling, as with a string attached at , resulting in a force and moment at origin.

⃗*f e*

We know the direction of and its line *pe*

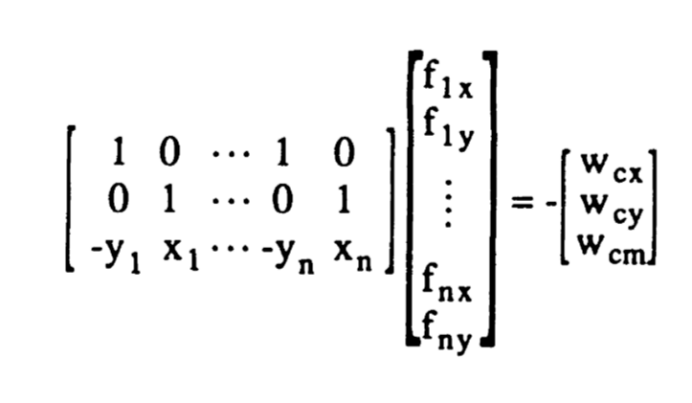
of action, given by , but not the

~~o~~ pe

(COR location)

p2

p1



magnitude. Thus we know the ratios

*fey* /*fex* | *fe* |/*me me*

and where is the

moment. We write this as a unit external

*ρ*

wrench with scaling factor :

[*fex*, *fey*, *pe* × *fe*] = *ρ*[*wex*,*wey*,*wem*]

Sakura (eq 4.2.9) writes the equilibrium

equation as:

[*W*] ⋅ ***f*** = − *ρ****w f***

where is a vector with

all the friction forces (x and y

components) and [*W*] is essentially a

wrench matrix of the sort we have seen

before.

Note: Sakurai uses ‘I’ for *ρ*, which can be a bit confusing…

3

Possibly it rotates about one of the contact points?

***f****i*

***r****ci*

p3

y

***f****e*

x

In this case, the COR is assumed known, so we know immediately the friction forces for all the other sliding contacts (*j* ≠ *i* ).

Again,

~~o~~ pe

(COR location)

p1

p2

1 0 *wex*

*fcx*

[*fex*, *fey*, *pe* × *fe*] = *ρ*[*wex*,*wey*,*wem*]

So we can rewrite things as:∑ *μfnirciy*

|*rci*|

Possibly COR is at one of the contact points

0 1 *wey* 0 0 *wem*

*fcy ρ*

**=**

−∑ *μfnircix*

|*rci*|

|*rci*| *piy* +*rciy*

(shown as p1 here) ***r****ci*

= vector from COR

∑ *μfni*(−*rcix*

|*rci*| *pix*)

to a sliding point

*What are the unknown quantities in these equations?*

net friction forces and moment at origin for all sliding points, given COR location

4

If it rotates about one of the contact points

***f****i*

***r****ci*

p3

y

***f****e*

x

In this case, the COR is assumed known so we know immediately the friction forces for all the other sliding contacts (*j* ≠ *i* ).

~~o~~ pe

(COR location)

We set up equilibrium equations

p1

p2

for the unknown forces

*fcx*, *fcy*

Possibly COR is at one

1 0 *wex*

*fcx*

at the COR location and *ρ*

∑ *μfnirciy*

|*rci*|

of the contact points (shown as p1 here)

0 1 *wey* 0 0 *wem*

*fcy ρ*

**=**

−∑ *μfnircix*

|*rci*|

|*rci*| *piy* +*rciy*

***r****ci*

= vector from COR

∑ *μfni*(−*rcix*

|*rci*| *pix*)

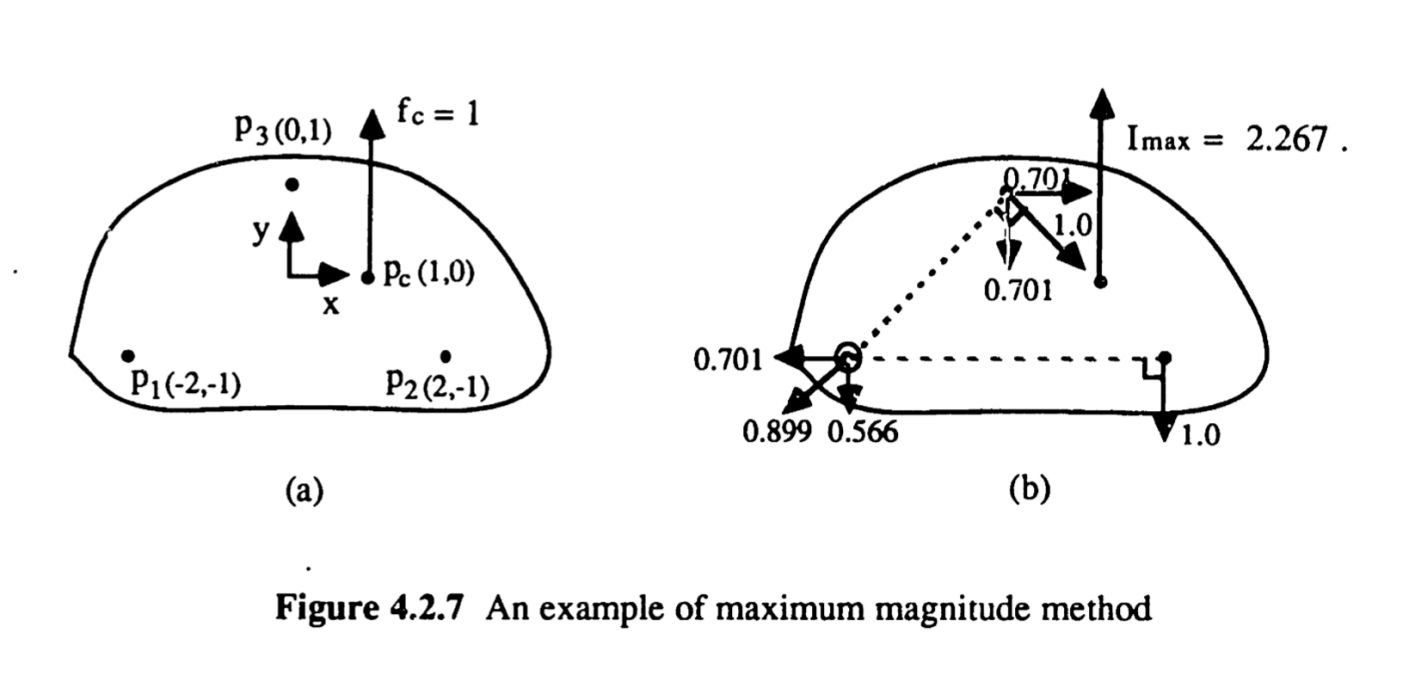
to a sliding point

net friction forces and moment at origin for all sliding points, given COR location

If solution results in values of fcx, fcy that are below the friction limit, it is the solution.

5

Sakurai example where CoR happens to be at p1



Note that Sakurai does not actually solve this special case using the method on the last slide. Instead he applies the general Maximum Work optimization method (see next slide), which gives the same result.

6

More generally…

p3

***f****ri*

y

***f****e*

(This method is implemented in SakuraiFriction.py)

**Sakurai eq 4.2.18 - 4.2.20:**

[*W*]***f***

produces a net friction wrench on the body.

At sliding, it is equal and opposite to the external ***w fe***

wrench, , produced by .

***f***

So we seek that will maximize the product

***r****ci*

x

([*W*]***f***) ⋅ ***w*** .

~~o~~ pe

We can rewrite this as **Maximize**:

p1

p2

*ρ* = (−*w*′⋅ [*W*]) ***f***

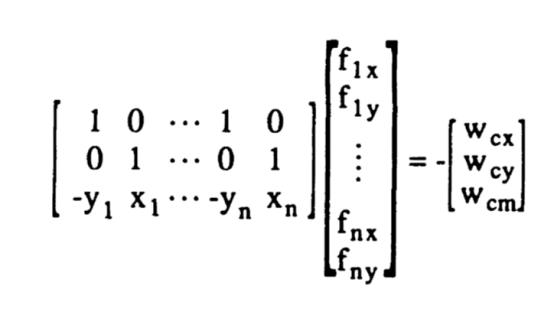
*ρ*

where is a scaling factor and *w* is a unit wrench in the direction of the external wrench.

**Subject** to the constraint that the net friction force must be antiparallel to the external force so:

*w* × [*W* ***f***] = 0

(Aeq, beq in linprog() )

eq (4.2.9) from Sakurai expresses the friction wrench that is equal and opposite to the external wrench

We must also satisfy **bounds** due to friction constraints:

*f* 2*ix* + *f* 2*iy* ≤ (*μf* 2*in*)

which we can approximate with linear friction polygons.

7