Contents


```
function bes=beschleunigung(spiel,farbe)
```

```
if strcmp( farbe, 'rot')
    ich=spiel.rot;
    gegner=spiel.blau;
else
    ich=spiel.blau;
    gegner=spiel.rot;
end
```

```
Not enough input arguments.

Error in beschleunigung (line 3)
if strcmp( farbe, 'rot')
```

Basisbeschleunigungsvektor

```
bes=[0,0];
```

Konstanten

```
bv=1.6; %Bremsweg-Sicherheits-Faktor%

k1=1; % Tankenmatrix Abstandsfaktor
k2=100; % Tankenmatrix Kosinusfaktor
k3=1; % Tankenmatrix Minengefahr???
k4=0; % Tankenmatrix Tankentreffzeitpunkt
```

```
kmw1=10; % Nextmine Abstandsfaktor
kmw2=1; % Kollisionskursfaktor
```

Sonderfälle

```
sonderfall_bande=0;
sonderbw=0;
```

Anfangswerte

```
geradetanke=0;
dang=0;
bande=0;
eckverd=0;
minendang=0;
gerade=0;
```

Persistente Variablen

```
persistent zpverd;
persistent nexteckepkt;
persistent Angriff6;
```

Ecken

```
eol=[0.006,0.994];
eor=[0.994,0.994];
eur=[0.994,0.006];
eul=[0.006,0.006];
```

Abstandsberechnung zu allen 4 Banden

```
abs_rb=1-ich.pos(1);
abs_ob=1-ich.pos(2);
abs_lb=ich.pos(1);
abs_ub=ich.pos(2);
```

Beträge der Geschwindigkeiten

```
ich_ges = norm(ich.ges);
```

Bremswegberechnungen + Sicherheit

```
bwx=((ich.ges(1))^2/(2*spiel.bes))+0.0101;
bwy=((ich.ges(2))^2/(2*spiel.bes))+0.0101;
bw = (ich_ges)^2/(2*spiel.bes);
```

Basis-Modusbeschleunigungen

```
besdefence=[0,0];
besangriff=[0,0];
bestanken=[0,0];
besmine=[0,0];
```

Abstandsberechnungen

```
abs_geg=norm(gegner.pos-ich.pos);

if abs_rb<=bwx*bv && ich.ges(1)>0 || abs_ob<=bwy*bv && ich.ges(2)>0 || abs_ub<=bwy*bv && ich.ges(2)<0 || abs_ub<=bwy*bv && ich.ges(2)<0
    bande=1;
end
```

Basis-Minen ausweichen

```
zp=ich.pos+(ich.ges/norm(ich.ges)*1);
zpges=[0,0];
zpradius=0;
```

%%

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Funktionen%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%%%

Kollisionsanalyse

```
    function kollision = szk(m1, ges1, r1, m2, ges2, r2)
```

```
    radien = r1 + r2; %Radien addieren
    e=m2 - m1; %Entfernung zu Minen
    f=ges2-ges1; %Vektor zwischen ich.pos und die Richtung der Mine

    alpha=dot(f, f);
    beta=dot(e, f);
    gamma=dot(e,e)-radien^2;
    delta=beta^2 - alpha*gamma;

    if delta>0
        lambda=[(-beta+sqrt(delta))/alpha; (-beta-sqrt(delta))/alpha];
        if lambda(1)<0
            lambda(1)=inf;
        end
        if lambda(2)<0
            lambda(2)=inf;
        end
    else
        lambda=[inf,inf];
    end
    kollision = sortrows(lambda);
end
```

Lambdax

```
function lambdax = szklx(m1, ges1, r1, m2, ges2, r2)

    radien = r1 + r2; %Radien addieren
    e=m2 - m1; %Entfernung zu Minen
    f=ges2-ges1; %Vektor zwischen ich.pos und die Richtung der Mine

    alpha=dot(f, f);
    beta=dot(e, f);
    gamma=dot(e,e)-radien^2;
    delta=beta^2 - alpha*gamma;

    if delta>0
        lambda=[(-beta+sqrt(delta))/alpha; (-beta-sqrt(delta))/alpha];
        if lambda(1)<0
            lambda(1)=inf;
        end
        if lambda(2)<0
            lambda(2)=inf;
        end
    else
        lambda=[inf,inf];
    end
    kollision = sortrows(lambda);
    lambdax=kollision(1);

end
```

Lambdax2

```
function lambdax = szklx2(m1, ges1, r1, m2, ges2, r2)

    radien = r1 + r2; %Radien addieren
    e=m2 - m1; %Entfernung zu Minen
    f=ges2-ges1;%Vektor zwischen ich.pos und die Richtung der Mine

    alpha=dot(f, f);
    beta=dot(e, f);
    gamma=dot(e,e)-radien^2;
    delta=beta^2 - alpha*gamma;

    if delta>0
        lambda=[(-beta+sqrt(delta))/alpha; (-beta-sqrt(delta))/alpha];
        if lambda(1)<0
            lambda(1)=110;
        end
        if lambda(2)<0
            lambda(2)=110;
        end
    else
        lambda=[110,110];
    end
    kollision = sortrows(lambda);
    lambdax=kollision(1);

end
```

Kollisionsanalyse Tanken //// wenn keine Kollision --> gut --> Werte 0,0 nicht inf,inf!!!!!!!

```
function lambdatank=szktanken1(m1, ges1, r1, m2, ges2, r2)
    radien = r1 + r2; %Radien addieren
    e=m2 - m1; %Entfernung zu Minen
    f=ges2-ges1; %Vektor zwischen ich.pos und die Richtung der Mine

    alpha=dot(f, f);
    beta=dot(e, f);
    gamma=dot(e,e)-radien^2;
    delta=beta^2 - alpha*gamma;
```

```
        if delta>0
            lambda=[(-beta+sqrt(delta))/alpha; (-beta-sqrt(delta))/alpha];
            if lambda(1)<0
                lambda(1)=1000;
            end
            if lambda(2)<0
                lambda(2)=1000;
            end
        else
            lambda=[1000,1000];
        end
        kollision = sortrows(lambda);
        lambdatank=kollision(1);
    end
```

Kollisionsanalyse Minenverteidigung

```
    function zeitmine1 = szkmine(m1, ges1, r1, m2, ges2, r2)

        radien = r1 + r2; %Radien addieren
        e=m2 - m1; %Entfernung zu Minen
        f=ges2-ges1; %Vektor zwischen ich.pos und die Richtung der Mine

        alpha=dot(f, f);
        beta=dot(e, f);
        gamma=dot(e,e)-radien^2;
        delta=beta^2 - alpha*gamma;

        if delta>0
            lambda=[(-beta+sqrt(delta))/alpha; (-beta-sqrt(delta))/alpha];
            if lambda(1)<0
                lambda(1)=inf;
            end
            if lambda(2)<0
                lambda(2)=inf;
            end
        else
            lambda=[inf,inf];
        end
        zeitmine1 = lambda(1);
    end
```

Vektorprojektion c

```
    function vecpro=vp(z,pos,posges)
        zw=z-pos;
        ba=(dot(zw,posges)/(dot(zw,zw)))*zw;
        vecpro=ba-posges;
    end
```

Vektorprojektion ba

```
    function ba=vpba(z,pos,posges)
        zw=z-pos;
        ba=(dot(zw,posges)/(dot(zw,zw)))*zw;

    end
```

Orthogonaler Vektor

```
    function ortho=og(a)
        ortho=[a(2),-a(1)];
    end
```

Sumabs

```
    function summeabs=sumabs(xb)
        if spiel.n_tanke>1
            tankdenmatrix=zeros(spiel.n_tanke,1);

            for x=1:spiel.n_tanke

                tankdenmatrix(x,1)=x;
                tankdenmatrix(x,2)=spiel.tanke(x).pos(1);
                tankdenmatrix(x,3)=spiel.tanke(x).pos(2);

            end
            abssum=sum(tankdenmatrix);
            abssumx=(abssum(1,2))/spiel.n_tanke;
            abssumy=(abssum(1,3))/spiel.n_tanke;

            summeabs=norm(spiel.tanke(xb).pos-[abssumx,abssumy]);
        else
            summeabs=100;
        end
    end
```

Mine auf dem Weg???

```
function wegminex=wegmin(zp) % Nächste Mine Kollision in Richtung Zielpunkt---liegt Mine auf dem Weg????
    if spiel.n_mine>0
        wegminematrix=zeros(spiel.n_mine,1);

        for x=1:spiel.n_mine

            wegminematrix(x,1)=x;
            wegminematrix(x,2)=norm(spiel.mine(x).pos-ich.pos);
            wegminematrix(x,3)=spiel.mine(x).pos(1);
            wegminematrix(x,4)=spiel.mine(x).pos(2);
            wegminematrix(x,5)=szklx(ich.pos,zp-ich.pos,spiel.spaceball_radius,spiel.mine(x).pos,[0,0],spiel.mine_radius);
            wegminematrix(x,6)=wegminematrix(x,2)*kmw1+wegminematrix(x,5)*kmw2;
        end

        wegminematrix1=sortrows(wegminematrix,6);

        wegminepos=[wegminematrix1(1,3),wegminematrix1(1,4)];

        wegminex=wegminematrix1(1,1);
    else
        wegminex=1;
    end
end
```

Mine auf dem Weg??? 2

```
function wegminex2=wegmin2(zp) % Nächste Mine Kollision in Richtung Zielpunkt---liegt Mine auf dem Weg????
    if spiel.n_mine>0
        wegminematrix2=zeros(spiel.n_mine,1);

        for x=1:spiel.n_mine

            wegminematrix2(x,1)=x;
            wegminematrix2(x,2)=szklx(ich.pos,zp-ich.pos,spiel.spaceball_radius,spiel.mine(x).pos,[0,0],spiel.mine_radius);
            wegminematrix2(x,3)=spiel.mine(x).pos(1);
            wegminematrix2(x,4)=spiel.mine(x).pos(2);

        end

        wegminematrix2_1=sortrows(wegminematrix2,2);

        wegminex2=wegminematrix2_1(1);
    end
end
```

Mine auf dem Weg 3

```
function wegminex3=wegmin3 % Nächste Mine Kollision in Richtung Zielpunkt---liegt Mine auf dem Weg????
    if spiel.n_mine>0
        wegminematrix3=zeros(spiel.n_mine,1);

        for x=1:spiel.n_mine

            wegminematrix3(x,1)=x;
            wegminematrix3(x,2)=szklx2(ich.pos,ich.ges,spiel.spaceball_radius,spiel.mine(x).pos,[0,0],spiel.mine_radius);
            wegminematrix3(x,3)=spiel.mine(x).pos(1);
            wegminematrix3(x,4)=spiel.mine(x).pos(2);

        end

        wegminematrix3_1=sortrows(wegminematrix3,2);

        wegminex3=wegminematrix3_1(1);
    end
end
```

Tangentenverfahren

```
function tng=tangver(zielp,nmp)
    g1m=nmp-ich.pos;
    g2z=zielp-ich.pos;

    tng=(vp(zielp,ich.pos,g1m)/(norm(vp(zielp,ich.pos,g1m))))*0.07;

end
```

Angriffsfunktion

```
function besangriff1=ang
    zp=gegner.pos;
    zpges=gegner.ges;
    zpradius=0.01;

    if spiel.n_mine>0
        agerademinematrix=zeros(spiel.n_mine,1);
        for x=1:spiel.n_mine;

            agerademinematrix(x,1)=x;
            agerademinematrix(x,2)=norm(spiel.mine(x).pos-ich.pos);
            agerademinematrix(x,3)=spiel.mine(x).pos(1);
            agerademinematrix(x,4)=spiel.mine(x).pos(2);
```

```
            agerademinematrix(x,5)=szklx2(ich.pos,ich.ges,spiel.spaceball_radius,spiel.mine(x).pos,[0,0],spiel.mine_radius+0.06);


        end


        agerademinem=sortrows(agerademinematrix,5);

        nextagerademinepos=[agerademinem(1,3),agerademinem(1,4)];


        if szklx(ich.pos, vpba(nextagerademinepos,ich.pos,ich.ges), spiel.spaceball_radius, nextagerademinepos, [0,0], (spiel.mine_radius+norm(ich.ges)*5))
            nomine=1;
        else
            nomine=0;
        end
    end
    % Angriff1
    besangriff1=(gegner.pos+1*gegner.ges)-ich.pos-ich.ges;   %gegner.pos-ich.pos;

    dang=1;

    if norm(ich.ges)>0.25
        % Angriff 2
        besangriff1=vp((gegner.pos+2*gegner.ges),ich.pos,ich.ges);

        dang=1;
        lambdagegich1=szk(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,spiel.spaceball_radius);

        lambdagegich=lambdagegich1(1);

        if lambdagegich>0.001 && abs_geg<0.15
            % Angriff 3
            besangriff1=(gegner.pos+gegner.ges)-ich.pos;

        end

    elseif abs_geg<0.15   %% wenn abstand gegner klein wird dann diese Modus%%
        % Angriff 4
        besangriff1=gegner.pos+gegner.ges-ich.pos;

        dang=1;
    end


    if norm(ich.ges)>0.01
        tecgeg=szk(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,[0,0],0.0008);
        zeitgeg=tecgeg(1);
        if spiel.n_mine>0
            if zeitgeg<13 && zeitgeg>0 && norm(ich.ges)>0.1 && wegmin2(gegner.pos)==inf

                persistent geradegeg;
                geradegeg=gegner.pos-ich.pos;
                % Angriff2
                besangriff2=gegner.bes;

                dang=1;



                if norm(ich.ges)<0.5 & szklx(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,spiel.spaceball_radius)<1;
                    % Angriff 22
                    besangriff2=gegner.pos-ich.pos;
                    dang=1;
                    if norm(ich.ges)>=0.5 & szklx(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,spiel.spaceball_radius)<1;
                        if wegmin2(gegner.pos)~=inf
                            % Angriff 22
                            besangriff2=(gegner.pos+gegner.bes)-ich.pos;

                        end
                    end
                end

            else
                tecgeg2=szklx2(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,0.0008);
                zeitgeg2=tecgeg2(1);
                if zeitgeg2<13 && zeitgeg2>0 && norm(ich.ges)>0.1 && spiel.n_mine<13 && norm(ich.ges)>norm(gegner.ges)*1.5 && nomine==1

                    Angriff6=true;
                    % Angriff 6
                    besangriff1=gegner.bes;

                    dang=1;

                    if norm(ich.ges)<0.5 & szklx(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,spiel.spaceball_radius)<1;
                       % Angriff 22
                        besangriff2=gegner.pos-ich.pos;
                        dang=1;
                        if norm(ich.ges)>=0.5 & szklx(ich.pos,ich.ges,spiel.spaceball_radius,gegner.pos,gegner.ges,spiel.spaceball_radius)<1;
                            if wegmin2(gegner.pos)~=inf
                                % Angriff 22
                                besangriff2=(gegner.pos+gegner.bes)-ich.pos;
                            end
                        end
                    end
                end
            end
        end
```

```
    end

    if Angriff6==true
        besangriff1=gegner.bes;
    else
        Angriff6=false;
    end

end
```

Eckenverteidigung

```
function besdef=verd

    zp=ich.pos+(ich.ges/norm(ich.ges)*1);
    zpges=[0,0];
    zpradius=0.01;

    besdef=[0,0];



    nexteckematrix=zeros(4,3);

    nexteckematrix(1,1)=eol(1);
    nexteckematrix(2,1)=eor(1);
    nexteckematrix(3,1)=eur(1);
    nexteckematrix(4,1)=eul(1);
    nexteckematrix(1,2)=eol(2);
    nexteckematrix(2,2)=eor(2);
    nexteckematrix(3,2)=eur(2);
    nexteckematrix(4,2)=eul(2);
    nexteckematrix(1,3)=norm(eol-ich.pos);
    nexteckematrix(2,3)=norm(eor-ich.pos);
    nexteckematrix(3,3)=norm(eur-ich.pos);
    nexteckematrix(4,3)=norm(eul-ich.pos);
    nexteckematrix(1,4)=szklx2(ich.pos,vpba(eol,ich.pos,ich.ges),spiel.spaceball_radius,eol,[0,0],0.01);
    nexteckematrix(2,4)=szklx2(ich.pos,vpba(eor,ich.pos,ich.ges),spiel.spaceball_radius,eor,[0,0],0.01);
    nexteckematrix(3,4)=szklx2(ich.pos,vpba(eur,ich.pos,ich.ges),spiel.spaceball_radius,eur,[0,0],0.01);
    nexteckematrix(4,4)=szklx2(ich.pos,vpba(eul,ich.pos,ich.ges),spiel.spaceball_radius,eul,[0,0],0.01);

    nexteckematrix1=sortrows(nexteckematrix,3);

    nexteckepos=[nexteckematrix1(1,1),nexteckematrix1(1,2)];
    nexteckepos2=[nexteckematrix1(2,1),nexteckematrix1(2,2)];

    nexteckegegmatrix=zeros(4,3);

    nexteckegegmatrix(1,1)=eol(1);
    nexteckegegmatrix(2,1)=eor(1);
    nexteckegegmatrix(3,1)=eur(1);
    nexteckegegmatrix(4,1)=eul(1);
    nexteckegegmatrix(1,2)=eol(2);
    nexteckegegmatrix(2,2)=eor(2);
    nexteckegegmatrix(3,2)=eur(2);
    nexteckegegmatrix(4,2)=eul(2);
    nexteckegegmatrix(1,3)=norm(eol-gegner.pos+gegner.ges);
    nexteckegegmatrix(2,3)=norm(eor-gegner.pos+gegner.ges);
    nexteckegegmatrix(3,3)=norm(eur-gegner.pos+gegner.ges);
    nexteckegegmatrix(4,3)=norm(eul-gegner.pos+gegner.ges);

    nexteckegegmatrix1=sortrows(nexteckegegmatrix,3);

    nexteckegegpos=[nexteckegegmatrix1(3,1),nexteckegegmatrix1(3,2)];



    besdef=nexteckepos-ich.pos;

    if norm(nexteckepos-ich.pos)<0.069
        nexteckepkt=1;
    end

    if nexteckepkt==1
        abdeft=5;
    else
        abdeft=2;
    end

    if szklx2(gegner.pos,gegner.ges,spiel.spaceball_radius,ich.pos,ich.ges,spiel.spaceball_radius)<abdeft
        t_ausw=true;
    else
        t_ausw=false;
    end

    if nexteckepkt==1

        eckverd=1;

        bv=1.1;

        abdef =0.12;

        if norm(eol-ich.pos)<abdef && norm(gegner.ges(1))>norm(gegner.ges(2))&& t_ausw==true
```

```
            zpverd=[0,-1];
        end

        if norm(eol-ich.pos)<abdef && norm(gegner.ges(1))<norm(gegner.ges(2))&& t_ausw==true
            zpverd=[1,0];
        end

        if norm(eor-ich.pos)<abdef && norm(gegner.ges(1))>norm(gegner.ges(2))&& t_ausw==true
            zpverd=[0,-1];
        end
        if norm(eor-ich.pos)<abdef && norm(gegner.ges(1))<norm(gegner.ges(2))&& t_ausw==true
            zpverd=[-1,0];
        end

        if norm(eur-ich.pos)<abdef && norm(gegner.ges(1))>norm(gegner.ges(2))&& t_ausw==true
            zpverd=[0,1];
        end
        if norm(eur-ich.pos)<abdef && norm(gegner.ges(1))<norm(gegner.ges(2))&& t_ausw==true
            zpverd=[-1,0];
        end

        if norm(eul-ich.pos)<abdef  && norm(gegner.ges(1))>norm(gegner.ges(2))&& t_ausw==true
            zpverd=[0,1];
        end
        if norm(eul-ich.pos)<abdef  && norm(gegner.ges(1))<norm(gegner.ges(2))&& t_ausw==true
            zpverd=[1,0];
        end



        besdef=zpverd+[0,0];
    else
        zpverd=[0,0];
        besdef=zpverd+[0,0]+nexteckepos-ich.pos-ich.ges;
    end

end
```

Tankfunktion

```
function bestanken1=tanken
    if spiel.n_tanke>0

        % Erstellen der Tankenmatrix
        nexttankematrix=zeros(spiel.n_tanke,1);
        for x=1:spiel.n_tanke;

            lambdax=szk(ich.pos,ich.ges,spiel.spaceball_radius,spiel.tanke(x).pos,[0,0],spiel.tanke_radius);
            lambdax1=lambdax(1);

            nexttankematrix(x,1)=x;
            nexttankematrix(x,2)=norm(spiel.tanke(x).pos-ich.pos);
            nexttankematrix(x,3)=spiel.tanke(x).pos(1);
            nexttankematrix(x,4)=spiel.tanke(x).pos(2);
            nexttankematrix(x,5)=szktanken1(ich.pos,vpba([nexttankematrix(x,3),nexttankematrix(x,4)],ich.pos,ich.ges),spiel.spaceball_radius,spiel.tanke(x)
            nexttankematrix(x,6)=0;
            nexttankematrix(x,7)= 0;

            %nexttankematrix(x,7)=dot(spiel.tanke(x).pos-ich.pos,ich.ges);
            nexttankematrix(x,8)=sumabs(nexttankematrix(x,1));
            nexttankematrix(x,9)=0;
            if spiel.n_tanke>0 && spiel.n_mine>0
                xtanke=wegmin([nexttankematrix(x,3),nexttankematrix(x,4)]);
                xtankeges=[nexttankematrix(x,3),nexttankematrix(x,4)]-ich.pos;

                szktank=szktanken1(ich.pos,xtankeges,spiel.spaceball_radius,spiel.mine(xtanke).pos,[0,0],spiel.mine_radius);

                if szktank* norm(vpba(spiel.mine(xtanke).pos,ich.pos,ich.ges))<norm([nexttankematrix(x,3),nexttankematrix(x,4)]-ich.pos)
                    nexttankematrix(x,9)=1;
                else
                    nexttankematrix(x,9)=0;
                end
            end

            % Alternative Gütegrade %

            %sehr gut!!!!!!!!!!                nexttankematrix(x,10)=abs(nexttankematrix(x,2)*100+nexttankematrix(x,5)+(nexttankematrix(x,9))*10)+nexttank
            %gut!!!       %% nexttankematrix(x,10)=abs(nexttankematrix(x,2)*1000+nexttankematrix(x,5)*10+(nexttankematrix(x,9)))   %%+nexttankematrix(x,8)*

            nexttankematrix(x,10)=abs(nexttankematrix(x,2)*500+nexttankematrix(x,5)*6+(nexttankematrix(x,9))*80);
        end
        % Sortieren der Tankenmatrix nach dem kleinsten Abstand
        nexttankematrix1=sortrows(nexttankematrix,10);

        % Bestimmung der nächsten Tankenkoordinaten
        nexttankepos=[nexttankematrix1(1,3),nexttankematrix1(1,4)];
        zp=nexttankepos;
        zpges=[0,0];
        zpradius=0.01;

        lambdatankeich1=szk(ich.pos, vpba(nexttankepos,ich.pos,ich.ges), spiel.spaceball_radius, nexttankepos, [0,0], spiel.tanke_radius);
        lambdatankeich=lambdatankeich1(1);

        lambdatankegegner1=szk(gegner.pos, vpba(nexttankepos,gegner.pos, gegner.ges), spiel.spaceball_radius, nexttankepos, [0,0], spiel.tanke_radius);
        lambdatankegegner=lambdatankegegner1(1);
```

```
if spiel.n_tanke>1
    geradetankematrix=zeros(spiel.n_tanke,1);
    for x=1:spiel.n_tanke;

        geradetankematrix(x,1)=x;
        geradetankematrix(x,2)=norm(spiel.tanke(x).pos-ich.pos);
        geradetankematrix(x,3)=spiel.tanke(x).pos(1);
        geradetankematrix(x,4)=spiel.tanke(x).pos(2);
        geradetankematrix(x,5)=dot(spiel.tanke(x).pos-ich.pos,ich.ges);
        geradetankematrix(x,6)=szklx2(ich.pos,ich.ges,spiel.spaceball_radius,spiel.tanke(x).pos,[0,0],spiel.tanke_radius);

    end


    geradetankem=sortrows(geradetankematrix,6);

    if  spiel.n_tanke>1 && geradetankem(1,6)<100 && geradetankem(2,6)<100 && wegmin3>szklx2(ich.pos,ich.ges,spiel.spaceball_radius,[geradetankem(2,
        && norm([geradetankem(2,3),geradetankem(2,4)]-ich.pos)<0.5
        % Tanken Gerade 1
        nexttankepos=[geradetankem(2,3),geradetankem(2,4)];
        zp=nexttankepos;
        zpges=[0,0];
        zpradius=0.01;

        gerade=1;
    end

    if spiel.n_tanke>2 && geradetankem(1,6)<100 && geradetankem(2,6)<100 && geradetankem(3,6)<100 && wegmin3>szklx2(ich.pos,ich.ges,spiel.spaceball
        && norm([geradetankem(3,3),geradetankem(3,4)]-ich.pos)<0.5
        % Tanken Gerade 2
        nexttankepos=[geradetankem(3,3),geradetankem(3,4)];
        zp=nexttankepos;
        zpges=[0,0];
        zpradius=0.01;

    end
end


if spiel.n_tanke>1

    % Bestimmung der 2. nächsten Tankenkoordinaten
    nexttankepos2=[nexttankematrix1(2,3),nexttankematrix1(2,4)];
    %                 zp=nexttankepos2;
    %                 zpges=[0,0];
    %                 zpradius=0.01;



elseif spiel.n_tanke==1 && lambdatankeich1(1)>(lambdatankegegner1(1)*0.98) && ich.getankt==4 && gegner.getankt==4
    bestanken=verd;
end


if ich.getankt<9
    % Tanken 1
    bestanken1=nexttankepos-ich.pos-ich.ges;


    if norm(nexttankepos-ich.pos-ich.ges)<0.05
        % Tanken 1 1
        bestanken1=nexttankepos-ich.pos;


    else
        bestanken1=nexttankepos-ich.pos-ich.ges;
    end

end


if ich.getankt<5 && lambdatankeich>lambdatankegegner && spiel.n_tanke>1 && geradetanke==0
    % Tanken 1 2
    bestanken1=nexttankepos2-ich.pos-ich.ges;

    zp=nexttankepos2;
    zpges=[0,0];
    zpradius=0.01;


    if norm(nexttankepos2-ich.pos-ich.ges)<0.05
        % Tanken 1 2 1
        bestanken1=nexttankepos2-ich.pos;

    else
        bestanken1=nexttankepos2-ich.pos-ich.ges;
    end
end
if norm(ich.pos-nexttankepos)<0.035
    % Tanken 2
    bestanken1=nexttankepos-ich.pos;

end
if gerade ==1;
    bestanken1=ich.ges;

end
```

```
        % Spontanangriff
        if gegner.getankt<ich.getankt && abs_geg<0.06 && norm(ich.ges)<0.1
            bestanken1=gegner.pos-ich.pos;
        end

        % Spontanverteidigung 0
        if gegner.getankt>ich.getankt && abs_geg<0.05
            bestanken1=vp(ich.pos,gegner.pos,gegner.ges);
        end
    end
end
```

Minenverteidigung

```
function minendef1=md
    minendef1=[0,0];

    nextminenmatrix=zeros(spiel.n_mine,1);
    for x=1:spiel.n_mine;
        nextminenmatrix(x,1)=x;
        nextminenmatrix(x,2)=norm(spiel.mine(x).pos-ich.pos);
        nextminenmatrix(x,3)=spiel.mine(x).pos(1);
        nextminenmatrix(x,4)=spiel.mine(x).pos(2);
        mm1=szk(ich.pos, ich.ges, spiel.spaceball_radius, spiel.mine(x).pos, [0,0], spiel.mine_radius+0.002); %0,0006
        nextminenmatrix(x,5)=mm1(1);
        nextminenmatrix(x,6)=mm1(2);
        nextminenmatrix(x,7)=nextminenmatrix(x,2)*nextminenmatrix(x,5);
    end

    nextminenmatrix1=sortrows(nextminenmatrix,7);
    nextminenmatrix1;


    nextminenmatrix1(1,3);
    nextminenmatrix1(1,4);


    nextminepos=[nextminenmatrix1(1,3),nextminenmatrix1(1,4)];


    if spiel.n_mine>0
        gerademinematrix=zeros(spiel.n_mine,1);
        for x=1:spiel.n_mine;

            gerademinematrix(x,1)=x;
            gerademinematrix(x,2)=norm(spiel.mine(x).pos-ich.pos);
            gerademinematrix(x,3)=spiel.mine(x).pos(1);
            gerademinematrix(x,4)=spiel.mine(x).pos(2);
            gerademinematrix(x,5)=szklx2(ich.pos,ich.ges,spiel.spaceball_radius,spiel.mine(x).pos,[0,0],spiel.mine_radius+0.06);


        end


        gerademinem=sortrows(gerademinematrix,5);

        nextgerademinepos=[gerademinem(1,3),gerademinem(1,4)];


        if szklx2(ich.pos, ich.ges, spiel.spaceball_radius, nextgerademinepos, [0,0], (spiel.mine_radius+0.007))<100
            minendang=1;
        end
    end

    % Tangentenverfahren

    zeit_nextminetang=szk(ich.pos, vpba(nextminepos,ich.pos,ich.ges), spiel.spaceball_radius, nextminepos, [0,0], spiel.mine_radius);
    zeit_zptang=szk(ich.pos,vpba(zp,ich.pos,ich.ges),spiel.spaceball_radius,zp,zpges,zpradius);
    zeit_zpt=zeit_zptang(1);
    zeit_nextmine0=szklx(ich.pos, ich.ges, spiel.spaceball_radius, nextminepos, [0,0], spiel.mine_radius+0.00011);


    bwmineba=(norm(vpba(nextminepos,ich.pos,vpba(nextminepos,ich.pos,ich.ges)))^2)/(2*spiel.bes);
    bwmine = (norm(vpba(nextminepos,ich.pos,ich.ges))^2)/(2*spiel.bes);



    if norm(nextminepos-ich.pos)-0.0601< bwmineba*3 && zeit_nextminetang(1)<zeit_zpt %&& zeit_nextmine0
        Angriff6=false;

        if eckverd==0
            bestanken=[0,0];
            besangriff=[0,0];
            besdefence=[0,0];

            minendef1=tangver(zp,nextminepos);

        end
    end

    % Minennotabwehrvektor

    %((((1))))%

    zeit_nextmine=szklx(ich.pos, ich.ges, spiel.spaceball_radius, nextminepos, [0,0], spiel.mine_radius+0.0027);
```

```
        bwmineba=(norm(vpba(nextminepos,ich.pos,vpba(nextminepos,ich.pos,ich.ges)))^2)/(2*spiel.bes);
        bwmine = (norm(vpba(nextminepos,ich.pos,ich.ges))^2)/(2*spiel.bes);

        norm(vpba(nextminepos,ich.pos,ich.ges));

        if norm(nextminepos-ich.pos)-0.0601< bwmineba*1.1 &&...
                zeit_nextmine<100;
            Angriff6=false;
            if eckverd==0
                bestanken=[0,0];
                besangriff=[0,0];
                besdefence=[0,0];
                minendef1=ich.pos-nextminepos;

            end
        end

    end
```

%%

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Modusentscheidung%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%%%

Tank-Modus

```
if ich.getankt<10 && gegner.getankt<5 && spiel.n_tanke>0

    bestanken=tanken;

end
```

Angriffs-Modus

```
if ich.getankt>=5 && spiel.n_tanke==0

    bestanken=[0,0];
    besangriff=ang;

end
```

Verteidigung

Ausweichen nach dem Tanken, wenn der Gegner mehr Tanken gesammelt hat als ich

```
if gegner.getankt>=5

    Angriff6=false;
    bestanken=[0,0];
    besangriff=[0,0];
    besdefence=verd;

end
```

Minenverteidigung

```
if spiel.n_mine>0
    besmine=md;
else
    besmine=[0,0];

end
```

Bremswegminenberechnung

```
if spiel.n_mine>0

    nextminenmatrixsf=zeros(spiel.n_mine,1);

    for n=1:spiel.n_mine;
        nextminenmatrixsf(n,1)=n;
        nextminenmatrixsf(n,2)=norm(spiel.mine(n).pos-ich.pos+0.001);
    end
    nextminenmatrixsf1=sortrows( nextminenmatrixsf,2);
    nextminesf=nextminenmatrixsf1(1,2);

    if nextminesf<bwmineba
        sonderbw=1;
    end
end
```

Ecken- und Bandenverteidigung

```
%Sonderfall
vecichges=1.6*(ich.ges/norm(ich.ges))*norm(bw);

%rechte Bande
absbwichges1=norm(vpba([1,ich.pos(2)],ich.pos,vecichges));
%obere Bande
absbwichges2=norm(vpba([ich.pos(1),1],ich.pos,vecichges));
%linke Bande
absbwichges3=norm(vpba([0,ich.pos(2)],ich.pos,vecichges));
%untere Bande
absbwichges4=norm(vpba([ich.pos(1),0],ich.pos,vecichges));

%rechte Bande
if (1.6*absbwichges1)>(abs_rb-0.0101) && ich.ges(1)>0
    sonderfall_bande=1;

end
%obere Bande
if  (1.6*absbwichges2)>(abs_ob-0.0101) && ich.ges(2)>0
    sonderfall_bande=1;

end
%linke Bande
if (1.6*absbwichges3)>(abs_lb-0.0101) && ich.ges(1)<0
    sonderfall_bande=1;

end
%untere Bande
if (1.6*absbwichges4)>(abs_ub-0.0101) && ich.ges(2)<0
    sonderfall_bande=1;

end

% Normale Bandenverteidigung

bes1=[0,0];
bes2=[0,0];
bes3=[0,0];
bes4=[0,0];

if Angriff6==false

    % Bandenprotektion rechte Bande
    if abs_rb<=bwx*bv && ich.ges(1)>0
        bestanken=[0,0];
        besangriff=[0,0];
        besdefence=[0,0];
        bes1=[-1,0];

    end

    % Bandenprotektion obere Bande
    if abs_ob<=bwy*bv && ich.ges(2)>0
        bestanken=[0,0];
        besangriff=[0,0];
        besdefence=[0,0];
        bes2=[0,-1];

    end

    % Bandenprotektion linke Bande
    if abs_lb<=bwx*bv && ich.ges(1)<0
        bestanken=[0,0];
        besangriff=[0,0];
        besdefence=[0,0];
        bes3=[1,0];

    end

    % Bandenprotektion untere Bande
    if abs_ub<=bwy*bv && ich.ges(2)<0
        bestanken=[0,0];
        besangriff=[0,0];
        besdefence=[0,0];
        bes4=[0,1];

    end
end

if nexteckepkt==1
    dtad=1.6;

    if abs_ob<=bwy*dtad && ich.ges(2)>0 && abs_lb<=bwx*dtad && ich.ges(1)<0
        bes2=[0,-1];
        bes3=[1,0];
    end
    if abs_ob<=bwy*dtad && ich.ges(2)>0 && abs_rb<=bwx*dtad && ich.ges(1)>0
        bes2=[0,-1];
        bes1=[-1,0];
    end
    if abs_ub<=bwy*dtad && ich.ges(2)<0 && abs_lb<=bwx*dtad && ich.ges(1)<0
        bes4=[0,1];
        bes3=[1,0];
    end
    if abs_ub<=bwy*dtad && ich.ges(2)<0 && abs_rb<=bwx*dtad && ich.ges(1)>0
        bes4=[0,1];
        bes1=[-1,0];
```

```
    end
end
```

Zusammenführung aller möglichen Teilbeschleunigungen

```
if  sonderfall_bande==1   && minendang==1
    bes=-ich.ges;
else
    bes=bes1+bes2+bes3+bes4+bestanken+besangriff+besdefence+besmine;
end
```

```
end
```

Published with MATLAB® R2016a